PIMD: User's Guide

Version 2.8.0

M. Shiga (*Japan Atomic Energy Agency*)

shiga.motoyuki@jaea.go.jp

November 11, 2026

PIMD is an open-source Fortran 90 package with Message Passing Interface (MPI) support for molecular simulations on parallel computers. It provides methods ranging from static calculations, geometry and normal-mode analysis, reaction-path searches, and classical molecular dynamics to path- integral methods, replica-exchange and hybrid Monte Carlo methods, metadynamics, and nonadiabatic dynamics.

The code supports classical force fields, machine-learning potentials, *ab initio* electronic-structure calculations, and hybrid potentials such as ONIOM and QM/MM. External electronic-structure and machine-learning programs can be used through the interfaces described in this manual.

PIMD is distributed under the Apache License, Version 2.0.


```
Copyright 2016--2026  M. Shiga.  All rights reserved.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```


The code is tested and maintained to the best of our ability. If you find a bug, please report it with enough information to reproduce the problem. Because the software is provided without warranty, users remain responsible for validating results for their own applications. We may not be able to answer every request for detailed application-specific advice.

Recent changes

Version 2.7.0 introduced the N2P2 interface for neural-network potentials and an interface to VASP 6. Version 2.7.0.r2 updated the build configuration in response to the discontinuation of the Intel `ifort` compiler.

Version 2.7.1 introduced the local parallel-computation scheme (XMPI) for path-integral simulations, added an interface to PFP universal machine-learning potentials, and added the GAL21 force field for metal–water interfaces [50]. Hardware-dependent behavior of the random-number generator used by BCMD and TRPMD was also corrected.

Version 2.7.2 added the MACE interface, including GPU acceleration through CUDA-enabled PyTorch for molecular-dynamics and path-integral simulations.

Version 2.8.0 reorganizes the distribution around a top-level CMake build and a curated example suite. A major user-facing change is the TOML-first input workflow: the distributed examples use `pimd.toml` as the authoritative input, and **`pimd-prep`** validates it and generates the legacy `input.dat` consumed by the Fortran engine. The introductory documentation emphasizes six representative workflows—PIMD, RPMD, BCMD, SLHMC, QTST with **`polymers.x`**, and metadynamics—while retaining the full set of specialized methods. Output conventions for `final.xyz` and path-integral `trj.xsf` were also clarified and standardized. This manual is synchronized with **`pimd-prep`** version 0.9.7.

Contents

I Getting Started 20

1 Installation and compilation 20

1.1 Build requirements . . . 20
1.2 Standard out-of-source build . . . 20
1.3 Choosing compilers and MPI . . . 21
1.4 Math libraries . . . 21
1.5 Executable targets . . . 22
1.6 Optional interfaces . . . 22
1.7 Running the test suite . . . 23
1.8 Checking and diagnosing a build . . . 24
1.9 Legacy makefile build . . . 24
1.10 Next step: the TOML-first workflow . . . 25

2 TOML input and `pimd-prep` 26

2.1 Running `pimd-prep` 0.9.9 . . . 26
2.2 A minimal example . . . 27
2.3 Structured settings . . . 27
2.3.1 Ensemble-specific thermostat and barostat settings . . . 28
2.4 Readable values and units . . . 28
2.4.1 Metadynamics TOML . . . 29
2.4.2 Built-in WATER models . . . 29
2.4.3 Structured constraints and TASS collective variables . . . 29
2.4.4 Named L-BFGS convergence thresholds . . . 30
2.4.5 Phonon-dispersion wave vectors . . . 30
2.4.6 SMASH options . . . 31
2.4.7 Rigid molecular components . . . 31
2.5 The `[files]` manifest . . . 31
2.6 Defaults and the generated `input.dat` . . . 32
2.7 Using `[legacy]` . . . 32
2.8 Validation, boundary conditions, and migration . . . 32
2.8.1 Required, conditional, optional, and generated runtime artifacts . . . 32
2.9 Workflow aliases . . . 34
2.10 Running the distributed examples . . . 34

3 Compatibility with older input formats 34

3.1 Docker . . . 35

II Simulation Methods 36

4 Methods **36**

4.1 Static calculation (STATIC) . . . 36
4.2 Geometry optimization (GEOOPT) . . . 39
4.3 Restarts and interruptions . . . 40
4.4 Normal mode analysis (NMA) . . . 41
4.5 Steepest descent (SD) . . . 42
4.6 Box optimization (BOXOPT) . . . 42
4.7 Full optimization (FULLOPT) . . . 43
4.8 Static elastic constants (ELASTIC) . . . 44
4.9 Classical molecular dynamics (MD) . . . 44
4.9.1 NVE molecular dynamics . . . 44
4.9.2 NVT molecular dynamics . . . 46
4.9.3 Constrained molecular dynamics . . . 47
4.9.4 NPT molecular dynamics in a cubic box . . . 48
4.9.5 NPT and NtT molecular dynamics in a hexahedron box . . . 50
4.10 Path integral simulations (PI) . . . 53
4.10.1 Path integral molecular dynamics (PIMD) . . . 53
4.10.2 Ring polymer molecular dynamics (RPMD) . . . 56
4.10.3 Brownian chain molecular dynamics (BCMD) . . . 57
4.10.4 Centroid molecular dynamics (CMD) . . . 57
4.10.5 Thermostatted ring polymer molecular dynamics (TRPMD) . . . 60
4.10.6 Path integral hybrid Monte Carlo (PIHMC) . . . 60
4.10.7 Centroid Intrinsic Reaction Coordinate (CIRC) . . . 62
4.11 Self-learning hybrid Monte Carlo (SLHMC) . . . 64

5 Quantum transition-state free energy (QTST with `polymers.x`) **66**

5.1 String method (STRING) . . . 67
5.2 Metadynamics (MTD) . . . 68
5.3 Replica exchange hybrid Monte Carlo (REHMC) . . . 73
5.3.1 Temperature replica exchange hybrid Monte Carlo . . . 74
5.3.2 Hamiltonian replica exchange hybrid Monte Carlo . . . 75
5.4 Mean force dynamics (AFED) . . . 77
5.4.1 Common keywords used in all the AFED methods (Free energy gradient) 78
5.4.2 Descent search for free energy minima . . . 81
5.4.3 Ascent search for free energy saddles . . . 82
5.4.4 Automated search for free energy stationary points . . . 83
5.4.5 Temperature accelerated molecular dynamics . . . 84
5.4.6 Logarithmic mean force dynamics . . . 86
5.5 Phonon calculation (PHONON) . . . 88
5.6 Optimization of OM action (OMOPT) . . . 90
5.7 User-made potentials . . . 91

5.7.1 Testing forces . . . 92
5.7.2 Testing virial . . . 93
5.7.3 Testing Ewald parameters . . . 93
5.8 Mechanochemistry . . . 94
5.9 Nonadiabatic dynamics . . . 95
5.9.1 Surface hopping dynamics . . . 95
5.9.2 Mean field dynamics . . . 98
5.10 Molecular dynamics of rigid molecules . . . 99
5.10.1 Gentlest ascent dynamics (GAD) . . . 101
5.11 Consecutive static calculations (SCAN) . . . 102

III Potential Models and Electronic-Structure Interfaces 102

6 Potentials 102

6.1 Empirical force fields . . . 103
6.1.1 TIP4P through CFF . . . 103
6.1.2 Embedded atom method (EAM) . . . 103
6.1.3 Angular-dependent potential (ADP) . . . 104
6.1.4 Tabulated pair potential . . . 104
6.1.5 Tersoff potential . . . 104
6.1.6 GAL21 potential . . . 105
6.2 User-defined potentials . . . 105
6.3 Born–Oppenheimer potentials . . . 105
6.3.1 SMASH . . . 106
6.3.2 ABINIT-MP . . . 107
6.3.3 CP2K . . . 107
6.3.4 DFTB – Internal . . . 111
6.3.5 DFTB - External . . . 113
6.3.6 GAMESS . . . 115
6.3.7 GAUSSIAN . . . 116
6.3.8 MOLPRO . . . 117
6.3.9 MOPAC . . . 119
6.3.10 NTCHEM . . . 120
6.3.11 ORCA . . . 121
6.3.12 PHASE/0 . . . 122
6.3.13 QUANTUM ESPRESSO . . . 124
6.3.14 OpenMX 4.0.1 . . . 126
6.3.15 MBX . . . 129
6.3.16 TURBOMOLE . . . 130
6.3.17 VASP6 Interface . . . 133
6.3.18 VASP - General Words of Caution . . . 134

6.3.19 VASP5 . . . 134
6.4 Machine learning potentials . . . 136
6.4.1 AENET . . . 136
6.4.2 AccelNet . . . 141
6.4.3 MTP . . . 147
6.4.4 N2P2 . . . 147
6.4.5 MACE . . . 150
6.4.6 ASE calculator interface . . . 154
6.4.7 PFP . . . 156

IV Hybrid Methods, Analysis, and Utilities 156

7 Hybrid potentials 156
7.1 QM/MM . . . 156
7.1.1 QM/MM setup . . . 157
7.1.2 QM/MM with SMASH . . . 158
7.1.3 QM/MM with other codes . . . 159
7.1.4 QM/MM with BEST method . . . 159
7.1.5 Multiple time scale QM/MM method . . . 160
7.2 ONIOM . . . 161
7.2.1 ONIOM setup . . . 161
7.2.2 ONIOM with any codes . . . 163
7.2.3 ONIOM with BEST method . . . 164

8 Data analysis 164
8.1 Visualization of trajectory . . . 164
8.2 Statistical analysis . . . 166

9 Codes 166
9.1 Main codes . . . 166
9.2 Codes for post-processing . . . 168
9.3 Subsidiary codes . . . 168
9.4 Utility codes . . . 168
9.5 External calls . . . 168

V Classical Force Fields, Input, and Output 168

10 Classical force field (CFF) 169
10.1 LAMMPS-compatible input . . . 169
10.1.1 Files and division of responsibility . . . 169
10.1.2 Commands interpreted by CFF . . . 169
10.1.3 Supported styles and coefficient forms . . . 170

10.1.4 Commands accepted for a shared LAMMPS file but not executed 171
10.1.5 data.lammps: topology actually used by CFF 172
10.1.6 1–2, 1–3, and 1–4 exclusions/scaling 172
10.1.7 PIMD-only CFF extensions 172
10.1.8 Example: water and the auto-generated q-TIP4P/F files 173
10.1.9 Example: alanine dipeptide (dialanine) 173
10.1.10 Example: EAM/alloy input for a Ni–H system 174
10.1.11 Unsupported input is fail-closed 175
10.2 Static preflight and cross-file consistency with pimd-prep 175
10.3 CFF with XMPI 175
10.4 TINKER conversion 175
10.5 How to convert TINKER files 176
10.6 Preparation of initial structure 176

11 Keywords **179**
11.1 Run-definition keywords in input.dat 179
11.1.1 <ensemble> 179
11.1.2 <ipotential> 180
11.1.3 <method> 181
11.1.4 <natom> 182
11.1.5 <nspec> 182
11.2 Primary simulation-control keywords in input.dat 183
11.2.1 <bath_type> 183
11.2.2 <dt> 183
11.2.3 <iboundary> 183
11.2.4 <iprint_rest> 184
11.2.5 <iprint_std> 184
11.2.6 <iprint_trj> 184
11.2.7 <iread_exit> 184
11.2.8 <nbead> 184
11.2.9 <nstep> 185
11.2.10 <temperature> 185
11.3 Additional common simulation-control keywords in input.dat 185
11.3.1 <beadspread> 185
11.3.2 <corrections> 185
11.3.3 <input_style> 186
11.3.4 <iprint_dcd> 186
11.3.5 <iprint_xsf> 186
11.3.6 <iprint_xyz> 186
11.3.7 <iprint_extxyz> 186
11.3.8 <ncolor> 186

11.3.9 <nnhc> . . . 187
11.3.10 <np_beads> . . . 187
11.3.11 <np_force> . . . 187
11.3.12 <nref> . . . 187
11.3.13 <nsymbol> . . . 187
11.3.14 <nys> . . . 187
11.3.15 <pimd_command> . . . 188
11.3.16 <pressure> . . . 188
11.3.17 <time_bath> . . . 188
11.4 Method-, potential-, and interface-specific keywords in input.dat . . . 188
11.4.1 <abinit_mp_exe_command> . . . 188
11.4.2 <abinit_mp_input> . . . 188
11.4.3 <abinit_mp_output> . . . 188
11.4.4 <abinit_mp_reuse_mo> . . . 189
11.4.5 AccelNet model and evaluation keywords . . . 189
11.4.6 <mbx_json_file> . . . 189
11.4.7 <mbx_input_file> . . . 189
11.4.8 <iprint_mbx_mol_dipoles> . . . 189
11.4.9 <alchem_dat_dir> . . . 189
11.4.10 <alchem_potential> . . . 190
11.4.11 <alchem_scr_dir> . . . 190
11.4.12 <auto_string> . . . 190
11.4.13 <box_anis> . . . 190
11.4.14 <cells_phonon> . . . 190
11.4.15 <cluster> . . . 190
11.4.16 <cp2k_lib_calcforce> . . . 190
11.4.17 <cp2k_lib_chargeout> . . . 190
11.4.18 <cp2k_lib_output> . . . 191
11.4.19 <cut_rec_3d> . . . 191
11.4.20 <delay_aenet> . . . 191
11.4.21 <dftb_exe_command> . . . 191
11.4.22 <dftb_lib_output> . . . 191
11.4.23 <dftb_version> . . . 191
11.4.24 <dir_save_aenet> . . . 192
11.4.25 <dir_save_n2p2> . . . 192
11.4.26 <dir_save_mace> . . . 192
11.4.27 <dir_train_aenet> . . . 192
11.4.28 <dir_train_mace> . . . 192
11.4.29 <dosrange_phonon> . . . 192
11.4.30 <dt_conv_gad> . . . 192

11.4.31 <dt_om> . . . 192
11.4.32 <dt_poly> . . . 193
11.4.33 <dtemp_meta> . . . 193
11.4.34 <dual_dat_dir> . . . 193
11.4.35 <dual_potential> . . . 193
11.4.36 <dual_scr_dir> . . . 193
11.4.37 <efei> . . . 193
11.4.38 <ends_poly> . . . 193
11.4.39 <ends_string> . . . 193
11.4.40 <eps_best> . . . 194
11.4.41 <equation_om> . . . 194
11.4.42 <fc_best> . . . 194
11.4.43 <fdiff> . . . 194
11.4.44 <g03_command> . . . 194
11.4.45 <g03_dip> . . . 194
11.4.46 <g03_grad> . . . 194
11.4.47 <g09_command> . . . 195
11.4.48 <g09_dip> . . . 195
11.4.49 <g09_grad> . . . 195
11.4.50 <g09_oniom> . . . 195
11.4.51 <g16_command> . . . 195
11.4.52 <g16_dip> . . . 195
11.4.53 <g16_grad> . . . 195
11.4.54 <g16_oniom> . . . 195
11.4.55 <g98_command> . . . 195
11.4.56 <g98_dip> . . . 195
11.4.57 <g98_grad> . . . 195
11.4.58 <gamess_command> . . . 196
11.4.59 <gamma_gad> . . . 196
11.4.60 <gamma_om> . . . 196
11.4.61 <generate_aenet> . . . 196
11.4.62 <gh_meta> . . . 196
11.4.63 <guess_poly> . . . 196
11.4.64 <gw_meta> . . . 196
11.4.65 <iatom_qtst> . . . 196
11.4.66 <iformat_trj> . . . 197
11.4.67 <iformat_xyz> . . . 197
11.4.68 <igamma> . . . 197
11.4.69 <ikind_best> . . . 197
11.4.70 <ikind_xyz> . . . 197

11.4.71 <integrator_bcmd> . . . 197
11.4.72 <integrator_trpmd> . . . 197
11.4.73 <iobest> . . . 197
11.4.74 <ioption_best> . . . 198
11.4.75 <ioption_meta> . . . 198
11.4.76 <ioption_xsf_aenet> . . . 198
11.4.77 <ioption_xsf_n2p2> . . . 198
11.4.78 <ioption_xsf_mace> . . . 198
11.4.79 <iorder_hmc> . . . 198
11.4.80 <iprint_akin> . . . 199
11.4.81 <iprint_alc> . . . 199
11.4.82 <iprint_best> . . . 199
11.4.83 <iprint_bond> . . . 199
11.4.84 <iprint_box> . . . 199
11.4.85 <iprint_cavg> . . . 199
11.4.86 <iprint_cons> . . . 199
11.4.87 <iprint_cv_meta> . . . 199
11.4.88 <iprint_cv_tass> . . . 200
11.4.89 <iprint_dip> . . . 200
11.4.90 <iprint_dual> . . . 200
11.4.91 <iprint_eavg> . . . 200
11.4.92 <iprint_hfx_aenet> . . . 200
11.4.93 <iprint_hfx_n2p2> . . . 200
11.4.94 <iprint_mech> . . . 200
11.4.95 <iprint_meta> . . . 200
11.4.96 <iprint_mfe> . . . 200
11.4.97 <iprint_minfo> . . . 201
11.4.98 <iprint_mom> . . . 201
11.4.99 <iprint_nac> . . . 201
11.4.100 <iprint_oniom> . . . 201
11.4.101 <iprint_poly> . . . 201
11.4.102 <iprint_qmmm> . . . 201
11.4.103 <iprint_rdf> . . . 201
11.4.104 <iprint_rdfbead> . . . 201
11.4.105 <iprint_rdfcent> . . . 202
11.4.106 <iprint_rec_meta> . . . 202
11.4.107 <iprint_rgy> . . . 202
11.4.108 <iprint_str> . . . 202
11.4.109 <iprint_tfs> . . . 202
11.4.110 <iprint_xsf_aenet> . . . 202

11.4.111 <iprint_xsf_n2p2> . . . 202
11.4.112 <iprint_xsf_mace> . . . 202
11.4.113 <irandom> . . . 202
11.4.114 <irem_type> . . . 203
11.4.115 <istate_init> . . . 203
11.4.116 <istep_adjust_hmc> . . . 203
11.4.117 <istep_ax_hmc> . . . 203
11.4.118 <istep_hmc> . . . 203
11.4.119 <istep_max_hmc> . . . 203
11.4.120 <istep_mul_hmc> . . . 203
11.4.121 <iprint_rec_tass> . . . 203
11.4.122 <istep_save_aenet> . . . 204
11.4.123 <istep_save_n2p2> . . . 204
11.4.124 <istep_save_mace> . . . 204
11.4.125 <iprint_tass> . . . 204
11.4.126 <istep_train_aenet> . . . 204
11.4.127 <istep_train_n2p2> . . . 204
11.4.128 <istep_train_mace> . . . 204
11.4.129 <ivar_qmmm> . . . 204
11.4.130 <joption_meta> . . . 205
11.4.131 <kdisp_phonon> . . . 205
11.4.132 <kdos_phonon> . . . 205
11.4.133 <kickstep_bcmd> . . . 205
11.4.134 <lstep_train_aenet> . . . 205
11.4.135 <lstep_train_n2p2> . . . 205
11.4.136 <lstep_train_mace> . . . 205
11.4.137 <mace_use_cuda> . . . 205
11.4.138 <mace_device_id> . . . 206
11.4.139 <mace_rmax> . . . 206
11.4.140 <mace_generate_command> . . . 206
11.4.141 <ase_infer_options_file> . . . 206
11.4.142 <mace_infer_options_file> . . . 206
11.4.143 <mace_train_options_file> . . . 206
11.4.144 <mech_type> . . . 206
11.4.145 <zone> . . . 207
11.4.146 <mg_meta> . . . 208
11.4.147 <minxsf_train_aenet> . . . 208
11.4.148 <minxsf_train_n2p2> . . . 208
11.4.149 <minxsf_train_mace> . . . 208
11.4.150 <model_water> . . . 208

11.4.151 <molpro_command> . . . 209
11.4.152 <mopac_command> . . . 209
11.4.153 <n2p2_en_unit> . . . 209
11.4.154 <n2p2_len_unit> . . . 209
11.4.155 <ncons> . . . 209
11.4.156 <ncycle_poly> . . . 210
11.4.157 <neighbor_list_skin> . . . 210
11.4.158 <ngrid_poly> . . . 210
11.4.159 <ngrid_string> . . . 210
11.4.160 <nmeta> . . . 210
11.4.161 <nmulti> . . . 211
11.4.162 <np_poly> . . . 211
11.4.163 <nph_type> . . . 211
11.4.164 <npoly> . . . 211
11.4.165 <npt_type> . . . 212
11.4.166 <nref_meta> . . . 212
11.4.167 <nref_tass> . . . 212
11.4.168 <nstep_modes> . . . 212
11.4.169 <nstate> . . . 212
11.4.170 <ntass_rec> . . . 212
11.4.171 <nth_type> . . . 213
11.4.172 <ntt_type> . . . 213
11.4.173 <ntype_aenet> . . . 213
11.4.174 <oniom_dat_dir> . . . 213
11.4.175 <oniom_hi_potential> . . . 213
11.4.176 <oniom_lo_potential> . . . 213
11.4.177 <oniom_scr_dir> . . . 213
11.4.178 <orca_command> . . . 214
11.4.179 <params_cons> . . . 214
11.4.180 <params_dual> . . . 214
11.4.181 <minweight_train_dual> . . . 214
11.4.182 <params_lbfgs> . . . 214
11.4.183 <params_rdf> . . . 215
11.4.184 <params_rec_meta> . . . 215
11.4.185 <phase0_proc> . . . 215
11.4.186 <projcmf_poly> . . . 215
11.4.187 <qe_input_file_name> . . . 215
11.4.188 <qe_output> . . . 215
11.4.189 <qmmm_embedding> . . . 216
11.4.190 <qmmm_dat_dir> . . . 216

11.4.191 <qmmm_potential> . . . 216
11.4.192 <qmmm_scr_dir> . . . 216
11.4.193 <ratio_max_hmc> . . . 216
11.4.194 <ratio_min_hmc> . . . 216
11.4.195 <scale_bcmd> . . . 216
11.4.196 <scale_trpmd> . . . 216
11.4.197 <scan_exe_shell> . . . 217
11.4.198 <skin_aenet> . . . 217
11.4.199 <smash_dip> . . . 217
11.4.200 <smash_grad> . . . 217
11.4.201 <smash_guess> . . . 217
11.4.202 <smash_options> . . . 217
11.4.203 <temp_tamd_tass> . . . 217
11.4.204 <temprange_phonon> . . . 218
11.4.205 <temprange_rem> . . . 218
11.4.206 <tension> . . . 218
11.4.207 <testforce_tolerance> . . . 218
11.4.208 <testvirial_tolerance> . . . 218
11.4.209 <time_baro> . . . 218
11.4.210 <time_cv_bath> . . . 218
11.4.211 <time_cv_meta> . . . 218
11.4.212 <time_fc_meta> . . . 219
11.4.213 <time_limit_meta> . . . 219
11.4.214 <time_mode> . . . 219
11.4.215 <train_aenet> . . . 219
11.4.216 <turbo_command> . . . 219
11.4.217 <turbo_guess> . . . 219
11.4.218 <user_command> . . . 220
11.4.219 <user_input_file> . . . 220
11.4.220 <user_output_file> . . . 220
11.4.221 <vasp_command> . . . 221
11.4.222 <vasp_energy> . . . 221
11.4.223 <vasp_keyword_energy> . . . 221
11.4.224 <vasp_keyword_gradient> . . . 221
11.4.225 <vasp_keyword_stress> . . . 221
11.4.226 <vasp_output> . . . 222
11.4.227 <vasp_reuse_wavefunction> . . . 222
11.4.228 <xmpi> . . . 222
11.4.229 <xtb_exe_command> . . . 222
11.5 Conditional and auxiliary keywords in input.dat . . . 223

11.5.1 <atom_change> . . . 223
11.5.2 <charge_libnnp> . . . 223
11.5.3 <components> . . . 223
11.5.4 <file_dms_libnnp> . . . 223
11.5.5 <file_ip_libnnp> . . . 224
11.5.6 <file_pes_libnnp> . . . 224
11.5.7 <ioption_dms_libnnp> . . . 224
11.5.8 <molecules> . . . 224
11.5.9 <natom_ip_libnnp> . . . 224
11.6 Keywords in calc.dat . . . 224
11.6.1 <ncalc> . . . 224
11.6.2 <iprint_calc> . . . 226
11.6.3 <iprint_std_calc> . . . 226
11.6.4 <iconf_fin_calc> . . . 226
11.6.5 <iconf_ini_calc> . . . 226
11.6.6 <jformat_xyz> . . . 226
11.6.7 <jorigin_xyz> . . . 226
11.6.8 <jprint_xyz> . . . 227
11.6.9 <jspec_xyz> . . . 227
11.6.10 <jxyz_atom> . . . 227
11.6.11 <jxyz_bead> . . . 227
11.6.12 <jxyz_origin> . . . 227
11.6.13 <params_lin_dens> . . . 227
11.6.14 <params_adiff_dens> . . . 227
11.6.15 <params_angl_dens> . . . 227
11.6.16 <params_diff_dens> . . . 228
11.6.17 <params_dih_dens> . . . 228
11.6.18 <3d_range_cube> . . . 228
11.7 EAM, ADP, and tabulated-pair LAMMPS input . . . 228
11.8 LAMMPS-compatible Tersoff input . . . 228
11.9 Common keywords shared with constrained MD and metadynamics . . . 229
11.9.1 <ncons> . . . 229
11.9.2 <nref> . . . 229
11.9.3 <params_cons> . . . 229
11.10 Common keywords shared in all types of AFED . . . 229
11.10.1 <afed_type> . . . 229
11.10.2 <fenergy_max_afed> . . . 230
11.10.3 <mdcycle_pro_afed> . . . 230
11.10.4 <niter_afed> . . . 230
11.10.5 <nstep_pre_afed> . . . 230

11.10.6 <nstep_pro_afed> . . . 230
11.10.7 <params_afed> . . . 230
11.11 Additional keywords in DESCENT, ASCENT and AUTO . . . 231
11.11.1 <algo_ascent_afed> . . . 231
11.11.2 <ascent_sampling_afed> . . . 231
11.11.3 <dt_ascent_afed> . . . 231
11.11.4 <dt_conv_afed> . . . 231
11.11.5 <dt_damp_afed> . . . 231
11.11.6 <dt_descent_afed> . . . 231
11.11.7 <fdiff_sampling_afed> . . . 231
11.11.8 <gamma_ascent_afed> . . . 232
11.11.9 <niter_gad2evf_afed> . . . 232
11.11.10 <niter_refresh_afed> . . . 232
11.11.11 <nmiss_auto_afed> . . . 232
11.11.12 <ioption_eigen_afed> . . . 232
11.11.13 <nshot_auto_afed> . . . 232
11.11.14 <radius_auto_afed> . . . 232
11.11.15 <iprint_xyz_afed> . . . 232
11.11.16 <iprint_test_afed> . . . 233
11.12 Additional keywords in TAMD and LogMFD . . . 233
11.12.1 <alpha_logmfd> . . . 233
11.12.2 <dt_logmfd> . . . 233
11.12.3 <dt_tamd> . . . 233
11.12.4 <fenergy_ini_logmfd> . . . 233
11.12.5 <fenergy_ini_tamd> . . . 233
11.12.6 <gamma_logmfd> . . . 233
11.12.7 <ioption_afed> . . . 234
11.12.8 <logmfd_type> . . . 234
11.12.9 <mass_afed> . . . 234
11.12.10 <niter_bath_afed> . . . 234
11.12.11 <tamd_type> . . . 234
11.12.12 <temperature_tamd> . . . 234
11.12.13 <start_afed> . . . 235
11.12.14 <start_therm_afed> . . . 235
11.13 Source-audited keywords added in version 2.8.0 . . . 235
11.13.1 <afir> . . . 235
11.13.2 <ax> . . . 235
11.13.3 <nsubstep_ax> . . . 235
11.13.4 <bond> . . . 235
11.13.5 <group> . . . 235

11.13.6 <esf> . . . 236
11.13.7 <dvr> . . . 236
11.13.8 <quartic> . . . 236
11.13.9 <mbead_matsubara> . . . 236
11.13.10 <iprint_matsubara> . . . 236
11.13.11 <ntass> . . . 236
11.13.12 <ntrial_tps> . . . 236
11.13.13 <temp_kick_tps> . . . 236
11.13.14 <qe7_command> . . . 236
11.13.15 <cutoff_helium> . . . 236
11.13.16 <ncart> . . . 237
11.13.17 <energy> . . . 237
11.13.18 <lengthunit> . . . 237
11.13.19 <coordinates> . . . 237
11.13.20 <hessian> . . . 237
11.13.21 <imode> . . . 237
11.13.22 <dipoles> . . . 237
11.13.23 <ljrule> . . . 237
11.13.24 <ljcutoff> . . . 237
11.13.25 <ljpair> . . . 238
11.13.26 <ljbond> . . . 238
11.13.27 <skin_ljpair> . . . 238
11.13.28 <skin_ewald> . . . 238
11.13.29 <pme_mesh_ewald> . . . 238
11.13.30 EAM/ADP/PAIR neighbor skin . . . 238
11.13.31 Tersoff neighbor skin . . . 238
11.13.32 <oh_cutoff_oss> . . . 238
11.13.33 <oo_cutoff_oss> . . . 238
11.13.34 <3b_cutoff_oss> . . . 238
11.13.35 <ioption_oss> . . . 239
11.13.36 <ewald_oss> . . . 239
11.13.37 <ewpol_oss> . . . 239
11.13.38 <nacme> . . . 239

12 Files **239**

12.1 Input files . . . 239
12.1.1 input.dat . . . 239
12.1.2 input_default.dat . . . 239
12.1.3 centroid.dat . . . 240
12.1.4 LAMMPS many-body/table files . . . 240
12.1.5 gamess.dat . . . 240

12.1.6 g03.dat . . . 240
12.1.7 g98.dat . . . 240
12.1.8 CFF files . . . 240
12.1.9 molpro.dat . . . 240
12.1.10 mopac.dat . . . 241
12.1.11 orca.dat . . . 241
12.1.12 structure.dat . . . 241
12.1.13 turbo.dat . . . 241
12.1.14 TAMD . . . 242
12.1.15 LogMFD . . . 242
12.2 Output files . . . 242
12.2.1 afed.out . . . 242
12.2.2 afed.xyz . . . 244
12.2.3 afed_weight.out . . . 244
12.2.4 alc.out . . . 244
12.2.5 best.out . . . 244
12.2.6 bond.out . . . 244
12.2.7 box.dcd . . . 245
12.2.8 box.out . . . 245
12.2.9 calc.xyz . . . 246
12.2.10 charges.dcd . . . 246
12.2.11 cons.out . . . 246
12.2.12 cv.out . . . 247
12.2.13 dip.dcd . . . 247
12.2.14 dipole.out . . . 247
12.2.15 dipole_scan.out . . . 248
12.2.16 dual.out . . . 248
12.2.17 eavg.out . . . 248
12.2.18 final.xyz . . . 249
12.2.19 forces.out . . . 249
12.2.20 forces_scan.out . . . 249
12.2.21 force.dcd . . . 249
12.2.22 force_high.dcd . . . 250
12.2.23 force_low.dcd . . . 250
12.2.24 hessian.out . . . 250
12.2.25 logmfd.out . . . 250
12.2.26 mech.out . . . 250
12.2.27 meta.out . . . 251
12.2.28 mfe.out . . . 251
12.2.29 nac.out . . . 251

12.2.30 oniom.out . . . 251
12.2.31 phonon_dos.out . . . 252
12.2.32 phonon_energy.out . . . 252
12.2.33 phonon_kdisp.out . . . 252
12.2.34 phonon_kdos.out . . . 252
12.2.35 pot.dcd . . . 252
12.2.36 potential_scan.out . . . 253
12.2.37 qmmm.out . . . 253
12.2.38 rdf.out . . . 253
12.2.39 rec.out . . . 253
12.2.40 rgy.out . . . 254
12.2.41 standard.out . . . 254
12.2.42 string.out . . . 255
12.2.43 string.xyz . . . 255
12.2.44 string_final.out . . . 255
12.2.45 template.xyz . . . 256
12.2.46 tfs.out . . . 256
12.2.47 trj.dcd . . . 256
12.2.48 trj.out . . . 257
12.2.49 trj.xsf . . . 257
12.2.50 trj.xyz . . . 257
12.2.51 trjext.xyz . . . 258
12.2.52 vel.dcd . . . 258
12.3 Restart files . . . 258
12.3.1 afed.ini . . . 258
12.3.2 auto.ini . . . 259
12.3.3 averages.ini . . . 260
12.3.4 bath.ini . . . 260
12.3.5 box.ini . . . 260
12.3.6 cstate.ini . . . 260
12.3.7 cv.ini . . . 261
12.3.8 gad.ini . . . 261
12.3.9 geometry.ini . . . 261
12.3.10 hills.ini . . . 261
12.3.11 step.ini . . . 261
12.3.12 string.ini . . . 262

VI Theoretical Background 262

13 Theoretical background **262**

13.1 Mechanochemistry . . . 263
13.2 Steepest descent . . . 264
13.3 Box optimization . . . 264
13.3.1 Zero pressure condition . . . 264
13.3.2 Finite pressure condition . . . 264
13.3.3 Finite tension condition . . . 265
13.4 Static elastic constants . . . 265
13.5 Normal mode analysis . . . 265
13.6 Phonon calculations . . . 266
13.7 String method . . . 267
13.8 OM action . . . 267
13.9 NVE molecular dynamics . . . 268
13.9.1 Velocity-Verlet algorithm . . . 268
13.10 NVT molecular dynamics . . . 269
13.10.1 Nosé–Hoover method . . . 269
13.10.2 Nosé–Hoover chain method . . . 270
13.10.3 Massive Nosé–Hoover chain method . . . 272
13.11 Path integral methods . . . 273
13.11.1 Fictitious mass . . . 274
13.11.2 Thermostats . . . 275
13.11.3 Barostats . . . 277
13.11.4 How to set up RPMD and CMD . . . 278
13.11.5 Dual-level hybrid Monte Carlo . . . 278
13.11.6 Fourth order path integral hybrid Monte Carlo . . . 278
13.11.7 Translational and rotational corrections . . . 278
13.12 Replica exchange hybrid Monte Carlo . . . 280
13.12.1 Temperature replica exchange . . . 280
13.12.2 Hamiltonian replica exchange . . . 280
13.13 Metadynamics . . . 281
13.14 Conventional and well-tempered metadynamics . . . 282
13.14.1 Coordination number . . . 283
13.15 Well-sliced metadynamics . . . 284
13.16 Mean force dynamics (AFED) . . . 285
13.16.1 Steepest descent . . . 286
13.16.2 Gentlest ascent dynamics . . . 287
13.16.3 Automated search . . . 290
13.16.4 Temperature accelerated molecular dynamics . . . 291
13.16.5 Logarithmic mean force dynamics . . . 292
13.17 Nonadiabatic dynamics . . . 294

13.18 Hybrid potentials: ONIOM and QM/MM 295
13.19 Multiple time scale method 296
13.20 BEST method 297
13.21 Ewald sum 297
13.22 Polarizable force field 298
13.23 Dihedral bond potential 300

VII Publications and References 301

14 Publications 301

Part I

Getting Started

1 Installation and compilation

Core workflows

For a new PIMD 2.8.0 installation, use the top-level CMake build. This section is the canonical description of the common build procedure. Backend chapters describe only the additional libraries and `-D...` options that are specific to that backend; they do not repeat the generic CMake build steps.

Unpack the release archive and enter the source tree:

```
> tar xzf PIMD-2.8.0.tar.gz
> cd PIMD-2.8.0
> ls -d CMakeLists.txt README.md cmake examples lib manual prep source tools
CMakeLists.txt  README.md  cmake  examples  lib  manual  prep  source  tools
```

The standard TOML/input-preparation front end distributed with this release is **pimd-prep** version 0.9.9. Its source-tree launcher is `prep/pimd-prep`. The former `tools/pimd-prep` launcher has been removed; use `prep/pimd-prep` instead.

1.1 Build requirements

The standard CMake configuration requires:

- CMake 3.20 or newer;
- C, C++, and Fortran compilers;
- an MPI development environment; and
- BLAS/LAPACK, unless the selected backend or math-library option provides the required linear-algebra libraries.

The current top-level CMake configuration always performs MPI discovery, even when only **pimd.x** will be built or used. Therefore an MPI development installation must be visible at CMake configuration time. Optional interfaces can add further requirements; those dependencies are documented only in the corresponding backend sections.

1.2 Standard out-of-source build

From the top-level PIMD directory, configure and build with

```
> cmake -S . -B build
> cmake --build build -j
```

An out-of-source build is strongly recommended. Do not copy the source tree into a separate compile directory for the normal CMake workflow. A separate build directory can be used for each compiler or optional backend, for example

```
> cmake -S . -B build-gnu
> cmake -S . -B build-mace -DMACE=ON
> cmake -S . -B build-openmx -DOPENMX=ON -DOPENMP=ON ...
```

This keeps independent CMake caches from interfering with one another.

The global `OPENMP` option defaults to OFF. When ON, it supplies OpenMP compiler flags and runtime libraries to all PIMD targets. Use `-DOPENMP=OFF` to add no global OpenMP requirement, provided the selected backends support it. Linked dependencies can still supply their own OpenMP flags and libraries. OpenMX requires ON; CP2K obtains its requirements from its package metadata. External libraries must be built separately with compatible compiler and OpenMP runtimes; this option does not rebuild them.

1.3 Choosing compilers and MPI

Select the compiler toolchain when the build directory is first configured. For example, a GNU build can be configured as

```
> CC=gcc CXX=g++ FC=gfortran cmake -S . -B build-gnu
```

and an Intel oneAPI build may use

```
> CC=icx CXX=icpx FC=ifx cmake -S . -B build-intel
```

The MPI implementation and all embedded libraries linked into one executable must be compatible with the selected compiler family. If MPI detection is ambiguous, inspect the MPI wrapper and the CMake cache before compiling, for example

```
> mpifort --show
> cmake -LAH -N build | grep -E 'CMAKE_.*COMPILER|MPI_.*COMPILER'
```

The exact MPI-wrapper diagnostic option can differ between MPI implementations.

Changing the compiler family in an already configured build directory is not recommended. Create a fresh build directory instead.

1.4 Math libraries

By default, CMake searches for BLAS/LAPACK and links the imported `LAPACK::LAPACK` target. Two alternative selections are available:

- On macOS, use `-DACCUSE=ON` to select the Accelerate framework.
- Use `-DMKLUSE=ON` when Intel MKL is available through the local CMake/toolchain installation.

Do not enable multiple math-library selections intentionally. Some embedded backends, notably CP2K or OpenMX with a complete external link line, can supply their own math-library stack; the backend-specific section explains the extra link requirements.

1.5 Executable targets

The top-level CMake build provides the following existing targets:

pimd.x	Serial PIMD executable.
pimd.mpi.x	MPI PIMD executable.
polymers.x	Polymer/QTST-related MPI executable.
convert_tinker.x	TINKER-to-PIMD conversion utility.
prep_liquid.x	Initial-structure utility for molecular liquids.

To build only one target, use the target name explicitly:

```
> cmake --build build --target pimd.x -j
> cmake --build build --target pimd.mpi.x -j
```

The available targets can be listed with

```
> cmake --build build --target help
```

Two optional targets are available after the same configuration:

```
> cmake --build build --target manual
> cmake --build build --target tools -j
```

With the Unix Makefiles generator, these are also `make manual` and `make tools` inside the build directory. Neither is part of the default build. The `manual` target requires a working PDF-producing LaTeX engine and the manual's packages (including `newtxtext`, `newtxmath`, `tcolorbox`, and `titlesec`) and produces `build/manual/manual.pdf` using three passes. All generated files stay in the build directory. The build selects the first runnable `pdflatex`, `lualatex`, or `xelatex` from `PATH`, in that order. To select an engine explicitly, set `-DPIMD_LATEX_EXECUTABLE=/path/to/engine` at configuration time. Fonts come from the TeX distribution; no operating-system fonts or `fontspec` are required. LaTeX is not needed to configure PIMD or build the executables.

The `tools` target builds the standalone Fortran utilities under **tools** into **build/tools**, with a `.x` suffix. Individual utilities can be selected, for example `cmake --build build --target xsf2npf.x`. MPI utilities use the configured MPI Fortran library, and tools requiring linear algebra use PIMD's selected math libraries.

The Fortran utilities and PIMD share the mathematical and physical constants in `source/physical_constants.F90`. Tools use the established engine values, replacing their independent and sometimes differing definitions. For manual compilation outside CMake, compile that module before the tool with the same Fortran compiler.

1.6 Optional interfaces

Optional interfaces are selected when the build directory is configured. Common options are summarized here only as an index; the corresponding backend section is the sole location for backend-specific prerequisites, patches, and library paths.

Interface	CMake option	Details
AccelNet	`-DACCELNET=ON`	Section 6.4.2
AENET	`-DAENET=ON`	Section 6.4.1
ASE	`-DASE=ON`	Section 6.4.6
MACE	`-DMACE=ON`	Section 6.4.5
N2P2	`-DN2P2=ON`	Section 6.4.4
Quantum ESPRESSO	`-DQE=ON`	Section 6.3.13
DFTB+	`-DDFTB+=ON`	Section 6.3.4
CP2K	`-DCP2K=ON`	Section 6.3.3
OpenMX	`-DOPENMX=ON`	Section 6.3.14
MBX	`-DMBX=ON`	Section 6.3.15
VASP6	`-DVASP6=ON`	Section 6.3.17
PME	`-DPME=ON`	PME/Ewald-related sections

Backend options can be combined when their compiler, MPI, and dependent libraries are mutually compatible. Prefer a descriptive build-directory name for each combination rather than repeatedly reconfiguring one directory.

1.7 Running the test suite

We recommend running the CTest suite after compilation completes, before starting production calculations, this requires setting `BUILD_TESTING=ON` when configuring.

```
> cmake -S . -B build -DBUILD_TESTING=ON
> cmake --build build -j
> ctest --test-dir build --output-on-failure
```

With the Unix Makefiles generator, run `ctest --output-on-failure` inside the build directory after `make` has completed.

The registered tests depend on the configured features. They include input and output checks, internal force/virial finite differences, and checks for enabled external methods when their required data are available. External force/virial checks use PIMD's `<method> = TESTFORCE` and `<method> = TESTVIRIAL` through the linked backend. These establish derivative consistency, comparisons against an independent implementation are separate test groups. Python 3 and a working MPI launcher are required. Enable `PME=ON` to include the MPI periodic-water checks.

Inspect the registered tests and select a group when diagnosing failures:

```
> ctest --test-dir build -N
> ctest --test-dir build -L external --output-on-failure
```

The `internal`, `lammps`, and backend labels select narrower groups. A group with no registered tests has not been validated. Check CMake's messages for omitted cases and missing optional data.

MPI ranks and OpenMP threads.. The shared derivative suites accept two CMake settings for their execution layout:

- `VALIDATION_MPI_RANKS` sets the MPI ranks per test. Its default, `0`, uses one rank per bead. Any positive value is used as given, including fewer ranks than beads. The runner distributes beads among rank groups, allowing each group to evaluate several beads and using multiple workers per group when possible.
- `VALIDATION_OMP_THREADS` sets `OMP_NUM_THREADS` for each test process and defaults to one. Enable `OPENMP=ON` when PIMD itself should use OpenMP; linked libraries also need compatible OpenMP support.

For example, configure six MPI ranks and two threads per process with

```
> cmake -S . -B build -DBUILD_TESTING=ON -DOPENMP=ON \
    -DVALIDATION_MPI_RANKS=6 -DVALIDATION_OMP_THREADS=2
> cmake --build build -j
> ctest --test-dir build --output-on-failure --parallel 12
```

CTest reserves ranks times threads for each shared derivative test. Its `--parallel` option controls concurrent test scheduling; it does not set MPI ranks within a test.

MPI launching uses CMake's `MPIEXEC_EXECUTABLE` and `MPIEXEC_NUMPROC_FLAG`. Additional launcher options can be supplied through `MPIEXEC_PREFLAGS` and `MPIEXEC_POSTFLAGS` at configuration time.

External test prerequisites and any additional configuration options are documented beside each suite under `tests/external`. Enable the method and provide its required runtime data when configuring the build. Some suites download verification models during configuration; their guides explain cache locations, offline alternatives, and how to disable downloads. Set `LAMMPS_EXECUTABLE` to add independent LAMMPS comparisons.

Tests retain their inputs, logs, and numerical diagnostics under the build directory. Inspect those files on failure; successful compilation alone does not establish numerical agreement. The suite layout, labels, and diagnostic files are described in `tests/README.md`.

Discovery and link probes can also be run before the main compilation with `cmake --build build --target check-prebuil`. These generally run only for external backends enabled in the configured build, where a probe is provided; enabling OpenMP also registers its compiler/runtime checks. A configuration without applicable probes may have no prebuild tests. These checks validate build setup, not numerical backend execution.

1.8 Checking and diagnosing a build

After configuration, inspect the cache and target list before a long build:

```
> cmake -LAH -N build
> cmake --build build --target help
```

For detailed compiler and linker commands, use

```
> cmake --build build --verbose
```

Typical configuration failures have the following causes:

- **MPI not found**: the MPI development package or compiler wrappers are not visible to CMake.
- **BLAS/LAPACK not found**: install the development libraries, or select a supported Accelerate/MKL configuration.
- **Unexpected compiler after reconfiguration**: remove the old build directory and configure a fresh one with the intended toolchain.
- **Backend library/module not found**: check the backend-specific path option and the matching backend section of this manual.

CMake cache variables that contain filesystem locations, such as `CP2K_DIR`, `VASP6_DIR`, and `HDF5_INC`, are path-valued cache entries. They can therefore be inspected directly with `cmake -LAH -N` or a CMake GUI without being represented as Boolean switches.

1.9 Legacy makefile build

The makefile-based build in `source` is retained for compatibility and for several historical interfaces. CMake is the recommended build system for new installations. The legacy makefile also builds historical utilities such as `calc.x`, `gau.x`, `run_molpro.x`, and `prep_best.x`; these are not targets of the current top-level CMake build.

When a backend chapter still documents a makefile-specific procedure, that procedure is backend-specific; the common CMake workflow is not repeated there.

1.10 Next step: the TOML-first workflow

PIMD 2.8.0 uses `pimd.toml` as the authoritative user input for the curated examples. The **`pimd-prep`** utility validates this file and generates the legacy `input.dat` consumed by the Fortran engine. After building PIMD, continue with Section 2 before reading the individual method chapters.

A typical example is launched as

```
> cd examples/02_path_integral_dynamics/H2O/pimd_nvt_water
> ./run.sh
```

The launcher regenerates `input.dat` from `pimd.toml` before execution. If the executables are in a separate build directory, set `PIMD_BIN_DIR` as described in Section 2.10.

2 TOML input and pimd-prep

Core workflows

In the PIMD 2.8.0 curated examples, pimd.toml is the authoritative user input. pimd-prep version 0.9.9 validates it and generates the legacy input.dat read by the Fortran engine. The generated file should normally be treated as a build artifact rather than as a second input file to maintain by hand.

This separation is a central change in PIMD 2.8.0. The Fortran input parser and its complete keyword set remain available for compatibility, while new examples can use structured section and key names, TOML syntax, explicit units for common physical quantities, and early validation of input mistakes. Omitted TOML settings are intentionally omitted from the generated input.dat; PIMD then obtains their values from the canonical input_default.dat. Consequently, a new input can contain only the settings that distinguish one calculation from the defaults.

2.1 Running pimd-prep 0.9.9

When installed on PATH, invoke the program directly:

```
pimd-prep --version
pimd-prep --help
```

From an unpacked source tree, use

```
./prep/pimd-prep --help
```

The distributed tar archive preserves the executable permission. If a ZIP extraction or file copy has lost that bit, no permission change is required; either of the following forms works:

```
sh prep/pimd-prep --help
python3 -m prep --help
```

The shell launcher checks for python3 first and prints a clear error when it is unavailable. Python 3.11 or later is recommended.

A new calculation normally starts with init. The main command workflow is

```
pimd-prep init PIMD VASP6
pimd-prep show
pimd-prep check
```

A successful init writes pimd.toml, copies the canonical PIMD 2.8.0 input_default.dat into the run directory when absent, and immediately generates input.dat. An existing local default file is preserved. After editing TOML by hand, regenerate the engine input with

```
pimd-prep build
pimd-prep build pimd.toml -o input.dat
```

The historical pimd-prep pimd.toml spelling remains a compatibility form of pimd-prep build pimd.toml.

The commands have distinct roles:

- init: interactively create, resume, or edit a setup;
- build: validate TOML-to-legacy conversion and write input.dat;
- check: perform strict run-readiness checks;

- `show`: display effective settings after applying defaults;
- `explain`: describe methods, potentials, TOML keys, legacy tags, and boundary support;
- `migrate`: convert legacy `input.dat` to TOML where the mapping is safe.

`build` deliberately does not require runtime backend files that may be supplied later; use `check` for the strict pre-run diagnostic.

2.2 A minimal example

The distributed static-water example contains only the required structure manifest and the choice of potential:

```
[files]
structure = "structure.dat"

[potential]
type = "WATER"
```

The default method is **<method>** = `STATIC`, so it does not need to be repeated. Convert the input explicitly with

```
> ../../../../prep/pimd-prep pimd.toml
```

or, from inside a distributed example, use `run.sh`, which performs the conversion automatically before launching PIMD.

The generated `input.dat` begins with a comment identifying **pimd-prep** and the source TOML file. It also fixes **<input_style>** = `NEW`. This is the normal structure format for new 2.8.0 workflows.

2.3 Structured settings

TOML settings are grouped by purpose rather than by the historical order of Fortran keywords. Representative sections are shown below; the distributed examples provide the most reliable starting points for specialized methods and interfaces.

[system]	Boundary conditions, cell data, and system topology.
[dynamics]	Method, ensemble, number of steps, timestep, temperature, and pressure.
[path_integral]	Number of beads and path-integral integration controls.
[thermostat], **[barostat]**	Thermostat and barostat settings.
[potential]	Potential or electronic-structure backend.
[parallel]	Bead/force parallelization and XMPI selection. For example, `xmpi = true` generates **<xmpi>**=ON.
[output]	Output intervals and output-format settings.
[files]	Top-level run/setup files and directories; backend-native dependencies are not duplicated here.
[legacy]	Escape hatch for supported engine-level keywords without a dedicated TOML mapping.

Additional dedicated sections are available for constraints, hybrid Monte Carlo, dual potentials, metadynamics, TASS, replica exchange, AFED, string and polymer calculations, machine-learning interfaces, nonadiabatic dynamics, QM/MM, ONIOM, phonons, mechanochemistry, and other specialized features. Unknown section names or keys are rejected; where possible, **pimd-prep** suggests a close valid spelling.

2.3.1 Ensemble-specific thermostat and barostat settings

The TOML front end exposes only controls that are physically active for the selected ensemble. In particular, NVE and NVT are fixed-cell ensembles and must not contain a `[barostat]` table. NVT uses thermostat controls for ordinary PIMD, but no pressure, cell-mode, tension, or barostat-time setting is required or accepted. For example:

```
[dynamics]
method      = "PIMD"
ensemble    = "NVT"
temperature = "300 K"

[thermostat]
type = "MNHC"
```

For NPT and NPH, the external scalar pressure belongs in `[dynamics].pressure`. Cell fluctuations, when explicitly changed from the engine default, belong in `[barostat]`:

```
[dynamics]
method   = "PIMD"
ensemble = "NPT"
pressure = "1 bar"

[barostat]
cell_mode = "CUBIC2"
time      = "1000 fs"
```

The single TOML key `cell_mode` is converted to the ensemble-specific legacy keyword **<npt_type>**, **<nph_type>**, **<ntt_type>**, or **<nth_type>**; users therefore do not need to select a legacy tag name.

For NTT and NTH, use `[barostat].tension_tensor` instead of scalar pressure. The current PIMD/PIHMC tension-ensemble branches require `cell_mode = "PPHEX"`. **pimd-prep** rejects pressure in NTT/NTH and rejects all barostat keys in NVE/NVT rather than silently ignoring them. When `init -edit` changes a variable-cell ensemble to NVT/NVE, stale pressure and barostat settings are removed automatically.

2.4 Readable values and units

For common physical quantities, **pimd-prep** accepts values with units. For example:

```
[dynamics]
method      = "PIMD"
ensemble    = "NVT"
temperature = "300 K"
timestep    = "0.25 fs"
```

Time values can be written in fs, ps, or as; temperature is expressed in K. Pressure values, when used by NPT/NPH, accept MPa, GPa, kPa, Pa, bar, or atm. Unitless numeric values remain accepted in the native PIMD units expected for the corresponding keyword. Method, ensemble, thermostat type, potential names, and textual XMPI values are normalized to the uppercase forms used by the engine. A typical XMPI block is

```
[parallel]
beads = 2
force = 2
xmpi  = true
```

Here `beads` and `force` map to **<np_beads>** and **<np_force>**; `xmpi = true` maps to **<xmpi>**=ON. **pimd-prep**

rejects unsupported XMPI method/ensemble combinations, force backends, and inference/training combinations instead of silently generating an ineffective or partially ignored setting.

Boundary conditions can likewise be written using descriptive values such as `free`, `periodic`, or `periodic_unwrapped`. The structure-aware modes `EXTXYZ`, `ANGSTROM`, and `BOHR` are also accepted.

2.4.1 Metadynamics TOML

Metadynamics uses `[metadynamics].walkers` rather than `[path_integral].beads`; both map to the historical **<nbead>**, but the walker terminology reflects the actual MTD interpretation. Collective variables are written as structured `cvN_*` keys rather than as a literal **<nmeta>** block. For example:

```
[metadynamics]
walkers = 1
collective_variables = 1
cv1_type = "DIFF"
cv1_bond1 = [1, 9]
cv1_bond2 = [9, 2]
hill_height = "100 K"
hill_widths_DIFF = "0.10 angstrom"
reconstruction_grid_DIFF = ["-1.5 angstrom", "1.5 angstrom", "0.02 angstrom"]
```

For multiple walkers, use `initial_walker_spread_temperature` for the initial spread. Hill deposition timing is written as `deposition_mode = "displacement_and_time"` or `"fixed_interval"`. The legacy **<ioption_meta>** walker-separation switch can be represented by the compatibility key `walker_hill_mode`, but the current Fortran source does not apply its final hill-suppression gate; new inputs should normally omit it. Length-like MTD values accept angstrom or bohr, angular values use degrees, and coordination-number widths/grids are dimensionless. Only active CV types are written in TOML; **pimd-prep** fills the inactive legacy rows internally. Raw `hill_widths` and `reconstruction_grid` blocks remain available for compatibility, but a raw block must contain all eight legacy CV types so that the Fortran reader cannot silently discard a partial override.

2.4.2 Built-in WATER models

With **<type>** = `WATER`, the TOML front end follows the Fortran source and selects q-TIP4P/F by default. No extra model key is needed:

```
[potential]
type = "WATER"  # q-TIP4P/F is the default.
```

The other built-in choices are selected only when needed:

```
water_model = "SPC/F"  # Flexible SPC/F2 in the source implementation.
# or
water_model = "RWK"
```

Numeric values 0, 1, and 2 remain accepted for legacy compatibility, but the named form is preferred in new TOML input.

2.4.3 Structured constraints and TASS collective variables

Constraint definitions use numbered tables instead of a literal **<ncons>** block. For example:

```
[constraints]
number = 1
output = false
force_constant_DIFF = 0.08
```

```
[constraints.1]
type = "DIFF"
bond1 = [1, 9]
bond2 = [9, 2]
target = "0.10 angstrom"
```

Setting `output = false` generates the historical disabled value without requiring the user to remember `-1`. A `force_constant_TYPE` key overrides only the named CV type in TOML, but **pimd-prep** expands it to all eight rows of **<params_cons>**. This is important because the Fortran reader consumes the complete eight-type table; a partial raw block can otherwise fall back to the defaults.

TASS uses the same readable CV definitions, but the sampling method is kept separate from the CV type:

```
[tass]
collective_variables = 2
cv_output = false

[tass.cv.1]
sampling = "CONSTRAINT"
type = "DIH"
atoms = [2, 7, 8, 9]
range = ["-180 degree", "168 degree"]
note = "phi; constrained X axis"

[tass.cv.2]
sampling = "METADYNAMICS"
type = "DIH"
atoms = [7, 8, 9, 17]
note = "psi; metadynamics Y axis"
```

The supported sampling names are `AFED`, `CONSTRAINT`, and `METADYNAMICS`. A constrained TASS CV requires a range; `AFED` does not take one. The optional `note` is a TOML commentary field and is not passed to the engine.

2.4.4 Named L-BFGS convergence thresholds

The legacy **<params_lbfgs>** record contains four ordered numbers. New TOML names each threshold explicitly:

```
[optimization]
max_displacement = "0.0001 bohr"
rms_displacement = "0.00001 bohr"
max_force        = "0.00001 hartree/bohr"
rms_force        = "0.000001 hartree/bohr"
```

Displacements accept bohr or angstrom; force thresholds accept hartree/bohr or eV/angstrom. **pimd-prep** writes the engine order (maximum displacement, RMS displacement, maximum force, RMS force).

2.4.5 Phonon-dispersion wave vectors

The components supplied to **<kdisp_phonon>** are dimensional wave-vector components used directly in the phase $\mathbf{g}\cdot\mathbf{d}$; they are not fractional reciprocal-lattice coordinates. The unit is therefore explicit in TOML:

```
[phonon]
dispersion_kpoints_unit = "1/bohr"
dispersion_kpoints = [[0.0, 0.0, 0.0], [0.5, 0.0, 0.0]]
```

Both 1/bohr and 1/angstrom are accepted and converted to the native engine unit.

2.4.6 SMASH options

For an embedded SMASH 3.0.2 calculation, TOML groups options by SMASH directive rather than asking the user to paste a raw <smash_options> block:

```
[smash.job]
method = "b3lyp"
basis  = "6-31g(d)"
memory = "1gb"

[smash.control]
print = "minimum"
```

Optional [smash.scf] and expert extra_directives are also supported. The <smash_options> block is generated internally; PIMD supplies the geometry. See Section 6.3.1.

2.4.7 Rigid molecular components

Molecular-component data are retained because ROTOR and liquid-preparation workflows need both a body-fixed reference geometry and the atom mapping for each molecule instance. The user-facing form is:

```
[molecular_components]
number_of_components = 1

[molecular_components.component.1]
name = "H2O"
shape = "NONLINEAR"
coordinate_unit = "angstrom"
reference_coordinates = [[0.0,0.0,0.0], [0.788,0.616,0.0], [-0.788,0.616,0.0]]
molecules = [[1,5,9], [2,6,10]]
```

The accepted shape names are NONLINEAR, LINEAR, and MONOATOMIC; the historical engine name GENERAL is generated internally for a nonlinear component.

2.5 The [files] manifest

The [files] table is a top-level run/setup manifest. Its entries do not generate legacy keyword blocks; **pimd-prep** uses them for early path checks. Information already owned by a backend-native input file is not repeated here. This avoids numbered keys such as pseudopotential_2 or n2p2_weights_2 and keeps a single source of truth for each dependency.

For example, an MTP calculation may contain

```
[files]
structure = "structure.dat"
mtp_model = "pot.mtp"
```

For Quantum ESPRESSO, the pseudopotential filenames are read from the ATOMIC_SPECIES card in qe.dat; they are therefore not copied into [files]. For N2P2, the conventional files input.nn, scaling.data, and weights.ZZZ.data are checked directly, with the required weight files inferred from the elements line in

input.nn. MACE and ASE model paths remain in their backend YAML files. Likewise, groups of related directories belong to their semantic sections, for example [oniom].data_directories, [qmmm].data_directory, and [alchemical].data_directories, rather than numbered manifest keys.

Keys beginning with external_ remain useful for top-level resources that must be supplied by the user or environment, such as a VASP POTCAR. A setup helper itself can be listed as setup_script; files produced by that helper need not be duplicated as downloaded_* entries when the backend input already names them.

2.6 Defaults and the generated input.dat

A TOML file does not need to restate the complete PIMD configuration. pimd-prep writes only the settings explicitly present in the TOML file, plus <input_style> = NEW. At run time, omitted engine keywords are supplied from input_default.dat. This design keeps short examples short and avoids maintaining the same default value in many places.

For this reason, do not normally edit the generated input.dat. Change pimd.toml and regenerate it instead. Directly maintained input.dat remains supported for existing calculations and for users who deliberately choose the legacy workflow; Section 12.1.1 describes the engine-level file format.

2.7 Using [legacy]

The structured TOML vocabulary covers common and curated workflows. A supported PIMD keyword that does not yet have a dedicated mapping can be provided in the [legacy] table. The key is the legacy keyword name without angle brackets:

```
[legacy]
some_engine_keyword = 10
```

When input_default.dat is available, pimd-prep checks legacy tag names against it, together with a small set of recognized method-specific tags. Duplicate settings that would generate the same legacy tag are rejected. Multi-line strings can be used when a legacy keyword requires a literal multi-line payload.

Important

Use [legacy] as a compatibility escape hatch, not as the default style for new examples. When a dedicated TOML key exists, prefer the structured section/key form so that validation, units, and intent remain clear.

2.8 Validation, boundary conditions, and migration

pimd-prep 0.9.7 checks TOML syntax, registered section and key names, duplicate legacy tags, required paths, method/backend compatibility, and backend-specific run-readiness conditions. The potential registry covers all engine potential names and records whether each backend supports periodic boundaries, free boundaries, or both. For newly initialized NEW-style structures, the canonical representations are

```
PERIODIC  -> EXTXYZ    (cell in Lattice="..." in structure.dat)
FREE      -> ANGSTROM  (no PIMD cell)
```

The method can impose an additional restriction. init offers only compatible choices and check rejects incompatible existing setups.

2.8.1 Required, conditional, optional, and generated runtime artifacts

Starting with pimd-prep 0.9.7, backend file diagnostics distinguish four cases instead of treating every familiar filename as mandatory:

- **required**: the selected execution path cannot start without the artifact;
- **conditionally required**: the artifact is required only when an enabled option or another input file references it;
- **optional**: the engine has a valid fallback or the file is only useful for interoperability/extra checking;
- **generated**: PIMD or `pimd-prep` creates the artifact, so its pre-run absence is never a missing-file error.

Only the first two categories, after their condition is active, are reported as `ACTION REQUIRED`. The end of `init` prints a non-fatal run-readiness summary and `check` remains the strict final validation. Adding a missing runtime file does not require rerunning `init`.

This distinction is important for source-dependent interfaces. For embedded VASP6, PIMD generates `POSCAR` from the current PIMD geometry and cell, so only `INCAR`, `KPOINTS`, and `POTCAR` are fixed pre-run VASP6 files. For EAM/ADP/PAIR/TERSOFF/METALWATER, `input.lammps` and the potential/table file selected by `pair_coeff` are required, but a `read_data` target is only a LAMMPS-side interoperability aid because the native PIMD force path takes coordinates, masses, and the cell from PIMD. CFF is different: its topology is actually read through `read_data`, so that referenced data file is conditionally required. MACE training YAML is required only when on-the-fly training is enabled. For TURBOMOLE, a pre-existing `coord` prototype is required because the interface first copies it to `coord.turbo`; PIMD then rewrites the geometry block from the current PIMD coordinates. The backend requirement policy is shared by `explain`, the `init` summary, and `check` so that optional files are not promoted to blocking diagnostics by separate hand-written message paths.

For CFF, version 0.9.6 and later also performs a strict LAMMPS-compatible force-field preflight before execution. It validates supported style names and coefficient forms, required coefficient coverage, topology counts and type/reference ranges in the data file selected by `read_data` (default `data.lammps`), and recursively included command files. It also checks cross-file consistency against `structure.dat`: atom count, FREE/PERIODIC boundary, periodic cell metric, optional species comments, and LAMMPS Masses coverage/positivity. The isotope labels H, D, T, and Mu are treated as one hydrogen chemical family for force-field consistency while their dynamical masses remain distinct in PIMD. Use

```
pimd-prep show pimd.toml --force-field
pimd-prep explain cff-styles
```

to inspect the effective parsed force field and the supported CFF style/coeff forms. Use, for example,

```
pimd-prep explain VASP6
pimd-prep explain OPENMX
pimd-prep explain boundaries
```

for the registered boundary support.

The utility uses Python's standard `tomllib` module when available and supports the compatible `tomli` fallback on older Python. Python 3.11 or later is recommended. The canonical default file can be selected explicitly with `-defaults /path/to/input_default.dat`; otherwise the run-directory copy takes precedence over the packaged fallback.

Legacy NEW-style input can be migrated with

```
pimd-prep migrate input.dat
pimd-prep migrate input.dat -o pimd.toml
```

OLD-style input can also be migrated when the required `centroid.dat` and **<nspec>** information is available. OLD-style QMMM/ONIOM migration is intentionally refused because the old representation does not preserve the region/link-atom metadata needed by NEW-style hybrid `structure.dat`.

2.9 Workflow aliases

User-facing SLHMC maps to the engine combination <method> = PIHMC plus <ipotential> = DUAL. Its proposal dynamics is not thermostatted and its pimd-prep default is <nbead> = 1; Section 4.11 gives the details. User-facing QTST is the centroid free-energy workflow driven by polymers.x; it is not emitted as an engine <method> value.

2.10 Running the distributed examples

Each curated example contains its own pimd.toml and a small run.sh launcher. For example:

```
> cd examples/02_path_integral_dynamics/H2O/pimd_nvt_water
> ./run.sh
```

The launcher regenerates input.dat, locates the required executable, and temporarily links the single canonical examples/input_default.dat. If the executables are in a separate build directory, set

```
> export PIMD_BIN_DIR=/path/to/PIMD-2.8.0/build
> ./run.sh
```

MPI examples can be run, for example, with

```
> NP=8 ./run.sh
```

and MPI_LAUNCHER=srun can be selected on systems where Slurm is the preferred launcher.

The complete example index is examples/CATALOG.csv, and the mapping between manual features and representative examples is summarized in examples/MANUAL_COVERAGE.md. The example tree can be checked with

```
> cd examples
> ./validate_examples.py
```

This validation checks paths, required files, method/backend coverage, the single-default-file invariant, and reproducibility of generated input.dat files from their colocated pimd.toml files.

Note

The complete legacy keyword reference remains authoritative for the Fortran engine. The TOML layer is intentionally an input and validation layer on top of that interface, allowing PIMD 2.8.0 to modernize user workflows without breaking existing engine-level inputs.

3 Compatibility with older input formats

Starting with version 2.2.0, PIMD introduced a new initial-geometry format. The previous centroid.dat file was replaced by structure.dat. The structure.dat format is xyz-like, is described in Section 12.1.12, and includes both the number of atoms and their atomic species. Consequently, the keywords <natom> and <nspec> are no longer required in input.dat for normal NEW-style input, although they remain relevant in certain special cases.

The old centroid.dat format can still be used together with <natom> and <nspec> by setting <input_style> = OLD. The default is <input_style> = NEW, which reads structure.dat.

The newer structure format also simplifies workflows that need atom/species metadata explicitly, including hybrid potentials (QM/MM and ONIOM) and methods that use collective variables, such as constrained MD,

metadynamics, and AFED.

Legacy makefile build

The CMake procedure above is the recommended general build method for PIMD 2.8.0. The traditional `source/makefile` workflow is retained only for legacy interfaces or environments that still require the original build procedure. The old top-level machine-specific **`makefiles`** directory is no longer distributed.

Compiler, preprocessor, and math-library settings in `source/makefile` and its associated include files are machine dependent and may need to be edited for the local environment. If LAPACK is unavailable, the legacy makefile can be compiled with `LAPACK=-Dnolapack`, but complex matrix diagonalization is then unavailable. Methods that require it, such as phonon and nonadiabatic calculations, cannot be used in that build. Similarly, `PME=-Dnopme` disables the PME implementation. Backend-specific legacy procedures, where still required, are described in the corresponding interface sections.

3.1 Docker

This section describes how to use a preinstalled PIMD Docker image linked with QE and AENET.

- **Step 1:** Download Docker from the website appropriate for your operating system, for example, macOS:

```
https://hub.docker.com/editions/community/docker-ce-desktop-mac/
```

- **Step 2:** Create a working directory for Docker, for example, `test`:

```
mkdir ~/test
cd ~/test
```

- **Step 3:** Download the PIMD Docker image from the following website:

```
https://hub.docker.com/r/cometscome/pimd_aenet_qe
```

  To download the image, open a terminal and type:

```
docker pull cometscome/pimd_aenet_qe
```

- **Step 4:** Start the Docker container by typing:

```
docker run --shm-size=2gb -v ~/test:/home/pimder/home --rm -it cometscome/pimd_aenet_qe
```

  The shared memory size should be adjusted depending on the number of parallel processes you plan to use. After starting the container, the `test` directory can be accessed at `/home/pimder/home` inside Docker.
- **Step 5:** You are ready to go. Confirm that PIMD runs correctly by executing the following examples:

```
cd ~/pimd/examples/02_path_integral_dynamics/Si/pimd_nvt_qe/
mpirun -np 2 pimd.mpi.x

cd ~/pimd/examples/02_path_integral_dynamics/SiO2/pimd_nvt_aenet/
mpirun -np 2 pimd.mpi.x

cd ~/pimd/examples/07_self_learning_ml/SiO2/slhmc_nvt_qe_aenet/
mpirun -np 2 pimd.mpi.x
```

Part II

Simulation Methods

4 Methods

To verify that the setup is correct, it is recommended to begin with simple test calculations. In this section, the keywords required for each simulation method are explained through a series of input examples.

Core workflows

For the principal path-integral and rare-event workflows, see PIMD (Section 4.10.1), RPMD (Section 4.10.2), BCMD (Section 4.10.3), SLHMC (Section 4.11), QTST with `polymers.x` (Section 5), and metadynamics (Section 5.2).

The distributed PIMD 2.8.0 examples are organized by method family under `examples/`; `CATALOG.csv` is the authoritative index. Each simulation directory contains an authoritative `pimd.toml`; `pimd-prep` generates the legacy `input.dat`. Some numerical values printed in the explanatory blocks below are intentionally more extensive than the shortened regression examples, but the referenced directories and keyword semantics are current.

In most examples, a single water molecule described by the built-in q-TIP4P/F potential [43] is used. q-TIP4P/F is the default WATER model; SPC/F and RWK remain available through `<model_water>`.

The built-in water potential can be specified using the keyword:

```
<ipotential>
WATER
```

This is a special case in which the water-model setup and any model-specific terms are implemented internally in the PIMD code, while the common periodic electrostatic machinery is shared with CFF. For all other choices of `<ipotential>`, additional files defining the potential must be prepared by the user. These files and their formats are explained in Section 6.

4.1 Static calculation (STATIC)

The smallest distributed example is a static energy/force calculation for one q-TIP4P/F water molecule. Begin with the curated example rather than by constructing a legacy input file from scratch:

```
> cd examples/01_basics/H2O/static_water
> cat pimd.toml
```

Its authoritative user input is intentionally minimal:

```
[files]
structure = "structure.dat"

[potential]
type = "WATER"
```

The omitted run settings are supplied by the canonical defaults. In particular, the default method is <method> = STATIC, the default ensemble is <ensemble> = NONE, and the default boundary condition is <iboundary> = 0 (an isolated system).

Run the example with

```
> ./run.sh
```

The launcher regenerates input.dat from pimd.toml, links the canonical examples/input_default.dat for the duration of the run, locates pimd.x, and writes the program output to standard.out. If the executable is in a separate build directory, set PIMD_BIN_DIR as described in Section 1.

Initial structure

A new calculation also needs structure.dat. For ordinary NEW-style input, its basic nonperiodic form is

```
N
unit
symbol  x  y  z
symbol  x  y  z
...
```

where N is the number of atoms, unit is either BOHR or ANGSTROM, and each subsequent line gives an atomic symbol and its Cartesian coordinates. Periodic EXTXYZ input and the complete format are described in Section 12.1.12.

Equivalent engine-level keywords

The method chapters in this manual show input.dat keyword blocks because they define the semantics read directly by the Fortran engine. In a 2.8.0 curated example, however, these settings should normally be expressed in pimd.toml and converted with pimd-prep.

For the present calculation, the essential engine-level settings are

```
<method>
STATIC
```

```
<ipotential>
WATER
```

```
<iboundary>
0
```

Here, STATIC selects a static calculation, WATER selects the built-in WATER potential (q-TIP4P/F by default), and 0 selects a free boundary condition. WATER also supports periodic boundaries. The H_2O connectivity is recognized automatically, so the atom order need not be O, H, H.

The generated input.dat does not need to repeat values that already match input_default.dat. This is why the distributed pimd.toml can remain much shorter than a fully explicit legacy input. The order of legacy keyword blocks is arbitrary, but each keyword tag must begin in the first column; see Section 11.

Program output

A successful calculation produces output similar to the following. The exact banner and numerical formatting may vary slightly between builds:

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

PIMD version x.x.x

Author:        M. Shiga

Last Updated:  Xxx. xx, 20xx.

Copyright(C) 20xx  M. Shiga  All rights reserved.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


Method:   static.

Model:    water.

Ensemble: none.

Atomic positions read from structure.dat. Molecular configuration initialized.

Free boundary condition - isolated system.

================================================================================
                          potential energy values
  bead                hartree               kcal/mol                  kJ/mol
--------------------------------------------------------------------------------
     1             0.00036497               0.229023               0.958230
--------------------------------------------------------------------------------

Normal termination of pimd.
```

The static calculation writes energy and force information, including `forces.out`; the output-file formats are described in Section 12.2. Restart data such as `geometry.ini` may also be created according to the active settings.

To repeat a distributed example from a clean state, remove output, trajectory, and restart files created by the previous run, then execute `./run.sh` again. For manual engine-level execution, the corresponding command is simply

```
> pimd.x
```

provided that the required `input.dat`, `input_default.dat`, and structure/restart files are present in the execution directory.

4.2 Geometry optimization (GEOOPT)

Geometry optimization is performed using the limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [2, 3].

As an example, a set of input files for the q-TIP4P/F water model is provided. First, copy the example to the directory *run*:

```
> cp -r examples/03_optimization_properties/H2O/geoopt_water run
```

and move to that directory:

```
> cd run/geoopt_water/
```

To prepare the file `input.dat`, all keywords described in Section 4.1 are required. In addition, the following keywords are important.

```
<method>
GEOOPT
```

This keyword sets the simulation method. In the present case, `GEOOPT` specifies a geometry optimization.

```
<nstep>
10
```

This keyword sets the maximum number of optimization steps. Here, the maximum number of geometry optimization steps is set to 10.

```
<iread_exit>
1
```

This keyword sets the interval (in steps) at which the code checks for the existence of the file `exit.dat` to allow for a soft exit (see Section 4.3). In this example, the check is performed at every step.

```
<iprint_std>
1
```

This keyword sets the print interval of the standard output file `standard.out`. Here, output is printed at every step.

```
<iprint_rest>
1
```

This keyword sets the interval for generating (and overwriting) the restart files `*.ini`. In this case, restart files are written at every step.

```
<iprint_trj>
1
```

This keyword sets the print interval of the standard trajectory file `trj.out`. Here, the trajectory is written at every step.

```
<iprint_xyz>
1
```

This keyword sets the print interval of the trajectory file in XYZ format, `trj.xyz`. In this example, the XYZ trajectory is written at every step.

In new TOML input, the four L-BFGS criteria are named explicitly as `max_displacement`, `rms_displacement`, `max_force`, and `rms_force` in `[optimization]`; see Section 2. The following is the corresponding legacy engine block.

```
<params_lbfgs>
  1.d-4,  1.d-5,  1.d-5,  1.d-6
```

This keyword specifies the convergence criteria for the L-BFGS algorithm. The four values correspond to:

- the maximum atomic displacement (bohr),
- the root-mean-square (RMS) atomic displacement (bohr),
- the maximum force component (hartree/bohr),
- the RMS force (hartree/bohr).

Before starting the calculation, make sure that the files `input.dat`, `input_default.dat`, and `structure.dat` are present in the execution directory. To run the geometry optimization, type:

```
> pimd.x
```

During execution, information similar to the following will be printed to the screen:

```
================================================================================
    step     energy [au]    max displ    rms displ     max force     rms force
                              0.00001      0.00000      0.000100      0.000010
--------------------------------------------------------------------------------
       0      0.00036497      0.00000      0.00000      0.009221      0.001357
       1      0.03310297      0.75483      0.05153      0.038251      0.006037
       2      0.00021197      0.71993      0.04915      0.008308      0.001669
       3      0.00005349      0.01012      0.00117      0.003593      0.000618
       4      0.00000292      0.01351      0.00151      0.000759      0.000114
       5      0.00000005      0.00335      0.00040      0.000184      0.000026
       6      0.00000000      0.00046      0.00004      0.000007      0.000001
       7      0.00000000      0.00001      0.00000      0.000000      0.000000
```

Upon convergence, restart files such as `geometry.ini` and `step.ini` are written, and the program terminates normally.

The output files `*.out` and the restart files `*.ini` are described in Section 12.2 and Section 12.3, respectively.

NOTE: In some cases, the GEOOPT calculation may stop with an error message indicating that the convergence criteria are too strict. If this happens, it is recommended to relax the convergence thresholds by increasing the values specified in the keyword `<params_lbfgs>`.

4.3 Restarts and interruptions

For each run, restart files are automatically created at the end of the calculation, unless this feature is intentionally disabled by setting the keyword `<iprint_rest>` to `-1`. To prepare for unexpected interruptions, such as a sudden power outage, restart files can also be written periodically during the run. This is controlled by the keyword `<iprint_rest>`, which specifies the print interval of the restart files.

It is strongly recommended to save restart files as well as output files from time to time, for example:

```
> cp *.ini *.out *.xyz somewhere
```

IMPORTANT: Restart files are always overwritten, replacing older ones.

If you wish to stop a job during execution, the *soft exit* mechanism can be used. To do this, simply create an empty file named `exit.dat` in the execution directory:

```
> touch exit.dat
```

During the run, the code periodically checks for the existence of `exit.dat`, with a frequency specified by the keyword **<iread_exit>**. When `exit.dat` is detected, the job terminates as soon as possible after generating the complete set of restart files. The file `exit.dat` is automatically removed at the end of the job, so that restarting the next run is straightforward.

NOTE: In some cases, especially for inexpensive classical potentials such as CFF, frequent disk access to check `exit.dat` may slow down the calculation. This behavior can be controlled by adjusting the keyword **<iread_exit>**. For example, setting **<iread_exit>** = 100 causes the check to be performed every 100 steps. The soft exit mechanism can be completely disabled by setting **<iread_exit>** = -1.

The current step number is written to a restart file called `step.ini`. To restart a calculation, the number of steps specified by the keyword **<nstep>** in the file `input.dat` must always be larger than the current step recorded in `step.ini`. Therefore, to extend a trajectory after a job has completed, **<nstep>** must be increased accordingly. If this is not done, the restarted job will terminate immediately, since the target number of steps has already been reached.

To restart the job, simply execute:

```
> pimd.x
```

or, to run it in the background,

```
> pimd.x >> monitor.out &
```

At the beginning of the run, the code automatically searches for restart files such as `geometry.ini`, `step.ini`, and others in the execution directory. If these files are not found, the default initial conditions are used for the restarted calculation.

4.4 Normal mode analysis (NMA)

In normal mode analysis, the Hessian matrix is obtained numerically as the second derivative of the potential energy surface with respect to atomic positions.

As an example, a set of input files for q-TIP4P/F water is available, so copy it to the *run* directory using the following command:

```
> cp -r examples/03_optimization_properties/H2O/nma_water run
```

Then, navigate to that directory:

```
> cd run/nma_water/
```

Before running the simulation, the optimized structure should be prepared. This can be done either manually by creating a `structure.dat` file or by using the restart file `geometry.ini`, which is produced at the end of a GEOOPT calculation.

To prepare the input file `input.dat`, all the keywords in Section 4.1 are required. In addition, the following keywords must be specified:

```
<method>
NMA
```

This keyword sets the simulation method; in this case, it corresponds to normal mode analysis.

```
<fdiff>
1.d-4
```

This keyword sets the finite difference shift of the geometry in bohr, which is used to compute the Hessian matrix. Here, a value of 10^{-4} bohr is used. The results are written to the output file named `nma.out` in a GAUSSIAN 98-like format.

4.5 Steepest descent (SD)

From a given starting point, the steepest descent trajectory is computed as the minimum energy path in the mass-weighted coordinate space. When the starting geometry is a saddle point, the trajectory corresponds to the intrinsic reaction coordinate (IRC). The theoretical background of this method is briefly explained in Section 13.2.

As an example, a set of input files for q-TIP4P/F water is available. To use it, copy the files to the directory *run*:

```
> cp -r examples/03_optimization_properties/H2O/sd_water run
```

and enter that directory:

```
> cd run/sd_water
```

To prepare the file `input.dat`, all the keywords in Sections 4.1 and 4.2 are important. In addition, the following keywords are also required:

```
<method>
SD
```

This keyword sets the simulation method. In the present case, "SD" corresponds to the steepest descent method.

```
<nstep>
10
```

This keyword sets the number of steps. In an SD calculation, this corresponds to the number of steepest descent steps.

```
<dt>
0.01d0
```

This keyword sets the SD step size in units of hartree$^{0.5}$ femtoseconds. The output files named `*.out` and the restart files named `*.ini` will be explained in Section 12.2 and Section 12.3, respectively.

4.6 Box optimization (BOXOPT)

Box optimization can be performed for a system with periodic boundary conditions. The theoretical background of this method is briefly explained in Section 13.3. The limited-memory Broyden-Fletcher-Goldfarb-

Shanno (BFGS) algorithm [2, 3] is used.

As an example, a periodic diamond cell using the Tersoff potential is available. Let us copy it to the directory `run`:

```
> cp -r examples/03_optimization_properties/diamond/boxopt_tersoff run
```

and enter that directory:

```
> cd run/boxopt_tersoff
```

To prepare the file `input.dat`, all the keywords in Sections 4.1 and 4.2 are important. In addition, the following keywords are also important.

```
<method>
BOXOPT
```

This keyword sets the simulation method. In the present case, "BOXOPT" corresponds to box optimization.

```
<iprint_box>
1
```

This keyword sets the print interval of the box matrix to the file `box.out`. In the present case, "1" means every step.

```
<iprint_rest>
1
```

This keyword sets the print interval of the restart files `*.ini`. In the present case, "1" means every step.

```
<fdiff>
1.d-4
```

This keyword sets the finite difference shift of the box matrix elements in Bohr, which is used to compute the virial matrix. In the present case, 10^{-4} Bohr is used.

```
<iboundary>
1
 33.20000000   0.00000000   0.00000000
  0.00000000  33.20000000   0.00000000
  0.00000000   0.00000000  33.20000000
```

This keyword sets the boundary condition and the initial box. In the present case, "1" means periodic boundary conditions, and the following three lines define the matrix of the initial box. Here, a cubic box with a side length of 33.2 Bohr is used. The output files `*.out` and the restart files `*.ini` will be explained in Section 12.2 and Section 12.3, respectively.

4.7 Full optimization (FULLOPT)

To optimize both the box and geometry, one can use the following method:

```
<method>
FULLOPT
```

4.8 Static elastic constants (ELASTIC)

The static elastic constants can be computed using the following method:

```
<method>
ELASTIC
```

This is usually done after optimizing both the box and geometry (FULLOPT). In this case, the files `geometry.ini` and `box.ini` are read at the beginning of the run. The static elastic constants are computed as the numerical derivatives of the stress tensor. The accuracy can be controlled by setting the finite difference parameter:

```
<fdiff>
1.d-4
```

In PIMD, the elastic constants can be computed for orthorhombic boxes only.

4.9 Classical molecular dynamics (MD)

In the PIMD code, classical molecular dynamics (MD) simulations are implemented for a variety of statistical ensembles, including the microcanonical (NVE), canonical (NVT), isobaric–isothermal (NPT), and isotensional–isothermal (NtT) ensembles.

In the NVT, NPT, and NtT ensembles, temperature control is achieved using Nosé–Hoover (NH), Nosé–Hoover chain (NHC), or massive Nosé–Hoover chain (MNHC) thermostat methods. These approaches enable efficient and stable sampling of the target ensemble by extending the system with additional thermostat degrees of freedom.

For simulations of isotropic liquids under pressure, a cubic simulation box can be employed in the NPT ensemble. For anisotropic solids, a general parallelepiped (triclinic) hexahedral simulation box is available, allowing independent fluctuations of the box vectors. This formulation is suitable for treating crystals and low-symmetry materials.

In simulations of anisotropic solids under external mechanical loading, the NtT ensemble can be used. In this case, the external thermodynamic tension $\overleftrightarrow{\mathbf{t}}$ is specified as a real symmetric 3×3 matrix, enabling the application of general stress states beyond hydrostatic pressure.

Details of the equations of motion and numerical integration schemes used for each ensemble are provided in the corresponding subsections, together with representative input examples.

4.9.1 NVE molecular dynamics

In the NVE ensemble, the equation of motion is given by Newton's equation. In the PIMD code, it is integrated using the velocity Verlet algorithm. The theoretical background is briefly explained in Section 13.9.

As an example, a set of input files for q-TIP4P/F water is available. Copy it to the directory *run* by typing

```
> cp -r examples/01_basics/H2O/md_nve_water run
```

and then move to the directory:

```
> cd run/md_nve_water/
```

To prepare the file `input.dat`, all the keywords described in Sections 4.1 and 4.2 are required. In addition, the following keywords are important.

```
<method>
```

```
MD
```

This keyword sets the simulation method. In the present case, `MD` corresponds to classical molecular dynamics.

```
<ensemble>
NVE
```

This keyword sets the statistical ensemble. Here, `NVE` denotes the microcanonical ensemble.

```
<dt>
0.25d0
```

This keyword sets the time step in femtoseconds. In the present case, the time step is 0.25 fs.

```
<nstep>
100
```

This keyword sets the total number of MD steps.

```
<temperature>
300.d0
```

This keyword sets the temperature in Kelvin. If the restart file `geometry.ini` exists, the atomic positions and velocities are read from this file. Otherwise, the atomic positions are read from `structure.dat`, and the velocities are generated according to the Maxwell–Boltzmann distribution at this temperature. Note that, in NVE molecular dynamics, the temperature is not controlled during the run and is only used for the initialization of velocities.

```
<corrections>
1  1
```

This keyword sets the translational and rotational corrections applied to the system. In the present case, both the translational and rotational corrections are applied at the initial step only.

```
<iprint_dip>
1
```

This keyword sets the print interval of the dipole moment. Here, the dipole moment is printed at every step.

```
<iprint_rdf>
10
```

This keyword sets the print interval of the atomic pair distribution function. In the present case, it is printed every ten steps. When `<iprint_rdf>` is activated, it is also useful to specify

```
<params_rdf>
1.d0  5.d0  0.01d0
```

This keyword sets the mesh parameters for the atomic pair distribution function. In the present case, the mesh points range from 1.0 bohr to 5.0 bohr with an increment of 0.01 bohr.

The output files `*.out` and the restart files `*.ini` are explained in Sections 12.2 and 12.3, respectively.

4.9.2 NVT molecular dynamics

In the PIMD code, molecular dynamics for the NVT ensemble is based on massive Nosé-Hoover chain thermostats [8, 10, 11]. The equations of motion are solved using the reversible RESPA algorithm [12, 13]. The theoretical background of NVT molecular dynamics is briefly explained in Section 13.10.

As an example, a set of input files for q-TIP4P/F water is available. Copy it to the directory *run*:

```
> cp -r examples/01_basics/H2O/md_nvt_water run
```

Then navigate to that directory:

```
> cd run/md_nvt_water
```

To prepare the file `input.dat`, all the keywords given in Sections 4.1 and 4.2 are important. In addition, the following keywords are required:

```
<method>
MD
```

This keyword sets the simulation method. In this case, "MD" corresponds to classical molecular dynamics.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble. Here, "NVT" corresponds to the NVT ensemble.

```
<dt>
0.25d0
```

This keyword sets the step size in femtoseconds.

```
<nstep>
100
```

This keyword sets the number of steps.

```
<temperature>
300.d0
```

This keyword sets the temperature in kelvin. If the restart file `geometry.ini` exists, atomic positions and velocities are read from this file. Otherwise, positions are read from `centroid.dat`, and velocities are generated at this temperature for the initial run. The temperature is controlled during the simulation.

```
<corrections>
0  0
```

This keyword sets the translational and rotational corrections. In this case, the corrections are not applied at the initial step.

```
<iprint_dip>
10
```

This keyword sets the print interval of the dipole moment. Here, it is printed every ten steps.

```
<iprint_rdf>
10
```

This keyword sets the print interval of the atomic pair distribution. Here, it is printed every ten steps.

```
<params_rdf>
1.d0  5.d0  0.01d0
```

This keyword is valid when <iprint_rdf> is activated. It sets the mesh parameters for the atomic pair distribution. Here, the mesh points range from 1.0 Bohr to 5.0 Bohr with an increment of 0.01 Bohr.

```
<bath_type>
MNHC
```

This keyword sets the type of thermostats. When NHC is chosen, a Nosé-Hoover thermostat chain (NHC) is attached to the whole system. When MNHC is chosen, massive Nosé-Hoover thermostat chains (MNHC) are attached to each degree of freedom.

```
<nnhc>
4
```

This keyword sets the chain length of the MNHC thermostats to four.

```
<ncolor>
1
```

This keyword sets the multiplicity of the MNHC thermostats to one.

```
<time_bath>
10.d0
```

This keyword controls the mass of the thermostats, which is related to the characteristic time scale of the system in femtoseconds, as designated by this keyword. The mass of the thermostats can be set close to the fastest vibrational time scale of the system. Here, it is set to 10 femtoseconds, which is close to the oscillation period of 8.881 femtoseconds for the anti-symmetric OH stretching mode of water, corresponding to 3756 cm^{-1}.

```
<nys>
5
```

This keyword controls the step size for the MNHC thermostats, with the thermostats updated five times per step.

The output files `*.out` and the restart files `*.ini` are explained in Section 12.2 and Section 12.3, respectively.

4.9.3 Constrained molecular dynamics

For new TOML input, constraints should be written as numbered `[constraints.N]` tables; see Section 2. This avoids copying the raw <ncons> and <params_cons> blocks and permits `output = false` instead of the historical `-1` print sentinel. The legacy engine-level form is documented below for compatibility.

In NVT and NVE molecular dynamics simulations, harmonic constraints can be added to calculate free energy gradients, which are the negative mean forces imposed on each constraint. The free energy profile can be obtained by integrating with respect to the shifts in the constraints. To accomplish this, independent trajectories (beads) with different constraint equilibrium positions are calculated. The following example is for a water molecule:

```
<nbead>
128
```

This keyword sets the number of independent trajectories (beads) with different constraints; in this case, “128”.

```
<ncons>
1
1  1  2  1.3  2.5
```

This keyword sets the number of constraints, followed by the parameters of each constraint. In this case, there is one constraint of type “1” (interatomic distance) between atoms “1” and “2”. As a function of the bead number, the position of the constraint is shifted linearly from 1.3 bohr to 2.5 bohr.

```
<params_cons>
1    1.d0
2    4.d-4
3    1.d-4
4    1.d0
5    1.d0
6    1.d0
7    1.d0
8    1.d0
```

This keyword sets the force constants for each type of constraint. In this case, the first line sets the force constant of type “1” (interatomic distance) to be 1.0 hartree/bohr^2.

```
<iprint_cons>
10
```

This keyword sets the print interval of constraint information to the file `cons.out`. In this case, it is every “10” steps.

4.9.4 NPT molecular dynamics in a cubic box

In the PIMD code, molecular dynamics in the NPT ensemble is implemented using massive Nosé–Hoover chain thermostats [8, 10, 16, 17, 11]. The equations of motion are integrated using the reversible RESPA algorithm [12, 13, 14, 15]. The theoretical background of NPT molecular dynamics is briefly described in Section 13.11.

As an example, a set of input files for liquid water is available. First, copy them to the directory *run*:

```
> cp -r examples/02_path_integral_dynamics/water_64/pimd_npt_tip4p run
```

and move to that directory:

```
> cd run/pimd_npt_tip4p
```

To prepare the file `input.dat`, all keywords described in Sections 4.1 and 4.2 are required. In addition, the following keywords are important.

```
<method>
PIMD
```

This keyword sets the simulation method. In general, “PIMD” corresponds to path integral molecular dynamics. In the special case of a single bead, <nbead> = 1 (the default), it reduces to classical molecular dynamics.

```
<ensemble>
NPT
```

This keyword specifies the statistical ensemble. Here, “NPT” denotes the isothermal–isobaric ensemble.

```
<npt_type>
CUBIC2
```

This keyword defines the shape of the simulation box and the equations of motion for the barostat. For a cubic box, either Andersen’s formulation (“CUBIC1”) or Martyna’s formulation (“CUBIC2”) can be used. In this example, a cubic box with Martyna’s equation of motion is chosen.

```
<dt>
0.25d0
```

This keyword sets the time step in femtoseconds.

```
<nstep>
100
```

This keyword sets the total number of MD steps.

```
<temperature>
300.d0
```

This keyword sets the temperature in kelvin. If the restart file `geometry.ini` exists, atomic positions and velocities are read from it. Otherwise, atomic positions are read from `structure.dat`, and velocities are generated from a Maxwell–Boltzmann distribution at this temperature. The temperature is controlled throughout the simulation.

```
<corrections>
0  0
```

This keyword controls translational and rotational corrections. In the present case, no corrections are applied at the initial step.

```
<bath_type>
MNHC
```

This keyword specifies the thermostat type. When “NHC” is selected, a Nosé–Hoover chain thermostat is attached to the whole system. When “MNHC” is selected, massive Nosé–Hoover chain thermostats are attached to each degree of freedom.

```
<nnhc>
4
```

This keyword sets the chain length of the MNHC thermostats. Here, a chain length of four is used.

```
<ncolor>
1
```

This keyword sets the multiplicity of the MNHC thermostats. In this example, the multiplicity is one.

```
<time_bath>
```

```
10.d0
```

This keyword controls the thermostat mass through the characteristic time scale of the system, given in femtoseconds. A reasonable value is close to the fastest vibrational time scale of the system. Here, it is set to 10 fs, which is comparable to the oscillation period (8.881 fs) of the anti-symmetric OH stretching mode of water (3756 cm^{-1}).

```
<nys>
5
```

This keyword controls the number of thermostat updates per MD step. In this case, the thermostats are updated five times per step.

```
<pressure>
100.d0
```

This keyword sets the external pressure in megapascals. Here, the pressure is set to 100 MPa.

```
<time_baro>
1000.d0
```

This keyword controls the mass of the barostat through the characteristic time scale of volume fluctuations, in femtoseconds. In this example, it is set to 1000 fs.

```
<iprint_box>
10
```

This keyword sets the print interval of the box matrix to the output file `box.out`. Here, the box matrix is written every ten steps.

```
<iboundary>
1
 37.2276d+00,  0.0000d+00,  0.0000d+00
  0.0000d+00, 37.2276d+00,  0.0000d+00
  0.0000d+00,  0.0000d+00, 37.2276d+00
```

This keyword specifies the boundary condition and the initial simulation box. Here, "1" indicates periodic boundary conditions with box vectors in bohr. The following three lines define the initial box matrix. A cubic box with a side length of 37.2276 bohr is used.

```
<iprint_trj>
10
```

This keyword sets the print interval of the trajectory to the output file `trj.out`. In this case, the trajectory is written every ten steps.

The output files `*.out` and the restart files `*.ini` are explained in Sections 12.2 and 12.3, respectively.

4.9.5 NPT and NtT molecular dynamics in a hexahedron box

In the PIMD code, molecular dynamics for the NPT and NtT ensembles (NtT: isotensional ensemble) are based on massive Nosé-Hoover chain thermostats [8, 10, 11]. The equations of motion are solved using the reversible RESPA algorithm [12, 13, 14, 15]. The theoretical background of NtT molecular dynamics is briefly explained in Section 13.11.

Setups for NPT MD simulation with a hexahedron box.. As an example, a periodic Ni–H EAM system is available. Copy it to the directory *run*:

```
> cp -r examples/02_path_integral_dynamics/NiH/pimd_ntt_eam run
```

and navigate to that directory:

```
> cd run/pimd_ntt_eam
```

To prepare the file `input.dat`, all the keywords given in Sections 4.1 and 4.2 are important. In addition, the following keywords are required.

```
<method>
PIMD
```

This keyword sets the simulation method. Generally, "PIMD" corresponds to path integral molecular dynamics, but in the special case of a single bead (`nbead=1` by default), it corresponds to classical molecular dynamics.

```
<ensemble>
NPT
```

This keyword sets the statistical ensemble. In this case, "NPT" corresponds to the NPT ensemble.

```
<npt_type>
PPHEX
```

This keyword sets the shape of the simulation box. Here, the box is set to a parallel-piped hexahedron.

```
<dt>
0.25d0
```

This keyword sets the step size in femtoseconds.

```
<nstep>
100
```

This keyword sets the number of steps.

```
<temperature>
300.d0
```

This keyword sets the temperature in kelvin. If the restart file `geometry.ini` exists, atomic positions and velocities are read from this file. Otherwise, positions are read from `structure.dat`, and velocities are generated at this temperature for the initial run. The temperature is controlled during the simulation.

```
<corrections>
0  0
```

This keyword sets the translational and rotational corrections. In this case, the corrections are not applied at the initial step.

```
<bath_type>
MNHC
```

This keyword sets the type of thermostats. When “NHC” is chosen, a Nosé-Hoover thermostat chain (NHC) is attached to the whole system. When “MNHC” is chosen, massive Nosé-Hoover thermostat chains (MNHC) are attached to each degree of freedom.

```
<nnhc>
4
```

This keyword sets the chain length of the MNHC thermostats, which is four in this case.

```
<ncolor>
1
```

This keyword sets the multiplicity of the MNHC thermostats; here, it is one.

```
<time_bath>
10.d0
```

This keyword controls the mass of the thermostats. The mass of the thermostats is related to the characteristic time scale of the system, in femtoseconds, which is specified by this keyword. This parameter can be set close to the fastest vibrational time scale of the system. Here it is set to 10 femtoseconds, which is close to the oscillation period of 8.881 femtoseconds for the anti-symmetric OH stretching mode of water, which corresponds to 3756 cm^{-1}.

```
<nys>
5
```

This keyword controls the step size for the MNHC thermostats. In this case, the thermostats are updated five times per step.

```
<pressure>
100.d0
```

This keyword sets the pressure in megapascals; here, it is 100 MPa.

```
<time_baro>
1000.d0
```

This keyword controls the mass of the barostat. The mass of the barostat is related to the characteristic time scale of the volume fluctuation, in femtoseconds, which is specified by this keyword. Here it is set to 1000 femtoseconds.

```
<iprint_box>
10
```

This keyword controls the print interval of the box matrix to the output file `box.out`. Here, it is set to every ten steps.

```
<iboundary>
1
 33.20000000   0.00000000   0.00000000
  0.00000000  33.20000000   0.00000000
  0.00000000   0.00000000  33.20000000
```

This keyword sets the boundary condition and the initial box. Here, “1” means periodic boundary, and the following three lines are the matrix of the initial cubic box with side length 33.2 bohr.

```
<iprint_trj>
10
```

This keyword controls the print interval of the trajectory to the output file `trj.out`. Here, it is set to every ten steps.

Setups for NtT MD simulation with a hexahedron box.. The setups in NtT MD are the same as those in NPT MD, except for the following keywords. The keyword `<pressure>` should be replaced by

```
<tension>
    0.0    0.0    0.0
    0.0    0.0    0.0
    0.0    0.0 1000.0
```

This keyword sets the tension tensor in megapascals; here, 1000 MPa is the zz element. Note that the tension tensor must be provided as a 3×3 real symmetric matrix. Then, designate the keyword

```
<iboundary>
1
 33.20000000   0.00000000   0.00000000
  0.00000000  33.20000000   0.00000000
  0.00000000   0.00000000  33.20000000
```

This keyword sets the boundary condition and the initial box. In the NtT ensemble, this will be the reference box matrix under zero tension. To provide the reference box, one must run a preliminary simulation under the same conditions (same temperature, etc.) with zero tension. The reference box matrix is evaluated as the average box matrix from this preliminary simulation. Within this simulation, the reference box matrix is arbitrary.

The output files **.out* and the restart files **.ini* are explained in Section 12.2 and Section 12.3, respectively.

4.10 Path integral simulations (PI)

In the PIMD code, the following path-integral methods are implemented:

- Path integral molecular dynamics (PIMD) with the second-order Suzuki–Trotter expansion [23].
- Ring polymer molecular dynamics (RPMD). [24].
- Brownian chain molecular dynamics (BCMD) [33].
- Centroid molecular dynamics (CMD) [26].
- Thermostatted ring polymer molecular dynamics (TRPMD) [34].
- Path integral hybrid Monte Carlo (PIHMC) with the second-order [23] and fourth-order Suzuki–Trotter expansion. [35]
- Centroid intrinsic reaction coordinate (CIRC). [32]

The theoretical background of these methods is briefly explained in Section 13.11.

4.10.1 Path integral molecular dynamics (PIMD)

As an example, a set of input files for q-TIP4P/F water is available. To use it, copy it to the directory *run*:

```
> cp -r examples/02_path_integral_dynamics/H2O/pimd_nvt_water run/
```

and navigate to that directory:

```
> cd run/pimd_nvt_water/
```

To prepare the file `input.dat`, all the keywords given in Sections 4.1 and 4.2 are important. Additionally, the following keywords are important:

```
<method>
PIMD
```

This keyword sets the simulation method. In this case, "PIMD" corresponds to path integral molecular dynamics.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble. In this case, "NVT" corresponds to the NVT ensemble.

```
<dt>
0.25d0
```

This keyword sets the step size in femtoseconds. In this case, the step size is 0.25 femtoseconds.

```
<nstep>
100
```

This keyword sets the number of steps. In this case, the simulation runs for 100 steps.

```
<temperature>
300.d0
```

This keyword sets the temperature in Kelvin. In this case, the temperature is 300 K. The temperature is controlled during the PIMD simulation.

```
<corrections>
0  0
```

This keyword sets the translational and rotational corrections. In this case, the corrections are not applied at the initial step.

```
<iprint_dip>
10
```

This keyword sets the print interval for the dipole moment. In this case, the dipole moment is printed every ten steps.

```
<iprint_rdf>
10
```

This keyword sets the print interval for the atomic pair distribution. In this case, the atomic pair distribution is printed every ten steps.

```
<params_rdf>
1.d0  5.d0  0.01d0
```

This keyword sets the mesh parameters for the atomic pair distribution. In this case, the mesh points range from 1.0 bohr to 5.0 bohr with an increment of 0.01 bohr.

```
<bath_type>
MNHC
```

This keyword sets the type of thermostats. When NHC is chosen, a Nosé–Hoover thermostat chain (NHC) is attached to the whole system. When "MNHC" is chosen, massive Nosé–Hoover thermostat chains (MNHC) are attached to each degree of freedom.

```
<nnhc>
4
```

This keyword sets the chain length of the MNHC thermostats. In this case, the chain length is four.

```
<ncolor>
1
```

This keyword sets the multiplicity of the MNHC thermostats. In this case, the multiplicity is one.

```
<time_bath>
10.d0
```

This keyword controls the mass of thermostats. The mass of thermostats is related to the characteristic time scale of the system, in femtoseconds, which is designated by this keyword. Here it is set to 10 femtoseconds.

```
<nys>
5
```

This keyword controls the step size for the MNHC thermostats. In the present case, thermostats are updated 5 times per step.

```
<nbead>
32
```

This keyword sets the number of beads, in the present case, 32 beads.

```
<nref>
1
```

This keyword controls the step size for updating the harmonic forces in path integrals. In the present case, "1" means the harmonic forces are updated once per step.

```
<beadspread>
500.d0
```

This keyword controls the bead spread for the initial configuration.

If the restart file `geometry.ini` exists, the positions and velocities of the beads are read from this file. Otherwise, the centroid positions being read from `structure.dat`, all the positions of beads are created randomly around the centroid positions with the standard deviation controlled by the keyword **<beadspread>**, while the velocities are generated at this temperature as an initial run.

The output files `*.out` and the restart files `*.ini` will be explained in Section 12.2 and Section 12.3, respectively.

4.10.2 Ring polymer molecular dynamics (RPMD)

As an example, a set of input files for the q-TIP4P/F water is available. So, let us copy it to the directory *run*:

```
> cp -r examples/02_path_integral_dynamics/H2O/rpmd_water run
```

and enter that directory:

```
> cd run/rpmd_water
```

To prepare the file `input.dat`, all the keywords given in Sections 4.1 and 4.2 are important. In addition, the following keywords are also important.

```
<method>
RPMD
```

This keyword sets the simulation method. In the present case, "RPMD" corresponds to ring polymer molecular dynamics.

```
<ensemble>
NVE
```

This keyword sets the statistical ensemble. In the present case, "NVE" corresponds to the NVE ensemble.

```
<dt>
0.25d0
```

This keyword sets the step size in femtoseconds. In the present case, the step size is 0.25 femtoseconds.

```
<nstep>
100
```

This keyword sets the number of steps. In the present case, there are 100 steps.

```
<temperature>
300.d0
```

This keyword sets the temperature in Kelvin.

```
<corrections>
2  2
```

This keyword sets the translational and rotational corrections. In the present case, both translation and rotation corrections are applied at each step.

```
<iprint_dip>
10
```

This keyword sets the print interval of the dipole moment. In this case, the dipole moment is printed every ten steps.

```
<iprint_rdf>
10
```

This keyword sets the print interval of the atomic pair distribution. In this case, the atomic pair distribution is printed every ten steps.

```
<params_rdf>
1.d0  5.d0  0.01d0
```

This keyword is valid when <iprint_rdf> is activated. It sets the mesh parameters for the atomic pair distributions. In the present case, the mesh points range from 1.0 bohr to 5.0 bohr with an increment of 0.01 bohr.

```
<nbead>
32
```

This keyword sets the number of beads. In the present case, there are 32 beads.

```
<beadspread>
500.d0
```

This keyword controls the bead spread used to generate the initial configuration.

If the restart file `geometry.ini` from a previous PIMD run exists, the atomic positions and velocities are read from this file (recommended in RPMD). Otherwise, the centroid positions are read from `structure.dat`, and all bead positions are generated randomly around the centroid positions with a standard deviation controlled by the keyword <beadspread>, while the velocities are generated at this temperature as an initial run (not recommended in RPMD). The temperature is not controlled during the RPMD run since it is an NVE dynamics.

The output files `*.out` and the restart files `*.ini` will be explained in Section 12.2 and Section 12.3, respectively.

4.10.3 Brownian chain molecular dynamics (BCMD)

The following keywords are important:

```
<method>
BCMD
```

This keyword sets the simulation method. In the present case, "BCMD" corresponds to Brownian chain molecular dynamics.

```
<ensemble>
NVE
```

This keyword sets the statistical ensemble. In the present case, "NVE" corresponds to the NVE ensemble.

```
<irandom>
1
```

This keyword sets the random number seed. In the present case, "1" sets a time-dependent seed.

4.10.4 Centroid molecular dynamics (CMD)

As an example, a set of input files for the q-TIP4P/F water model is available. Let us copy it to the directory *run*:

```
> cp -r examples/02_path_integral_dynamics/H2O/cmd_water run
```

Then, navigate to that directory:

```
> cd run/cmd_water
```

To prepare the file `input.dat`, all the keywords given in Sections 4.1 and 4.2 are important. In addition, the following keywords are also important:

```
<method>
CMD
```

This keyword sets the simulation method. In the present case, "CMD" corresponds to centroid molecular dynamics.

```
<ensemble>
NVE
```

This keyword sets the statistical ensemble. In the present case, "NVE" corresponds to the NVE ensemble.

```
<dt>
0.025d0
```

This keyword sets the step size in femtoseconds. In the present case, the step size is 0.025 femtoseconds. Note that in centroid molecular dynamics, the step size should be smaller than usual to deal with the fast motion of non-centroid modes. It is recommended to set `<dt>` to be the original step size divided by the factor `<igamma>`.

```
<nstep>
1000
```

This keyword sets the number of steps. In the present case, it is set to 1000 steps.

```
<temperature>
300.d0
```

This keyword sets the temperature in Kelvin. In the present case, it is set to 300 K.

```
<corrections>
2  2
```

This keyword sets the translational and rotational corrections. In the present case, both translation and rotational corrections are applied every step.

```
<iprint_dip>
100
```

This keyword sets the print interval of the dipole moment. In the present case, it is set to every 100 steps.

```
<iprint_rdf>
100
```

This keyword sets the print interval of the atomic pair distribution. In the present case, it is set to every 100 steps.

```
<params_rdf>
1.d0  5.d0  0.01d0
```

This keyword is valid when <iprint_rdf> is activated. It sets the mesh parameters for the atomic pair distributions. In the present case, the mesh points range from 1.0 bohr to 5.0 bohr with an increment of 0.01 bohr.

```
<bath_type>
MNHC
```

This keyword sets the type of thermostats. When MNHC is chosen, massive Nosé–Hoover thermostat chains (MNHC) are attached to each degree of freedom. In the case of CMD, the MNHC thermostats are attached only to the non-centroid modes, and they are not attached to the centroids.

```
<nnhc>
4
```

This keyword sets the chain length of the MNHC thermostats. In the present case, it is set to four.

```
<ncolor>
1
```

This keyword sets the multiplicity of the MNHC thermostats. In the present case, it is set to one.

```
<time_bath>
10.d0
```

This keyword controls the mass of the thermostats. The mass of the thermostats is related to the characteristic time scale of the system, in femtoseconds, which is designated by this keyword. Here it is set to 10 femtoseconds.

```
<nys>
5
```

This keyword controls the step size for the MNHC thermostats. In the present case, thermostats are updated 5 times.

```
<nbead>
32
```

This keyword sets the number of beads. In the present case, 32 beads are used.

```
<nref>
1
```

This keyword controls the step size for updating the harmonic forces in path integrals. In the present case, “1” means the harmonic forces are updated once per step.

```
<beadspread>
500.d0
```

This keyword controls the bead spread for the initial configuration.

```
<igamma>
10
```

This keyword controls the adiabaticity in centroid molecular dynamics. The accuracy of CMD will increase as the value of <igamma> becomes larger. Note that the adiabaticity parameter, γ, is given by the inverse of the

square of <igamma>; in the present case, it is 0.01.

If the restart file geometry.ini from a previous PIMD run exists, the atomic positions and velocities are read from this file (recommended in CMD). If the restart file bath.ini from a previous PIMD run also exists, the bath positions and velocities are read from this file (recommended in CMD). Otherwise, the centroid positions are read from structure.dat, and all bead positions are generated randomly around the centroid positions with a standard deviation controlled by the keyword <beadspread>. The velocities are generated at this temperature as an initial run (not recommended in CMD). The temperature is not controlled with respect to the centroid coordinates in the CMD method. Temperature control is applied to the non-centroid modes only.

The output files *.out and the restart files *.ini will be explained in Section 12.2 and Section 12.3, respectively.

4.10.5 Thermostatted ring polymer molecular dynamics (TRPMD)

The following keywords are important:

```
<method>
TRPMD
```

This keyword sets the simulation method. In the present case, "TRPMD" corresponds to thermostatted ring polymer molecular dynamics. The non-centroid ring-polymer normal modes are thermostatted with friction $\gamma_k = 2\lambda\widetilde{\omega}_k$, where λ is set by <scale_trpmd>. The PIMD 2.8.0 default is $\lambda = 0.5$, matching the near-optimal value used by Rossi, Ceriotti, and Manolopoulos [34]; $\lambda = 1$ is critical damping and $\lambda = 0$ recovers RPMD.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble. In the present case, "NVT" corresponds to the NVT ensemble.

4.10.6 Path integral hybrid Monte Carlo (PIHMC)

As an example, a set of input files for a water molecule is available. So, let us copy it to the directory *run*:

```
> cp -r examples/02_path_integral_dynamics/H2O/pihmc_nvt_water run
```

and enter that directory:

```
> cd run/pihmc_nvt_water
```

Let us look at the keywords one by one.

```
<method>
PIHMC
```

This keyword sets the simulation method, in the present case, path integral hybrid Monte Carlo.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble, in the present case, the NVT ensemble.

```
<iorder_hmc>
2
```

This keyword sets the order of the Suzuki–Trotter expansion, in the present case, the second order.

```
<istep_hmc>
1
```

This keyword sets the number of MD updates per MC step in the hybrid Monte Carlo method.

```
<ipotential>
WATER
```

This keyword sets the potential, in the present case, the built-in WATER potential; q-TIP4P/F is selected by default. SPC/F and RWK remain available through <model_water>.

```
<iboundary>
0
```

This keyword sets the boundary condition. In the present case, “0” means the free boundary condition.

```
<nbead>
32
```

This keyword sets the number of beads to 32.

```
<nstep>
1000
```

This keyword sets the number of steps of PIHMC to 1000.

```
<dt>
1.5d0
```

This keyword sets the step size of PIHMC to 1.5 femtoseconds.

```
<temperature>
300.d0
```

This keyword sets the temperature to 300 K.

```
<beadspread>
500.d0
```

This keyword sets the initial bead spread to 500 K.

```
<iread_exit>
100
```

This keyword sets the interval for reading `exit.dat` to every 100 steps.

```
<iprint_std>
100

<iprint_bond>
100

<iprint_eavg>
100
```

```
<iprint_rest>
100

<iprint_rdf>
100

<iprint_trj>
100

<iprint_xyz>
100

<iprint_dip>
100
```

These keywords set the intervals for printing the standard output, average bond distances, energy averages, restart files, radial distributions, trajectories in regular and xyz formats, and dipole moments, respectively.

The output files `*.out` and restart files `*.ini` will be explained in Section 12.2 and Section 12.3, respectively.

4.10.7 Centroid Intrinsic Reaction Coordinate (CIRC)

The CIRC calculations can be carried out using the code **polymers.x**. In addition to the PIMD setups mentioned in Subsection 4.10.7, the following keywords are also important.

```
<npoly>
16
```

This keyword sets the number of centroid images. In the present case, there are 16 images.

```
<np_poly>
16
```

This keyword sets the parallel number with respect to centroid images. In the present case, all images are computed in parallel with a parallel number of 16.

```
<ends_poly>
FIXED
```

This keyword sets the option for the end centroid images. In this case, the end centroid images are fixed.

```
<dt_poly>
0.04d0
```

This keyword sets the mass-weighted evolution step of the centroid images. Its dimension is $\sqrt{\text{mass}}\,\text{length}$, equivalently $\sqrt{\text{energy}}\,\text{time}$; it is not an ordinary MD time step.

```
<ncycle_poly>
1
```

This keyword sets the number of update cycles of centroid images by the string method. In this case, there is only one cycle.

```
<iprint_poly>
10
```

This keyword sets the print interval of the centroid mean force during the PIMD runs. In this case, the centroid mean force is printed every 10 PIMD steps.

```
<guess_poly>
LINE
```

This keyword sets the initial guess for the centroid images. In this case, the initial guess is a linear interpolation of the two ends read from the file `structure.dat`.

```
<projcmf_poly>
OFF
```

This keyword sets the projection of mean forces applied to centroid images. In this case, no projection is applied.

```
<ngrid_poly>
10
```

This keyword sets the number of grid points used for interpolation in the string method. In this case, 10 grid points are used.

4.11 Self-learning hybrid Monte Carlo (SLHMC)

Self-learning hybrid Monte Carlo combines a high-level reference potential with a lower-cost surrogate potential. The surrogate generates molecular- dynamics proposals, while the reference potential supplies the target energy used in the hybrid-Monte-Carlo acceptance step. The surrogate can be updated from newly evaluated reference structures during the simulation.

IMPORTANT: In PIMD 2.8.0, SLHMC is a user-facing workflow alias rather than a separate engine method. At engine level it uses `<method> = PIHMC` together with `<ipotential> = DUAL`; the two models are selected by `<dual_potential>`.

A representative current example is `examples/07_self_learning_ml/SiO2/slhmc_nvt_vasp6_mace`. A QE+AENET variant is provided in `examples/07_self_learning_ml/SiO2/slhmc_nvt_qe_aenet`. These workflows require the corresponding external reference backend and machine-learning model/training environment.

With `pimd-prep` 0.9.7, a typical setup starts with

```
pimd-prep init SLHMC VASP6+MACE
```

The essential legacy input generated from the TOML configuration is:

```
<method>
PIHMC

<ensemble>
NVT

<ipotential>
DUAL

<dual_potential>
VASP6  MACE

<nbead>
1
```

Here the first entry in `<dual_potential>` is the high-level reference model and the second is the lower-cost proposal model. Other supported combinations are represented by the examples under `examples/07_self_learning_ml`.

IMPORTANT: SLHMC uses unthermostatted HMC proposal dynamics. `pimd-prep` therefore does not ask for or write a `[thermostat]` section for SLHMC. The canonical `<bath_type> = NONE` from `input_default.dat` remains active. The SLHMC default is `<nbead> = 1`. This is a default, not a restriction: specify a larger bead count explicitly for path-integral SLHMC. These rules are specific to the SLHMC workflow and do not alter ordinary PIHMC setup choices.

For SLHMC, the same ensemble-scoping rule applies to the cell degrees of freedom. A canonical fixed-cell calculation is simply

```
[dynamics]
ensemble = "NVT"

[path_integral]
beads = 1
```

and contains neither `[thermostat]` nor `[barostat]`. The former is absent because SLHMC proposal trajectories are unthermostatted; the latter is absent because NVT has fixed volume.

For SLHMC/NPT, pressure and barostat controls are active, for example

```
[dynamics]
ensemble = "NPT"
pressure = "1 bar"

[barostat]
cell_mode = "CUBIC2"
time      = "1000 fs"
```

For SLHMC/NTT, scalar pressure is not used; specify the tension tensor and use `cell_mode = "PPHEX"`. The generated engine input remains PIHMC and uses **<npt_type>** or **<ntt_type>** as appropriate.

For heterogeneous HIGH+LOW SLHMC, the backend input/model files remain in the SLHMC run directory. The MPI DUAL routine switches **<ipotential>** internally and calls the two force backends directly. The historical **<dual_dat_dir>** mechanism belongs to the older same-potential DUAL route and should not be introduced merely to separate two different SLHMC backends.

When on-the-fly training is enabled for a LOW AENET, N2P2, or MACE surrogate, the initial trained model itself may be absent. In that case **pimd-prep** enables an internal bootstrap path automatically; there is no separate user-facing bootstrap keyword. Until the LOW model files exist, the proposal force is evaluated with the HIGH backend. The ordinary SLHMC analysis path still writes the resulting reference structures and calls the existing backend trainer at its configured training interval after the minimum number of structures has accumulated. Once the model products appear, the next outer SLHMC step automatically resumes normal LOW-model proposal dynamics. The ready state is then latched: because training only adds or replaces model products, the steady-state LOW force loop does not repeatedly query their file existence.

A bootstrap configuration must therefore have positive structure-output, training-interval, and minimum-structure controls. For MACE the minimum must be at least three structures, consistent with the current trainer. Prediction- only calculations do not use this fallback: a missing LOW model remains an input error.

Rejected trial structures can be screened before they enter an on-the-fly training set by setting, for example,

```
<minweight_train_dual>
1.0d-6
```

The filter uses the pure HIGH-level HMC weight $\exp[-\beta\Delta H_{\mathrm{high}}]$, not the mixed DUAL acceptance Hamiltonian. It applies to rejected-inclusive XSF output (**<ioption_xsf_aenet>**, **<ioption_xsf_n2p2>**, or **<ioption_xsf_mace>** equal to 1); accepted-only output already omits rejected HMC trials. The comparison is performed as $\beta\Delta H_{\mathrm{high}} \leq -\log(w_{\mathrm{min}})$. The default value is -1.0, which disables the filter. An active cutoff must satisfy $0 < w_{\mathrm{min}} \leq 1$. The same definition is used for PIHMC NVT, NPT, and NTT and for the DUAL REHMC T/TX paths; REHMC uses the inverse temperature of each replica.

The hybrid-Monte-Carlo proposal length and adaptation are controlled, for example, by

```
<istep_adjust_hmc>
30

<istep_max_hmc>
512

<istep_hmc>
4
```

For the MACE self-learning workflow, the distributed example additionally uses training/output controls such as

```
<istep_train_mace>
30

<lstep_train_mace>
9999999

<minxsf_train_mace>
10

<iprint_xsf_mace>
1

<dir_train_mace>
.
```

For an AENET surrogate, keep the top-level [aenet] table minimal: it contains only number_of_species. Put each model in [aenet.species.N]. Prediction requires only name and network; SLHMC training additionally uses fingerprint, reference_energy, and hidden_layers. Put the common self-learning controls in [aenet.training]. pimd-prep then generates the AENET TYPES, SETUPS, and NETWORKS blocks automatically. Do not repeat AENET network or fingerprint files in [files] and do not use numbered keys such as network_2 or fingerprint_setup_2.

The precise training controls depend on the surrogate backend. Users should therefore start from the closest directory in examples/07_self_learning_ml rather than translating an SLHMC setup from another machine-learning interface by keyword substitution alone. The pimd.toml file in each directory is authoritative and should be regenerated with pimd-prep before execution.

5 Quantum transition-state free energy (QTST with polymers.x)

The QTST workflow highlighted in PIMD 2.8.0 evaluates a centroid free-energy profile with the executable polymers.x. A set of centroid images is prepared along a reaction coordinate or path, and an independent path-integral simulation is sampled around each centroid image. The mean centroid forces are then assembled and integrated to obtain the potential of mean force (PMF). This centroid free-energy profile is the principal input needed for centroid-based quantum transition-state analysis.

This workflow is distinct from the lower-level <ensemble> = QTST option of pimd.mpi.x. The latter uses <iatom_qtst> to constrain the centroid of one selected atom along the z direction. In this section, "QTST" refers to the multi-image polymers.x free-energy workflow.

A representative distributed example is examples/05_reaction_paths_rates/FeH/qtst_mtp. Its authoritative user input is pimd.toml; the corresponding legacy input.dat is generated by pimd-prep. The example is run with

```
cd examples/05_reaction_paths_rates/FeH/qtst_mtp
./run.sh
```

where run.sh selects polymers.x.

The central settings in this example correspond to

```
<iboundary>
EXTXYZ
```

```
<ipotential>
MTP

<nbead>
48

<npoly>
23

<ncycle_poly>
10

<guess_poly>
SPLINE

<ends_poly>
FREE
```

Here, <nbead> is the number of imaginary-time beads used for each centroid image, <npoly> is the number of centroid images, <ncycle_poly> is the number of update cycles for the centroid path, <guess_poly> selects the initial interpolation of the images, and <ends_poly> controls the treatment of the end images. Additional polymer-path controls such as <dt_poly>, <ngrid_poly>, <projcmf_poly>, and <iprint_poly> are read from input.dat or, when omitted, from input_default.dat.

For each centroid image, polymers.x performs path-integral sampling, accumulates the centroid mean force and energy estimators, and then combines all images. The principal outputs are

- poly.out: free-energy components for each centroid image and update cycle;
- poly_final.out: reaction-coordinate grid and integrated centroid PMF;
- poly.one.xyz: the current centroid images as separate XYZ frames;
- poly.all.xyz: the centroid path history in a combined XYZ trajectory;
- poly.N/: per-image PIMD restart and standard-output data.

The Fe–H example uses an MTP force engine, but polymers.x shares many of the MPI force backends used by pimd.mpi.x. The executable must therefore be built with the backend required by the chosen example.

5.1 String method (STRING)

The theoretical background of the string method (STRING) is briefly described in Section 13.7.

As an example, a set of input files for the water dimer is provided. Copy the example files to the directory *run* as follows:

```
> cp -r examples/05_reaction_paths_rates/H4O2/string_cff run
```

and move to that directory:

```
> cd run/string_cff
```

Let us examine the keywords in input.dat one by one.

```
<method>
STRING
```

This keyword sets the simulation method to the string method.

```
<ipotential>
CFF
```

This keyword specifies the potential model. In this example, the LAMMPS-compatible CFF force field is employed.

```
<iboundary>
0
```

This keyword sets the boundary condition. In the present case, "0" corresponds to the free boundary condition.

```
<nbead>
32
```

This keyword sets the number of images in the string, which is equal to the number of beads. In this example, 32 images are used.

```
<nstep>
1000
```

This keyword sets the maximum number of string updates. Here, the string is evolved for up to 1000 steps.

```
<dt>
0.01d0
```

This keyword sets the step size for updating the string. In this case, the step size is 0.01 hartree$^{1/2}$ femtoseconds.

```
<ends_string>
FIXED
```

This keyword specifies the boundary condition for the end points of the string. Here, both end points are fixed during the string optimization.

```
<iprint_std>
100
```

This keyword sets the print interval of the standard output. In the present case, output is written every 100 steps.

The output files (`*.out`) and restart files (`*.ini`) are explained in Sections 12.2 and 12.3, respectively.

When starting a calculation from scratch, the initial string can be generated automatically as a straight-line interpolation between the reactant and product structures provided in the file `structure.dat`. Alternatively, the user may prepare a restart file `string.ini`, which explicitly contains the atomic coordinates of all images (e.g., 32 images in this example). If both `string.ini` and `structure.dat` are present in the execution directory, the restart file `string.ini` always takes priority.

5.2 Metadynamics (MTD)

The theoretical background of metadynamics (MTD) is briefly explained in Section 13.13.

In PIMD, MTD in the NVT ensemble based on massive Nosé-Hoover thermostats is implemented.

So far, eight types of collective variables have been implemented, and the set of collective variables should be chosen from the following list:

- type 1: bond distance (between two atoms)
- type 2: bond angle (among three atoms)
- type 3: dihedral angle (among four atoms)
- type 4: bond difference (among three atoms)
- type 5: coordination number (for atomic species)
- type 6: difference in coordination numbers (for atomic species)
- type 7: Cartesian coordinate of the geometric center (unweighted coordinate average of an atomic kind/group)
- type 8: difference between geometric-center Cartesian coordinates (two atomic kinds/groups)

It is possible to choose one, two, or three collective variables, corresponding to one-, two-, or three-dimensional metadynamics, respectively. It is also possible to employ two or three collective variables of the same type, such as two bond distances.

In the PIMD implementation, the Gaussian width is set for each type of collective variable, rather than for each individual collective variable. For example, if two bond distances are specified, the same Gaussian width value is shared by both.

As an example, a compact two-dimensional MTD calculation of a water molecule is provided. It exercises multiple collective variables while using the same keyword machinery described below. Copy the input files to the directory *run* by executing:

```
> cp -r examples/04_enhanced_sampling/H2O/metadynamics_2d_water run
```

Then enter the directory:

```
> cd run/metadynamics_2d_water
```

For new calculations, the recommended input is `pimd.toml`; **`pimd-prep`** converts it to the legacy `input.dat` read by the Fortran engine. For example, the essential metadynamics part of the water example is

```
[metadynamics]
walkers = 1  # Number of metadynamics walkers.
collective_variables = 2

cv1_type = "ANGL"
cv1_atoms = [2, 1, 3]  # angle 2-1-3

cv2_type = "DIFF"
cv2_bond1 = [2, 1]
cv2_bond2 = [1, 3]  # r(2,1) - r(1,3)

hill_height = "100 K"
hill_widths_ANGL = "5 degree"
hill_widths_DIFF = "0.0529177210903 angstrom"

reconstruction_grid_ANGL = ["0 degree", "180 degree", "2 degree"]
reconstruction_grid_DIFF = ["-1.5875 angstrom", "1.5875 angstrom", "0.1058 angstrom"]
```

Only CV types actually used by `cvN_type` need a `hill_widths_TYPE` or `reconstruction_grid_TYPE` entry. **`pimd-prep`** expands these active-type settings to the complete eight-row legacy blocks required by the current Fortran reader. This avoids relying on the ordering of DIST/ANGL/DIH/DIFF/CN/DCN/XYZ/DXYZ rows.

The preferred structured CV forms are:

```
# distance, angle, and dihedral
cv1_type = "DIST"; cv1_atoms = [1, 2]
```

```
cv1_type = "ANGL"; cv1_atoms = [1, 2, 3]
cv1_type = "DIH";  cv1_atoms = [1, 2, 3, 4]

# bond-distance difference
cv1_type = "DIFF"
cv1_bond1 = [1, 9]
cv1_bond2 = [9, 2]

# coordination number
cv1_type = "COORDINATION"
cv1_groups = [1, 2]
cv1_exponents = [6, 12]  # [nu, mu]
cv1_reference_distance = "1.5 angstrom"

# Cartesian coordinate of an unweighted geometric center
cv1_type = "CARTESIAN"
cv1_axis = "Z"
cv1_group = 2
```

For `COORDINATION`, the switching function is proportional to $(1-(r/d)^{\nu})/(1-(r/d)^{\mu})$ and the TOML exponent order is therefore `[nu, mu]`. `COORDINATION_DIFFERENCE` defines two complete coordination numbers and subtracts the second from the first. `CARTESIAN_DIFFERENCE` similarly uses `group1 - group2` along the selected axis. The Cartesian group coordinate is a simple average of unfolded atomic coordinates; it is not mass weighted.

For multiple-walker metadynamics, `walkers` maps to the historical **<nbead>**; `initial_walker_spread_temperature` maps to **<beadspread>**. The hill-deposition trigger is written as `deposition_mode = "displacement_and_time"` or `"fixed_interval"` instead of the legacy integer switch. The legacy **<ioption_meta>** walker-separation switch is retained for compatibility, but in the current source its separation flag is computed while the final hill-suppression test is commented out. New TOML input should not rely on that switch having an effect.

The remainder of this section documents the generated legacy keywords and is also useful when maintaining an existing hand-written `input.dat`.

To prepare the file `input.dat`, all keywords described in Sections 4.1 and 4.2 are required. In addition, the following keywords are important:

```
<method>
MTD
```

This keyword sets the simulation method, which in this case is metadynamics.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble, here the NVT ensemble.

```
<nstep>
10000
```

This keyword sets the total number of steps, in this case 10,000.

```
<dt>
0.25d0
```

This keyword sets the time step in femtoseconds, here 0.25 fs.

```
<temperature>
300.d0
```

This keyword sets the system temperature in Kelvin, here 300 K.

```
<bath_type>
MNHC
```

This keyword sets the type of thermostats. In metadynamics, only MNHC thermostats are implemented.

```
<ncolor>
1
```

This keyword sets the multiplicity of the MNHC thermostats, in this case one.

```
<nnhc>
4
```

This keyword sets the chain length of the MNHC thermostats, here four.

```
<time_bath>
10.d0
```

This keyword controls the mass of the thermostats attached to the system. The thermostat mass is related to the characteristic time scale of the system, in femtoseconds, specified by this keyword. In the present case, a time scale of 10 fs is chosen.

```
<nys>
5
```

This keyword controls the time integration of the MNHC thermostats. In the present case, the thermostats are updated five times per <dt>.

```
<nbead>
1
```

This keyword sets the number of walkers, i.e., the number of beads. In this case, a single walker is used.

```
<beadspread>
500.d0
```

This keyword sets the temperature controlling the initial bead spread of the walkers. In the present case, 500 K is chosen.

The keywords specific to metadynamics are listed below.

```
<mg_meta>
50000
```

This keyword sets the maximum number of history-dependent Gaussian hills, here 50,000.

```
<nmeta>
1
1,   2    3
```

This keyword specifies the parameters of the collective variables. The first line gives the dimensionality of the collective variables, which is one in this case. The following line defines the collective variable(s); here, a one-dimensional collective variable corresponding to the bond distance (type 1) between atoms #2 and #3.

```
<nref_meta>
10
```

This keyword controls the update frequency of the forces arising from the harmonic potential and Gaussian hills. In this case, the forces are updated ten times within one <dt>.

```
<gh_meta>
150.d0
```

This keyword sets the height of the Gaussian hills. In the present case, a height of 150 K is chosen.

```
<gw_meta>
1    0.100d0
2    5.000d0
3   10.000d0
4    0.100d0
5    0.100d0
6    0.100d0
7    0.100d0
8    0.100d0
```

This keyword defines the Gaussian widths for each type of collective variable. In the present case, only the first line is relevant, which specifies a Gaussian width of 0.1 bohr for the bond distance (type 1).

```
<time_cv_meta>
75.0
```

This keyword sets the diffusion time scale of the collective variables, which determines the mass of the fictitious particle. Here, a time scale of 75 fs is chosen.

```
<time_fc_meta>
30.d0
```

This keyword sets the oscillation period of the harmonic potential between the fictitious particle and the collective variables. In this case, 30 fs is chosen.

```
<time_limit_meta>
225.0
```

This keyword sets the minimum time interval for adding a new Gaussian hill. In the present case, 225 fs is chosen, which corresponds to three times the value of <time_cv_meta>.

```
<time_cv_bath>
1000.0
```

This keyword sets the time scale of the thermostat attached to the fictitious particle, which controls the thermostat mass. Here, it is set to 1000 fs.

```
<iprint_cv_meta>
1
```

This keyword sets the print interval for the collective variables and the fictitious particle to the output file `cv.out`. In this case, they are printed at every step.

```
<iprint_meta>
1
```

This keyword sets the print interval of the energy components in metadynamics to the output file `meta.out`. In this case, the data are printed at every step.

```
<iprint_rec_meta>
100
```

This keyword sets the print interval of the reconstructed free energy to the output file `rec.out`. In the present case, it is printed every 100 steps.

```
<params_rec_meta>
1   1.0    6.0   0.02
2   0.0  180.0   1.00
3   0.0  360.0   2.00
4  -3.0    3.0   0.02
5   0.0    4.0   0.02
6   0.0    4.0   0.02
7   1.0    6.0   0.02
8   1.0    6.0   0.02
```

This keyword defines the mesh points used to reconstruct the free energy landscape for each type of collective variable. In the present case, only the first line is relevant, specifying a range from 1.0 bohr to 6.0 bohr with an increment of 0.02 bohr.

```
<cut_rec_3d>
100.0
```

This keyword controls the numerical accuracy of the reconstruction of the free energy surface in three-dimensional metadynamics. In the present case, this option is inactive because the metadynamics is one-dimensional.

```
<joption_meta>
0
```

This keyword controls the timing of adding new Gaussian hills. Legacy value 0 requires both sufficient displacement and the minimum elapsed time; value 1 uses the elapsed-time condition alone (fixed-interval mode). In the present case, both the displacement and time conditions must be satisfied.

To perform well-tempered metadynamics, one must additionally specify:

```
<dtemp_meta>
1000.0
```

This keyword sets the temperature parameter of well-tempered metadynamics, here 1000 K. If this keyword is not specified, `<dtemp_meta>` = `0.0` corresponds to conventional metadynamics.

The output files `*.out` and the restart files `*.ini` are described in Sections 12.2 and 12.3, respectively.

5.3 Replica exchange hybrid Monte Carlo (REHMC)

In the PIMD code, the following replica exchange hybrid Monte Carlo (REHMC) methods are implemented:

- Temperature replica exchange (TX) between systems with different temperatures.
- Hamiltonian replica exchange (HX) between systems with different alchemical intermediates.

The theoretical background of these methods is briefly explained in Section 13.12.

5.3.1 Temperature replica exchange hybrid Monte Carlo

For example, a temperature-exchange REHMC input for alanine dipeptide is available. To use it, copy the files to the directory *run* by executing the following command:

```
> cp -r examples/04_enhanced_sampling/dialanine/rehmc_cff run
```

Then enter that directory:

```
> cd run/rehmc_cff
```

Next, let's examine the keywords one by one:

```
<method>
REHMC
```

This keyword sets the simulation method to replica exchange hybrid Monte Carlo.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble to NVT.

```
<irem_type>
TX
```

This keyword sets the type of replica exchange method to temperature exchange.

```
<istep_hmc>
1
```

This keyword sets the number of MD updates per MC step in the hybrid Monte Carlo method.

```
<ipotential>
WATER
```

This keyword sets the atomic species of the system. In this case, the system is a water molecule.

```
<iboundary>
0
```

This keyword sets the boundary condition. Here, "0" means a free boundary condition.

```
<nstep>
100000
```

This keyword sets the number of steps to 100000.

```
<dt>
```

```
2.d0
```

This keyword sets the step size to 2 femtoseconds.

```
<nbead>
4
```

This keyword sets the number of replicas to four.

```
<temprange_rem>
300.d0 2400.d0
```

This keyword sets the temperature range by specifying the lowest temperature T_1 and the highest temperature T_n among the replicas. In this case, the temperature range is set between 300 K and 2400 K, with the first, second, third, and fourth replicas set to 300 K, 600 K, 1200 K, and 2400 K, respectively.

```
<beadspread>
500.d0
```

This keyword sets the initial bead spread to 500 K.

```
<iread_exit>
1000
```

This keyword specifies the read interval for the file `exit.dat`.

```
<iprint_std>
1000

<iprint_rest>
1000

<iprint_rdf>
1000

<iprint_trj>
1000

<iprint_xyz>
1000
```

These keywords specify the print intervals for the standard output, the restart files, the radial distribution functions, and the trajectories in the regular and xyz formats, respectively.

The output files `*.out` and the restart files `*.ini` will be explained in Section 12.2 and Section 12.3, respectively.

5.3.2 Hamiltonian replica exchange hybrid Monte Carlo

As an example, a set of input files for an $M^+(H_2O)$ cluster (M = K and Cs) is available. To use these input files, copy them to the directory *run*:

```
> cp -r examples/04_enhanced_sampling/CsH2O/rehmc_alchemical_cff run
```

and navigate to that directory:

```
> cd run/rehmc_alchemical_cff
```

Now, let's examine the keywords one by one:

```
<method>
REHMC
```

This keyword sets the simulation method, which in this case is replica exchange hybrid Monte Carlo.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble, which in this case is NVT.

```
<irem_type>
HX
```

This keyword sets the type of replica exchange method, which in this case is Hamiltonian exchange.

```
<istep_hmc>
1
```

This keyword sets the number of molecular dynamics updates per Monte Carlo step in the hybrid Monte Carlo method.

```
<ipotential>
ALCHEM
```

This keyword specifies the use of potentials of alchemical intermediates.

```
<iboundary>
0
```

This keyword sets the boundary condition, where "0" indicates a free boundary condition.

```
<nstep>
100000
```

This keyword sets the number of steps to 100000.

```
<dt>
2.d0
```

This keyword sets the step size to 2 femtoseconds.

```
<nbead>
4
```

This keyword sets the number of replicas with different alchemical intermediates, which in this case is four.

```
<alchem_dat_dir>
dat_1  dat_2
```

This keyword sets the input directories for the alchemical potentials.

```
<alchem_scr_dir>
scr_1  scr_2
```

This keyword sets the temporary scratch directories for the alchemical potentials.

```
<beadspread>
500.d0
```

This keyword sets the initial bead spread to 500 K.

```
<iread_exit>
1000
```

This keyword specifies the read interval for the file `exit.dat`.

```
<iprint_std>
1000

<iprint_rest>
1000

<iprint_rdf>
1000

<iprint_trj>
1000

<iprint_xyz>
1000
```

These keywords specify the print intervals for the standard output, restart files, radial distribution functions, and trajectories in the regular and xyz formats, respectively.

The output files `*.out` and the restart files `*.ini` will be explained in Section 12.2 and Section 12.3, respectively.

5.4 Mean force dynamics (AFED)

The theoretical background of mean force dynamics is briefly explained in Section 13.16. Throughout this manual, “adiabatic free energy dynamics (AFED)" is used to indicate “mean force dynamics". In PIMD, the following AFED-related techniques are implemented.

- GRAD: calculation of the free energy gradient with a constant constraint.
- DESCENT: descent search for free energy minimum points using the descent method.
- ASCENT: ascent search for free energy saddle points using ascent dynamics.
- AUTO: automated search for free energy stationary points by combining DESCENT and ASCENT.
- TAMD: temperature-accelerated molecular dynamics. The options are NVE, NVT, and velocity scaling.
- LOGMFD: logarithmic mean force dynamics. The options are NVE, NVT, and velocity scaling.

Eight types of collective variables have been implemented (so far). The set of collective variables should be chosen from the list below.

- type 1: bond distance (between two atoms)
- type 2: bond angle (among three atoms)

- type 3: dihedral angle (among four atoms)
- type 4: bond difference (among three atoms)
- type 5: coordination number (for atomic species)
- type 6: difference in coordination numbers (for atomic species)
- type 7: center of mass (of atomic species)
- type 8: difference in centers of mass (of two atomic species)

5.4.1 Common keywords used in all the AFED methods (Free energy gradient)

As an example, a set of input files for one-dimensional AFED of a butene molecule is available. In this case, the collective variable is chosen to be the C-C-C-C bond dihedral. Let us copy it to the directory *run*:

```
> cp -r examples/04_enhanced_sampling/C4H10/afed_grad_cff run
```

and enter that directory:

```
> cd run/afed_grad_cff
```

To prepare the file `input.dat`, all the keywords given in Sections 4.1 and 4.2 are important. In addition, the following keywords, which can be categorized into three groups, are also important.

First group: Keywords for molecular dynamics. This keyword sets the number of atoms in the system. In the present case, there are 14 atoms in the C_4H_{10} molecule.

```
<ipotential>
CFF
```

This keyword sets the potential model. In the present case, "CFF" selects the LAMMPS-compatible classical force-field interface.

```
<iprint_std>
100
```

This keyword sets the print interval of the standard output `standard.out`. In the present case, every 100 steps.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble for molecular dynamics. In the PIMD code, the "NVT" ensemble must always be chosen for AFED (the NPT ensemble for AFED is currently not available).

```
<bath_type>
MNHC
```

This keyword sets the type of thermostat. In the PIMD code, the massive Nosé-Hoover chain thermostat, "MNHC", must always be chosen for AFED. The thermostat settings can be controlled by the keywords `<time_bath>`, `<nnhc>`, `<ncolor>`.

```
<temperature>
500.d0
```

This keyword sets the system temperature in kelvin; in the present case, "500" K.

```
<dt>
0.25d0
```

This keyword sets the molecular dynamics time step in femtoseconds; in the present case, 0.25 fs.

```
<nys>
8
```

This keyword controls the step size for the MNHC thermostat. In the present case, the thermostat is updated "8" times per step.

Second group: Keywords for constraints.

```
<nref>
8
```

This keyword controls the step size for the harmonic constraint. In the present case, the harmonic constraint is updated "8" times per step.

```
<iprint_cons>
-1
```

This keyword sets the print interval for constraint information to the file `cons.out`; in the present case, it is not printed.

```
<ncons>
1
3   1   5   8  11
```

This keyword sets the number of constraints followed by the parameters of each constraint. In the present case, there is "1" constraint of type "3" (dihedral angle) among atoms "1, 5, 8, 11".

```
<params_cons>
1    5.0d+0
2    5.0d-2
3    5.0d-3
4    2.5d+0
5    1.0d+0
6    1.0d+0
7    5.0d+0
8    5.0d+0
```

This keyword sets the force constant for each type of constraint. In the present case, only the third line is important, which sets the force constant of type "3" (dihedral angle) to be "5.0×10^{-3}" hartree/degree^2.

Third group: Keywords for adiabatic free energy dynamics.

```
<method>
AFED
```

This keyword sets the simulation method. In the present case, "AFED" means adiabatic free energy dynamics.

```
<afed_type>
GRAD
```

This keyword sets the type of AFED. In the present case, "GRAD" means free energy gradient calculation.

```
<nbead>
8
```

This keyword sets the number of beads (replicas). In the present case, “8” beads.

```
<params_afed>
1    0.08d0    3.0d0   10.0d0
2    3.00d0    0.0d0  180.0d0
3    5.00d0 -180.0d0  180.0d0
4    0.04d0   -3.0d0    3.0d0
5    0.02d0    0.0d0    4.0d0
6    0.02d0    0.0d0    4.0d0
7    0.08d0    3.0d0   10.0d0
8    0.08d0    3.0d0   10.0d0
```

This keyword sets the AFED parameters for all types of collective variables. In the present case, the active ones are the first four numbers in the third line. For the collective variable of type “3” (dihedral angle), the reference shift $\Delta\xi_d^{\mathrm{ref}}$ is “5.0” degrees, the lower bound ξ_d^{min} is “-180” degrees, and the upper bound ξ_d^{max} is “180” degrees. All the remaining values are inactive.

```
<nstep_pre_afed>
1000
```

This keyword sets the number of preliminary MD steps to equilibrate the system. In the present case, “1000” steps.

```
<nstep_pro_afed>
2000
```

This keyword sets the number of productive MD steps to collect averages of the free energy gradient. In the present case, “2000” steps.

To start the calculation, type

```
> pimd.x
```

Then the following message will appear on the screen:

```
    ....      ..........      ..........    ......  ........................
    ....      ..........      ..........    ......  ........................
    1700      1.32412595      0.03728593    485.62  2017-02-23 THU 12:06:50.47
    1800      1.32412021      0.03356013    469.85  2017-02-23 THU 12:06:50.92
    1900      1.32411461      0.03331202    527.77  2017-02-23 THU 12:06:51.49
    2000      1.32412219      0.03664502    416.04  2017-02-23 THU 12:06:52.05

================================================================================
 step st cv  r-ideal   r-mean   f-energy     -df/dr     dr/dt     dt     n  xi
--------------------------------------------------------------------------------
    0 GR  1  179.010  179.016   0.000000   0.000094   0.00000  0.500  1.00   0

Restart files: geometry.ini, step.ini, bath.ini, cv.ini, afed.ini.

Normal termination of pimd.
```

The last line provides the following information:

- `step`: AFED step number (iteration number).
- `st`: AFED status.
  - `GR`: free energy gradient.
  - `TE`: convergence test.
  - `AS`: ascent trajectory.
  - `DE`: descent trajectory.
  - `D1`: first descent trajectory from a saddle point.
  - `D2`: second descent trajectory from a saddle point.
  - `EQ`: minimum point.
  - `TS`: saddle point.
  - `TA`: temperature-accelerated molecular dynamics.
  - `LO`: logarithmic mean force dynamics.
- `r-ideal`: position of the fictitious particle.
- `r-mean`: mean position of the collective variables.
- `f-energy`: free energy (potential of mean force).
- `-df/dr`: mean force (negative of the free energy gradient).
- `dr/dt`: velocity of the collective variables.
- `dt`: AFED step size.
- `n`: unit vector.
  - For ascent trajectories, vector $\mathbf{n}$.
  - For descent trajectories, vector parallel to the mean force.
- `xi`: angle θ between the force vectors of the last two AFED steps, used to judge convergence.

To test the convergence of mean forces, one can use

```
<afed_type>
TEST
```

In this case, the run is controlled by the following keywords:

```
<nstep_pro_afed>
20000
```

This keyword sets the length of the production MD run.

```
<iprint_test_afed>
1000
```

This keyword sets the print interval of the mean forces during the production MD run.

The output files `*.out`, `*.xyz`, and the restart files `*.ini` will be explained in Section 12.2 and Section 12.3, respectively.

5.4.2 Descent search for free energy minima

To prepare the file `input.dat`, all the keywords given in Sections 4.1, 4.2, and 5.4.1 are important. In addition, the following keywords are also important:

```
<method>
AFED
```

This keyword sets the simulation method. In the present case, “AFED” means adiabatic free energy dynamics.

```
<afed_type>
DESCENT
```

This keyword sets the type of AFED. In the present case, “DESCENT” means descent search.

```
<niter_afed>
200
```

This keyword sets the number of iterations (updates of the fictitious particle) in adiabatic free energy dynamics. In the present case, “200” iterations.

```
<dt_descent_afed>
0.5d0
```

This keyword sets the initial AFED step size for the descent search. In the present case, “0.5” (dimensionless).

```
<dt_damp_afed>
0.7d0
```

This keyword sets the damping factor of the AFED step size as convergence is approached. In the present case, “0.7” (dimensionless).

```
<dt_conv_afed>
0.05d0
```

This keyword sets the AFED step size at convergence. In the present case, “0.05” (dimensionless).

5.4.3 Ascent search for free energy saddles

To prepare the file `input.dat`, all the keywords given in Sections 4.1, 4.2, and 5.4.1 are important. In addition, the following keywords are also important.

```
<method>
AFED
```

This keyword sets the simulation method. In the present case, “AFED” means adiabatic free energy dynamics.

```
<afed_type>
ASCENT
```

This keyword sets the type of AFED. In the present case, “ASCENT” means ascent search.

```
<niter_afed>
200
```

This keyword sets the number of iterations (updates of the fictitious particle) in adiabatic free energy dynamics. In the present case, “200” iterations.

```
<dt_ascent_afed>
1.0d0
```

This keyword sets the initial AFED step size for the ascent search. In the present case, “1.0” (dimensionless).

```
<dt_damp_afed>
0.7d0
```

This keyword sets the damping factor of the AFED step size as convergence is approached. In the present case, “0.7” (dimensionless).

```
<dt_conv_afed>
0.05d0
```

This keyword sets the AFED step size at convergence. In the present case, “0.05” (dimensionless).

```
<gamma_ascent_afed>
1.0d0
```

This keyword sets the parameter γ^{gad} for AFED ascent search in hartree. In the present case, “1.0” hartree.

```
<ascent_sampling_afed>
NUMERICAL
```

This keyword sets the bead-sampling method used to evaluate the gradient and Hessian terms in AFED ascent search. In the present case, “NUMERICAL” selects finite-difference sampling. “ANALYTICAL” is also available. The canonical default is “NUMERICAL”.

```
<fdiff_sampling_afed>
5.0
```

This keyword sets the finite difference parameter for the numerical evaluation of the Hessian term in AFED ascent search. In this case, “5.0”.

5.4.4 Automated search for free energy stationary points

To prepare the file `input.dat`, all the keywords given in Sections 4.1, 4.2, and 5.4.1 are important. In addition, the following keywords are also important.

```
<method>
AFED
```

This keyword sets the simulation method. In the present case, “AFED” means adiabatic free energy dynamics.

```
<afed_type>
AUTO
```

This keyword sets the type of AFED. In the present case, “AUTO” means automated search by combining descent and ascent methods.

```
<niter_afed>
200
```

This keyword sets the number of iterations (updates of the fictitious particle) in adiabatic free energy dynamics. In the present case, “200” iterations.

```
<dt_descent_afed>
0.5d0
```

This keyword sets the initial AFED step size for descent search. In the present case, “0.5” (dimensionless).

```
<dt_ascent_afed>
1.0d0
```

This keyword sets the initial AFED step size for ascent search. In the present case, “1.0” (dimensionless).

```
<dt_damp_afed>
0.7d0
```

This keyword sets the damping factor of the AFED step size as convergence is approached. In the present case, “0.7” (dimensionless).

```
<dt_conv_afed>
0.05d0
```

This keyword sets the AFED step size at convergence. In the present case, “0.05” (dimensionless).

```
<gamma_ascent_afed>
1.0d0
```

This keyword sets the gamma parameter for AFED ascent search in hartree. In the present case, “1.0” hartree.

```
<ascent_sampling_afed>
NUMERICAL
```

This keyword sets the bead-sampling method used to evaluate the gradient and Hessian terms in AFED ascent search. In the present case, “NUMERICAL” selects finite-difference sampling. “ANALYTICAL” is also available. The canonical default is “NUMERICAL”.

```
<fdiff_sampling_afed>
5.0
```

This keyword sets the finite difference parameter for the numerical evaluation of the Hessian term in AFED ascent search. In this case, “5.0”.

```
<nshot_auto_afed>
10
```

This keyword sets the maximum number of shots in AFED ascent search from a minimum in automated calculations.

```
<nmiss_auto_afed>
5
```

This keyword sets the maximum number of consecutive misses in AFED ascent search from a minimum in automated calculations.

5.4.5 Temperature accelerated molecular dynamics

To prepare the file `input.dat`, all the keywords given in Sections 4.1, 4.2, and 5.4.1 are important. In addition, the following keywords are also important.

```
<method>
AFED
```

This keyword sets the simulation method. In the present case, “AFED” means adiabatic free energy dynamics.

```
<afed_type>
TAMD
```

This keyword sets the type of AFED. In the present case, “TAMD” means temperature accelerated molecular dynamics.

```
<tamd_type>
NVE
```

This keyword sets the type of TAMD. In the present case, “NVE” means NVE-type TAMD. NVT (Nosé-Hoover dynamics) and VS (velocity scaling dynamics) are also available.

```
<niter_afed>
200
```

This keyword sets the number of AFED steps (total updates of the fictitious particle) in TAMD. In the present case, “200” iterations.

```
<dt_tamd>
1.0
```

This keyword sets the AFED step size, Δt_{cv}, for TAMD. In the present case, “1.0” (since the dynamics of TAMD is physically meaningless, the unit can be arbitrarily chosen; e.g., fs).

```
<start_afed>
MANUAL
0.2  1.3e-2
1.67  -6.3e-1
...
```

This keyword sets the initial position and velocity of each fictitious CV particle. If “MANUAL” is specified in the first line, the initial position and velocity of each CV particle should be given for each line as in the example above. In this case, the initial position of the first CV particle is 0.2 and its velocity is 1.3e-2. Those of the second CV particle are 1.67 and -6.3e-1, respectively. If “AUTOMATIC” is instead specified (default setting), the initial positions are automatically estimated from the initial configuration of the MD system, and the initial velocities are set by random numbers corresponding to the physical temperature given by the keyword <temperature>.

```
<start_therm_afed>
MANUAL
0  0
```

This keyword sets the initial position and velocity of the Nosé-Hoover thermostat variable η. If “MANUAL” is specified, its initial position and velocity should be given; in this case, $\eta(0) = 0$ and $v_\eta(0) = 0$ (default setting). If “AUTOMATIC” is instead specified, η is set to “0” and v_η is automatically set according to the preset physical temperature.

```
<mass_afed>
MANUAL
1
4
...
```

This keyword sets the parameters that re-adjust the mass of the fictitious CV particles. If “MANUAL” is set, the mass determined by <params_afed> (see Sec.5.4.1) is multiplied by the number given below “MANUAL” (the ith line is for the mass of the ith particle). In this example, the mass of the first CV particle is multiplied by “1” and that of the second CV particle is multiplied by “4”. If “AUTOMATIC” is instead specified (default setting), the masses are not re-adjusted.

```
<temperature_tamd>
3000.0
```

This keyword sets the temperature of TAMD, T_{tamd}. In the present case, “3000.0” kelvin.

```
<niter_bath_afed>
100
```

This keyword sets the number of AFED steps, n_b, associated with the mass of the Nosé-Hoover thermostat, Q_η. In the present case, “100” (no unit); i.e., $Q_\eta = Dk_BT(n_b\Delta t_{\mathrm{cv}})^2$, where D is the number of fictitious particles and T is the physical temperature given by the keyword <temperature>.

```
<ioption_afed>
1 or 2
```

This keyword sets the weight W_l in parallel dynamics discussed in Sec.13.16. With “1”, $W_l = 1$ for all beads (replicas), while W_l is set by Eq.(204) (see Sec.13.16) with “2”. In the current PIMD 2.8.0 source there is no <iprint_weight> input keyword. When the parallel-weight branch (<ioption_afed>=2) is active, the replica weights are appended to afed_weight.out at each AFED iteration after initialization. The legacy <iprint_weight> tag must not be used.

5.4.6 Logarithmic mean force dynamics

To prepare the file input.dat, all the keywords given in Sections 4.1, 4.2, and 5.4.1 are important. In addition, the following keywords are also important.

```
<method>
AFED
```

This keyword sets the simulation method. In the present case, “AFED” means adiabatic free energy dynamics.

```
<afed_type>
LOGMFD
```

This keyword sets the type of AFED. In the present case, “LOGMFD” means logarithmic mean force dynamics.

```
<logmfd_type>
NVE
```

This keyword sets the type of LogMFD. In the present case, “NVE” means NVE-type LogMFD. NVT (Nosé-Hoover dynamics) and VS (velocity scaling dynamics) are also available.

```
<niter_afed>
200
```

This keyword sets the number of AFED steps (total updates of the fictitious particle) in LogMFD. In the present case, “200” iterations.

```
<dt_logmfd>
```

```
1.0
```

This keyword sets the AFED step size, $\Delta t_{\rm cv}$, for LogMFD. In the present case, "1.0" (since the dynamics of LogMFD is physically meaningless, the unit can be arbitrarily chosen; e.g., fs).

```
<alpha_logmfd>
4.0
```

This keyword sets the parameter $\alpha_{\log}(>0)$ in LogMFD. In `input.dat`, $\alpha'(=\alpha_{\log}k_BT)$ should instead be given by this keyword, where T is the physical temperature set by the keyword <temperature>. In the present case, $\alpha_{\log}=4/k_BT$ hartree^{-1}.

```
<gamma_logmfd>
0.25
```

This keyword sets the parameter $\gamma_{\log}$ in LogMFD. In `input.dat`, $\gamma'(=\gamma_{\log}/k_BT)$ should instead be given by this keyword. Normally, $\gamma_{\log}=1/\alpha_{\log}$ works well (default setting). In the present case, $\gamma_{\log}=0.25k_BT$ hartree.

```
<start_afed>
MANUAL
0.2  1.3e-2
1.67  -6.3e-1
...
```

This keyword sets the initial position and velocity of each fictitious CV particle. If "MANUAL" is specified in the first line, the initial position and velocity of each CV particle should be given for each line as in the example above. In this case, the initial position of the first CV particle is 0.2 and its velocity is 1.3e-2. Those of the second CV particle are 1.67 and -6.3e-1, respectively. If "AUTOMATIC" is instead specified (default setting), the initial positions are automatically estimated from the initial configuration of the MD system, and the initial velocities are set by random numbers corresponding to the physical temperature given by the keyword <temperature>.

```
<start_therm_afed>
MANUAL
0  0
```

This keyword sets the initial position and velocity of the Nosé-Hoover thermostat variable η. If "MANUAL" is specified, its initial position and velocity should be given; in this case, $\eta(0)=0$ and $v_\eta(0)=0$ (default setting). If "AUTOMATIC" is instead specified, η is set to "0" and v_η is automatically set according to the preset physical temperature.

```
<mass_afed>
MANUAL
1
4
...
```

This keyword sets the parameters that re-adjust the mass of the fictitious CV particles. If "MANUAL" is set, the mass determined by <params_afed> (see Sec.5.4.1) is multiplied by the number given below "MANUAL" (the ith line is for the mass of the ith particle). In this example, the mass of the first CV particle is multiplied by "1" and that of the second CV particle is multiplied by "4". If "AUTOMATIC" is instead specified (default setting), the masses are not re-adjusted.

```
<niter_bath_afed>
100
```

This keyword sets the number of AFED steps, n_b, associated with the mass of the Nosé-Hoover thermostat, Q_η. In the present case, "100" (no unit); i.e., $Q_\eta = Dk_BT(n_b\Delta t_{\rm cv})^2$, where D is the number of fictitious particles and T is the physical temperature given by the keyword <temperature>.

```
<fenergy_ini_logmfd>
0.01
```

This keyword sets the initial value of the free energy estimated on-the-fly during the LogMFD run. This defines the origin of the energy scale. In the present case, the initial free energy is "0.01" hartree.

```
<ioption_afed>
1 or 2
```

This keyword sets the weight W_l in parallel dynamics discussed in Sec.13.16. With "1", $W_l = 1$ for all beads (replicas), while W_l is set by Eq.(204) (see Sec.13.16) with "2". In the current PIMD 2.8.0 source there is no <iprint_weight> input keyword. When the parallel-weight branch (<ioption_afed>=2) is active, the replica weights are appended to afed_weight.out at each AFED iteration after initialization. The legacy <iprint_weight> tag must not be used.

5.5 Phonon calculation (PHONON)

The theoretical background of phonon calculation (PHONON) is briefly explained in Section 13.6.

As an example, a set of input files for nickel FCC crystal is available. So, let us copy it to the directory *run*:

```
> cp -r examples/03_optimization_properties/Ni/phonon_eam run
```

and enter that directory:

```
> cd run/phonon_eam
```

Let us look at the keywords one by one.

```
<method>
PHONON
```

This keyword sets the simulation method, in the present case, the phonon calculation.

```
<ipotential>
EAM
```

This keyword sets the potential, in the present case, the embedded atom method.

```
<iboundary>
1
 33.25917740   0.00000000   0.00000000
  0.00000000  33.25917740   0.00000000
  0.00000000   0.00000000  33.25917740
```

This keyword sets the boundary condition. In the present case, "1" means that the periodic boundary condition is applied, and the following nine numbers indicate that the simulation box is cubic with a side length of 33.25917740 bohr.

Now, the keywords specific to the phonon calculation are listed below.

```
<cells_phonon>
5 5 5
```

This keyword sets the number of unit cells in the box along the a, b, c axes. In the present case, there are $125 = 5 \times 5 \times 5$ unit cells in the box with the size $\vec{a}_{u.c.} = (L, 0, 0)$, $\vec{b}_{u.c.} = (0, L, 0)$, and $\vec{c}_{u.c.} = (0, 0, L)$, where $L = 33.25917740/5 = 6.65183548$.

```
<kdisp_phonon>
20
0.00000 0.0 0.0
0.05000 0.0 0.0
0.10000 0.0 0.0
0.15000 0.0 0.0
0.20000 0.0 0.0
0.25000 0.0 0.0
0.30000 0.0 0.0
0.35000 0.0 0.0
0.40000 0.0 0.0
0.45000 0.0 0.0
0.50000 0.0 0.0
0.55000 0.0 0.0
0.60000 0.0 0.0
0.65000 0.0 0.0
0.70000 0.0 0.0
0.75000 0.0 0.0
0.80000 0.0 0.0
0.85000 0.0 0.0
0.90000 0.0 0.0
0.94458 0.0 0.0
```

This keyword sets the dimensional wave vectors for the calculation of the phonon dispersion curve. The three components enter the phase $\mathbf{g} \cdot \mathbf{d}$ directly and therefore have units of inverse length; they are not fractional reciprocal-lattice coordinates. The legacy native unit is bohr^{-1}. In TOML, set `dispersion_kpoints_unit = "1/bohr"` or `"1/angstrom"` explicitly. In the present case, 20 points are computed up to $k_{\max} = 2\pi/L = 0.94458$ bohr^{-1}. The results are displayed in the output file `phonon_kdisp.out`.

```
<kdos_phonon>
3 3 3
```

This keyword sets the k-point sampling for the calculation of the phonon density of states. In the present case, $3 \times 3 \times 3$ is chosen.

```
<dosrange_phonon>
0.0  500.0   5.0
```

This keyword sets the frequency range of the phonon density of states, which is printed to the output file `phonon_dos.out`. In the present case, the range is set from 0.0 cm^{-1} to 500.0 cm^{-1} with an increment of 5.0 cm^{-1}.

```
<temprange_phonon>
0.0   900.0   5.0
```

This keyword sets the temperature range of the phonon energies to be displayed in the output file `phonon_energy.out`. In the present case, the range is set from 0.0 K to 900.0 K with an increment of 5.0 K.

The output files `*.out` will be explained in Section 12.2 and Section 12.3, respectively.

5.6 Optimization of OM action (OMOPT)

The theoretical background of the OM action is briefly explained in Section 13.8.

As an example, a set of input files for the water dimer is available. Copy it to the directory *run*:

```
> cp -r examples/05_reaction_paths_rates/H4O2/omopt_cff run
```

and enter that directory:

```
> cd run/omopt_cff
```

Let us look at the keywords one by one.

```
<method>
OMOPT
```

This keyword sets the simulation method, the string method.

```
<ipotential>
CFF
```

This keyword sets the potential. In the present case, the CFF force field is chosen.

```
<iboundary>
0
```

This keyword sets the boundary condition. Here, "0" means the free boundary condition.

```
<nbead>
120
```

This keyword sets the number of images, i.e., the number of beads in the string method. In the present case, 120 images are chosen.

```
<nstep>
10000
```

This keyword sets the maximum number of updates of the string. Here, 10000 steps are chosen.

```
<dt>
0.625d0
```

This keyword sets the step size for updates of the string. In the present case, 0.625 femtoseconds.

```
<iprint_std>
100
```

This keyword sets the print interval of standard output. Here, it is set to every 100 steps.

```
<gamma_om>
0.00625d0
```

This keyword sets the friction constant in femtoseconds^{-1}. In the present case, it is 0.00625 femtoseconds^{-1}.

```
<fdiff>
1.d-6
```

This keyword sets the maximum shift of atomic coordinates, δ, used to obtain the numerical gradient of the OM action.

The output files `*.out` and the restart files `*.ini` are explained in Section 12.2 and Section 12.3, respectively.

To run from scratch, one can start from a straight-line interpolation of the atomic positions of the reactant and product ends, which are given in the file `structure.dat`. Alternatively, one can prepare the restart file `string.ini`, in which all the atomic positions of the 120 images are listed. When both the restart file (`string.ini`) and the input file (`structure.dat`) are present, the restart file always has priority.

When the keyword

```
<method>
TESTOM
```

is used instead of **<method>** = `OMOPT`, the numerical gradient of the OM action is tested.

5.7 User-made potentials

The PIMD code can be modified to implement user-defined potentials. The simplest approach is to edit the source files `force_user.F90` and `force_user_MPI.F90`, which are used by the serial and parallel versions of the code, respectively.

```
c***********************************************************************
      subroutine force_user
c***********************************************************************

c-----------------------------------------------------------------------
c     /*   shared variables                                           */
c-----------------------------------------------------------------------

      use common_variables, only :
     &   x, y, z, fx, fy, fz, pot, natom, nbead

c-----------------------------------------------------------------------
c     /*   local variables                                            */
c-----------------------------------------------------------------------

      implicit none

      .....
      .....

      return
      end


c***********************************************************************
      subroutine force_user_MPI
c***********************************************************************

c-----------------------------------------------------------------------
c     /*   shared variables                                           */
c-----------------------------------------------------------------------

      use common_variables, only :
```

```
     &   x, y, z, fx, fy, fz, pot, natom, nbead, myrank, nprocs

c-----------------------------------------------------------------------
c     /*   local variables                                            */
c-----------------------------------------------------------------------

      implicit none

      .....
      .....

      return
      end
```

After modifying these routines, compile the code in the usual way, as described in Section 1. To use the user-defined potential, set <ipotential> = USER in input.dat.

To verify the correctness of user-made potentials, it is strongly recommended to compare analytical forces and virial values with numerical ones obtained by finite differences. For this purpose, the following test methods are available:

- <method> = TESTFORCE: Tests analytical forces against numerical derivatives obtained by displacing atomic positions. This test works for both free boundary conditions (<iboundary> = 0) and periodic boundary conditions (<iboundary> = 1).
- <method> = TESTVIRIAL: Tests the analytical virial matrix by comparing it with numerical derivatives of the potential energy with respect to the box matrix. This test is valid only for periodic boundary conditions (<iboundary> = 1).

The absolute acceptance tolerances are controlled by <testforce_tolerance> (default 10^{-7} hartree/bohr) and <testvirial_tolerance> (default 10^{-5} hartree/bohr). The latter applies to the cell derivative $-\partial V/\partial h$, rather than to the virial tensor in energy units. A component passes only when its full-precision absolute error is strictly below the tolerance. For example:

```
<testforce_tolerance>
1.e-7
<testvirial_tolerance>
1.e-5
```

In TOML, use [numerical].force_tolerance and [numerical].virial_tolerance.

A successful derivative test returns exit status zero and prints the usual normal-termination message. A failed test uses the normal error-handling path with a nonzero exit status. Numerical mismatches are reported as FAIL; non-finite values or incomplete sweeps as ERROR. On failure, PIMD prints a DERIVATIVE diagnostic in the normal output and standard.out: the method, checked and expected component counts, failed and non-finite component counts, maximum finite absolute error, configured tolerance, and the reason for failure.

5.7.1 Testing forces

Starting from a static calculation setup, replace <method> = STATIC with

```
<method>
TESTFORCE
```

The precision of the numerical differentiation is controlled by the keyword

```
<fdiff>
1.e-4
```

which sets the displacement δ used to compute numerical forces:

$$-\frac{\partial V}{\partial r_{i\alpha,l}} = -\frac{V(r_{i\alpha,l}+\delta) - V(r_{i\alpha,l}-\delta)}{2\delta}. \tag{1}$$

During execution, output similar to the following is printed:

```
================================================================================
  atom  bead  F (analytical)   F (numerical)  stat  wall clock time
--------------------------------------------------------------------------------
     1     1    0.2275176522    0.2275176520    OK  2015-05-19 TUE 10:08:23.N
     1     1   -0.0197918981   -0.0197918968    OK  2015-05-19 TUE 10:08:23.N
     1     1    0.0547412167    0.0547412165    OK  2015-05-19 TUE 10:08:23.N
   ...   ...    ............    ............    ..  .......... ... ..........
```

The third and fourth columns show the analytical and numerical values of $-\partial V/\partial r_{i\alpha,l}$ for each Cartesian component. The status column reports "OK" if the difference is within the tolerance; otherwise it is marked as "X".

5.7.2 Testing virial

To test the virial, replace <method> = STATIC with

```
<method>
TESTVIRIAL
```

The numerical differentiation step size is controlled by

```
<fdiff>
1.e-3
```

which sets the displacement δ of the box matrix:

$$-\frac{\partial V}{\partial h_{\alpha\beta}} = -\frac{V(h_{\alpha\beta}+\delta) - V(h_{\alpha\beta}-\delta)}{2\delta}. \tag{2}$$

The analytical derivatives are obtained from the virial matrix:

$$-\frac{\partial V}{\partial h_{\alpha\beta}} = \sum_{\gamma=1}^{3} A_{\alpha\gamma} h^{-1}_{\beta\gamma}, \tag{3}$$

where $A_{\alpha\gamma}$ is the virial matrix.

Typical output is as follows:

```
================================================================================
     i     j  -dV/dH (analy)  -dV/dH (numer)  stat  wall clock time
--------------------------------------------------------------------------------
     1     1      0.15682910      0.15682911    OK  2015-05-19 TUE 16:21:53.N
     1     2     -0.11805355     -0.11805355    OK  2015-05-19 TUE 16:21:54.N
     ...
```

Here, "OK" indicates agreement within tolerance. By definition, virial tests are only valid for periodic boundary conditions.

5.7.3 Testing Ewald parameters

For non-polarizable CFF force fields (<ipotential> = CFF) or polarizable force fields (<ipotential> = POL, OSS), the Ewald parameters can be tested. These include the precision parameter ϵ and the ratio between real and reciprocal space contributions.

Starting from a static calculation, replace <method> = STATIC with

```
<method>
TESTEWALD
```

for charge–charge interactions, or

```
<method>
TESTEWPOL
```

for charge–dipole and dipole–dipole interactions in polarizable force fields.

During execution, output similar to the following is produced:

```
================================================================================
 epsilon   ratio n1 n2 n3 l1 l2 l3     energy [au]             wall clock time
--------------------------------------------------------------------------------
 0.1E-07   0.100  3  3  3  5  5  5  -339.755346433  2020-05-10 Sun 09:11:53.93
 0.1E-06   0.100  3  3  3  5  5  5  -339.755347885  2020-05-10 Sun 09:11:55.52
 ...
```

Here, n1 n2 n3 denote real-space cutoffs, and l1 l2 l3 denote reciprocal-space cutoffs, which are automatically determined from ϵ and the ratio. Initial values of these parameters are controlled by the CFF electrostatic settings in input.lammps, including kspace_style and the # PIMD_CFF ewpol extension for polarizable calculations. As with virial tests, Ewald parameter tests are valid only under periodic boundary conditions.

5.8 Mechanochemistry

The theoretical background of mechanochemical calculations (MECH) is briefly explained in Section 13.1.

The external force explicitly included (EFEI) method for mechanochemical calculations [40] is implemented in the PIMD code. The EFEI method can, in principle, be combined with any other methods. An example for a water molecule can be found in the directory **examples/03_optimization_properties/H2O/geoopt_efei_water**. Enter that directory:

```
> cd run/geoopt_efei_water
```

The following keywords specify the inclusion of the external force.

```
<mech_type>
EFEI
```

This keyword sets the type of mechanochemical calculation to the external force explicitly included (EFEI) method.

```
<efei>
2  3  0.01
```

This keyword sets the parameters of the EFEI method. In this example, a repulsive force is applied between the pair of atoms 2 and 3 with an amplitude of $F = 0.01$ hartree/bohr.

```
<iprint_mech>
1
```

This keyword sets the print interval for the energy components to the output file, mech.out.

When the PIMD code is executed, the output file, `mech.out`, will be generated. See Section 12.2 for the data format of the output file.

5.9 Nonadiabatic dynamics

The following methods for nonadiabatic dynamics are implemented in the PIMD code:

- `TFS`: surface hopping dynamics (Tully's fewest switches).
- `MFE`: mean field dynamics (Ehrenfest mean field).

5.9.1 Surface hopping dynamics

The theoretical background of surface hopping dynamics is briefly explained in Section 13.17.

A self-contained one-particle example using the built-in Tully model is distributed and is recommended for testing the TFS implementation. To use it, copy the files to the directory *run*:

```
> cp -r examples/09_nonadiabatic/H/tfs_tully run
```

and enter that directory:

```
> cd run/tfs_tully/
```

The distributed Tully example uses short smoke-test settings. The keyword blocks below show the more general multi-trajectory setup used with an electronic-structure backend; numerical values therefore need not match the minimal Tully case. To prepare the file `input.dat`, all the keywords given in Sections 4.1 and 4.2 are important. In addition, the following keywords are also required.

```
<method>
TFS
```

This keyword sets the simulation method. In the present case, surface hopping dynamics.

```
<nbead>
4
```

This keyword sets the number of independent trajectories, i.e., the number of beads; in the present case, four trajectories.

```
<dt>
0.25d0
```

This keyword sets the step size, in the present case, 0.25 femtoseconds.

```
<nstep>
400
```

This keyword sets the number of steps, in the present case, 400 steps.

```
<nref>
100
```

This keyword controls the time increment of the state coefficients; in the present case, 1/100 of <dt>.

```
<nstate>
2
```

This keyword sets the number of adiabatic electronic states. In the present case, two states are chosen: the ground state and the first excited state.

```
<istate_init>
2
```

This keyword sets the state occupied initially. In the present case, the second lowest state (the first excited state) is chosen.

```
<temperature>
300.d0
```

This keyword sets the initial temperature of the system. The velocities are generated randomly according to the Maxwell-Boltzmann distribution at this temperature; in the present case, 300 kelvin.

```
<iprint_tfs>
1
```

This keyword sets the print interval of the output file `tfs.out`; in the present case, every step.

```
<iprint_nac>
1
```

This keyword sets the print interval of the output file `nac.out`; in the present case, every step.

Input data for MOLPRO.. In the PIMD code, nonadiabatic dynamics, i.e., surface hopping dynamics and mean field dynamics, can be carried out with the MOLPRO code. The MOLPRO input file, `molpro.dat`, should be present in the execution directory. The following is an example for the LiH molecule using the CASSCF method and the 6-31g(d,p) basis set. The input file sets up the static calculation of the potential energy, the dipole moment, the forces, and the nonadiabatic coupling elements. It outputs the data to the file *table.molpro*.

```
> cat molpro.dat

***,title
memory,100,m
basis=6-31g(d,p)
geomtyp=xyz
geometry={nosym;noorient;
2
comment
li  0.0000   0.0000   0.0000
h   5.0000   0.0000   0.0000
}
int;
multi;
occ,3;frozen,0;closed,1;wf,4,1,0;state,2;weight,1,1
cpmcscf,nacm,1.1,2.1,accu=1.d-7,record=5101.1
cpmcscf,grad,1.1,accu=1.d-7,record=5102.1
cpmcscf,grad,2.1,accu=1.d-7,record=5103.1

e=energy
table,e                    !print a table with results
```

```
save,table.molpro          !save to file
title,<energy>             !title for the table
format,(e24.16)            !explicit format

table,dmx,dmy,dmz          !print a table with results
save,table.molpro          !save to file
title,<dipole>             !title for the table
format,(3e24.16)           !explicit format

force
samc,5101.1
varsav
table,gradx,grady,gradz    !print a table with results
save,table.molpro          !save to file
title,<nacme> 1 2          !title for the table
format,(3e24.16)           !explicit format

force
samc,5102.1
varsav
table,gradx,grady,gradz    !print a table with results
save,table.molpro          !save to file
title,<grad> 1             !title for the table
format,(3e24.16)           !explicit format

force
samc,5103.1
varsav
table,gradx,grady,gradz    !print a table with results
save,table.molpro          !save to file
title,<grad> 2             !title for the table
format,(3e24.16)           !explicit format
```

This file can be tested by running MOLPRO directly:

```
> molpro < molpro.dat
```

If this works correctly, the PIMD code is ready to run.

Upon executing PIMD, the output file `tfs.out` is generated.

```
> less tfs.out
```

```
.....
=============================================================================
    step  bead   s occ     pop/s   ratio/s     potential/s   potential/occ
-----------------------------------------------------------------------------
       0     1   1   2  0.000000  0.000000     -7.92966939     -7.86722151
       0     1   2   2  1.000000  1.000000     -7.86722151     -7.86722151
       0     2   1   2  0.000000  0.000000     -7.92966939     -7.86722151
       0     2   2   2  1.000000  1.000000     -7.86722151     -7.86722151
       0     3   1   2  0.000000  0.000000     -7.92966939     -7.86722151
       0     3   2   2  1.000000  1.000000     -7.86722151     -7.86722151
       0     4   1   2  0.000000  0.000000     -7.92966939     -7.86722151
       0     4   2   2  1.000000  1.000000     -7.86722151     -7.86722151
       1     1   1   2  0.000004  0.000000     -7.92988085     -7.86730272
       1     1   2   2  0.999996  1.000000     -7.86730272     -7.86730272
       1     2   1   2  0.000003  0.000000     -7.92947539     -7.86714613
```

```
       1     2   2   2  0.999997  1.000000     -7.86714613     -7.86714613
       1     3   1   2  0.000004  0.000000     -7.92988085     -7.86730272
       1     3   2   2  0.999996  1.000000     -7.86730272     -7.86730272
       1     4   1   2  0.000003  0.000000     -7.92947539     -7.86714613
       1     4   2   2  0.999997  1.000000     -7.86714613     -7.86714613
.....
```

The first three columns are the step number, the bead number, and the state number. The fourth column is the occupied state of the corresponding bead. The fifth column is the population of the corresponding state for each bead. The sixth column is the occupancy ratio of the corresponding state among all the beads. The seventh and eighth columns are the potential energies of the corresponding state and the occupied state, respectively, for each bead.

5.9.2 Mean field dynamics

The theoretical background of mean field dynamics is briefly explained in Section 13.17.

A self-contained one-particle example using the built-in Tully model is distributed and is recommended for testing mean-field dynamics. Copy it to the *run* directory:

```
> cp -r examples/09_nonadiabatic/H/mfe_tully run
```

Then enter that directory:

```
> cd run/mfe_tully/
```

The distributed Tully case uses deliberately short validation settings. The keyword and output blocks below retain the historical multi-trajectory illustration. The keywords for the file `input.dat` are basically the same as those in Section 5.9.1, except that the keyword **<method>** = TFS is replaced by

```
<method>
MFE
```

and the keyword **<iprint_tfs>** is replaced by

```
<iprint_mfe>
1
```

which controls the print interval of the output file, `mfe.out`.

When PIMD is executed, the output file `mfe.out` is generated.

```
> less mfe.out
```

```
.....
================================================================================
    step  bead   s occ     pop/s     occ/s       pot/state         pot/occ
--------------------------------------------------------------------------------
       0     1   1   0  0.000000  0.000000     -7.92966939     -7.86722151
       0     1   2   0  1.000000  1.000000     -7.86722151     -7.86722151
       0     2   1   0  0.000000  0.000000     -7.92966939     -7.86722151
       0     2   2   0  1.000000  1.000000     -7.86722151     -7.86722151
       0     3   1   0  0.000000  0.000000     -7.92966939     -7.86722151
       0     3   2   0  1.000000  1.000000     -7.86722151     -7.86722151
       0     4   1   0  0.000000  0.000000     -7.92966939     -7.86722151
       0     4   2   0  1.000000  1.000000     -7.86722151     -7.86722151
       1     1   1   0  0.000004  0.000003     -7.92988085     -7.86730277
```

```
       1     1   2   0  0.999996  0.999997     -7.86730272     -7.86730277
       1     2   1   0  0.000003  0.000003     -7.92947539     -7.86714619
       1     2   2   0  0.999997  0.999997     -7.86714613     -7.86714619
       1     3   1   0  0.000004  0.000003     -7.92988085     -7.86730277
       1     3   2   0  0.999996  0.999997     -7.86730272     -7.86730277
       1     4   1   0  0.000003  0.000003     -7.92947539     -7.86714619
       1     4   2   0  0.999997  0.999997     -7.86714613     -7.86714619
.....
```

The first three columns are the step number, the bead number, and the state number. The fourth column is always zero (not meaningful, since the occupied state is mixed). The fifth column shows the population of the corresponding state for each bead. The sixth column shows the occupancy ratio of the corresponding state among all beads. The seventh and eighth columns are the potential energies of the corresponding state and the mixed state, respectively, for each bead.

5.10 Molecular dynamics of rigid molecules

A compact rigid-water example is distributed for direct testing. The more heterogeneous H_2O/CN/Na input blocks shown below are retained as an illustration of the GENERAL, LINEAR, and MONOATOM component forms. To use it, copy the files to the directory *run*:

```
> cp -r examples/03_optimization_properties/water_32/rotor_nvt_cff run
```

and enter that directory:

```
> cd run/rotor_nvt_cff/
```

To prepare the file `input.dat`, all the keywords given in Sections 4.1 and 4.2 are important. In addition, the following keywords are also required.

```
<method>
ROTOR
```

This keyword sets the simulation method, which in the present case is molecular dynamics of rigid molecules.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble to NVT.

```
<dt>
0.25d0
```

This keyword sets the step size, which in the present case is 0.25 femtoseconds.

```
<nstep>
400
```

This keyword sets the number of steps, which in the present case is 400.

```
<temperature>
300.d0
```

This keyword sets the temperature of the system, which in the present case is 300 kelvin.

```
<iboundary>
   1
  37.237981   0.000000   0.000000
   0.000000  37.237981   0.000000
   0.000000   0.000000  37.237981
```

This keyword sets the boundary condition and the initial box. Here, "1" indicates a periodic boundary, and the following three lines define the matrix of the initial box. In the present case, a cubic box with a side length of 37.237981 bohr is used.

```
<iread_exit>
10
```

This keyword sets the interval for reading `exit.dat`, which in the present case is every 10 steps.

```
<iprint_std>
10

<iprint_rest>
10

<iprint_rdf>
10

<iprint_trj>
10

<iprint_xyz>
10

<iprint_dip>
10
```

These keywords set the print intervals for the standard output, the restart files, the radial distribution functions, the trajectories in the regular and xyz formats, and the dipole moments, respectively.

Now, the keywords specific to **<method>** = ROTOR are introduced. For new TOML input, define the rigid reference geometry and molecule instances in `[molecular_components]` and `[molecular_components.component.N]`; see Section 2. The following **<components>** and **<molecules>** records are the legacy engine representation.

```
<components>
 3
 H2O     4     GENERAL
   -1.430429   -0.984140    0.000000
    1.430429   -0.984140    0.000000
    0.000000    0.123017    0.000000
    0.000000   -0.160441    0.000000
 CN      4     LINEAR
    0.000000    0.000000   -1.020452
    0.000000    0.000000    0.805023
    0.000000    0.000000    1.190527
    0.000000    0.000000    1.610047
 Na      1     MONOATOM
    0.000000    0.000000    0.000000
```

This keyword defines the molecular components. There are three components: H_2O, CN, and Na. The H_2O

and CN molecules each have four interaction sites, while Na has a single interaction site. Since the CN molecule is linear, its atoms must be arranged along the z axis.

```
<molecules>
  256
H2O     1  255  509  763
H2O     2  256  510  764
...   ...  ...  ...  ...
...   ...  ...  ...  ...
H2O   254  508  762 1016
CN   1017 1018 1019 1020
Na   1021
```

This keyword sets the number of molecules, which in the present case is 256. The following lines assign the site numbers for each molecule.

5.10.1 Gentlest ascent dynamics (GAD)

In the PIMD code, gentlest ascent dynamics is implemented [81].

As an example, a small H_4O_2 molecular-mechanics GAD case is available. Let us copy it to the directory *run*:

```
> cp -r examples/05_reaction_paths_rates/H4O2/gad_cff run
```

and enter that directory:

```
> cd run/gad_cff
```

To prepare the file `input.dat`, all the keywords given in Sections 4.1 and 4.2 are important. In addition, the following keywords are also required.

```
<method>
GAD
```

This keyword sets the simulation method. In the present case, “GAD” corresponds to gentlest ascent dynamics.

```
<dt>
0.01d0
```

This keyword sets the step size in hartree$^{0.5}$ femtoseconds.

```
<nstep>
1000
```

This keyword sets the number of GAD steps.

```
<iprint_xyz>
1
```

This keyword sets the print interval for the trajectory in xyz format. In the present case, every ten steps.

```
<nbead>
8
```

This keyword sets the number of GAD trajectories computed in parallel. In the present case, eight trajectories.

```
<fdiff>
1.d-3
```

This keyword sets the finite difference parameter used in GAD. In the present case, 10^{-3} bohr.

```
<irandom>
0
```

This keyword sets the random number seed. In the present case, the random numbers start from a predefined seed.

```
<gamma_gad>
0.1
```

This keyword sets the gamma value for GAD. In the present case, "0.1" atomic units. The legacy tag <energy_max_gad> is not read by the current PIMD 2.8.0 source and must not be used. The present GAD implementation has no corresponding user input keyword for a maximum-energy cutoff.

The output files *.out and the restart files *.ini will be explained in Section 12.2 and Section 12.3, respectively.

5.11 Consecutive static calculations (SCAN)

Consecutive static calculations are performed using <method> = SCAN. A series of molecular configurations is read from the file structure.dat. Upon executing the PIMD code, the potential energy, forces, and dipole moment are computed for each molecular configuration based on <ipotential>. The results are printed to the following files: coord_scan.out (atomic coordinates), potential_scan.out (potential energy), forces_scan.out (forces), and dipole_scan.out (dipole moment).

In addition, an external shell script can be executed for each structure. The file name of the shell script is specified by the keyword <scan_exe_shell>.

Part III

Potential Models and Electronic-Structure Interfaces

6 Potentials

This section explains the data contained in the potential files. The file names and required data depend on the potential being used. Input files for external codes (e.g., *ab initio* electronic-structure programs) must be pre-

pared by the user. In principle, the PIMD code reuses the same input files that are valid for the corresponding external codes. For simplicity, most examples presented below use a water molecule with different potential models.

IMPORTANT: Only the geometry information provided in the PIMD input files—`structure.dat`, `geometry.ini`, or `string.ini`—is used by the PIMD code. Consequently, any geometry specified directly in the input files of external codes is ignored during PIMD simulations.

6.1 Empirical force fields

In the PIMD code, the following empirical force fields are implemented:

- CFF: LAMMPS-compatible classical force field.
- EAM: embedded atom method.
- ADP: angular dependent potential.
- PAIR: pair potential.
- TERSOFF: Tersoff potential.

6.1.1 TIP4P through CFF

The preferred built-in q-TIP4P/F model is selected with **<ipotential>** = WATER and the default **<model_water>** = 0; it does not require user-supplied force-field files. TIP4P-family models can also be supplied explicitly through **<ipotential>** = CFF. In that explicit CFF route, the molecular geometry and force-field parameters are supplied in the ordinary CFF files `input.lammps`, `data.lammps`, and `parm.lammps`. The TIP4P virtual-site geometry is enabled by a PIMD CFF extension in `input.lammps`, for example

```
# PIMD_CFF tip4p O H 0.73612
```

where the first two fields are the oxygen and hydrogen symbols and the final value is γ in $\vec{r}_{\mathrm{M}} = \gamma\vec{r}_{\mathrm{O}} + (1-\gamma)(\vec{r}_{\mathrm{H}'} + \vec{r}_{\mathrm{H}''})/2$. For every O atom, two unshared H atoms must form the water molecule; invalid TIP4P connectivity is rejected. Charges, bonded interactions, nonbonded exclusions, pair coefficients, and long-range electrostatics are specified with the corresponding CFF/LAMMPS records. TIP4P combined with induced polarization is currently not supported. See **examples/02_path_integral_dynamics/ice_ih/pimd_npt_qtip4p** for a complete explicit-CFF q-TIP4P/F input set, and **examples/02_path_integral_dynamics/water_64/pimd_npt_tip4p** for the built-in WATER route.

6.1.2 Embedded atom method (EAM)

The embedded atom method is selected with

```
<ipotential>
EAM
```

and is configured from `input.lammps`. The supported direct LAMMPS form is

```
units metal
atom_style atomic
read_data data.lammps
pair_style eam/alloy
pair_coeff * * potential.eam.alloy Ni H
```

The element list after the potential file maps PIMD atom types to the element names stored in the DYNAMO setfl file. Every PIMD atom type must be mapped. The density and radial grids in the setfl file are retained independently; there is no requirement that `Nrho` and `Nr` be equal. Unsupported EAM families, including `eam/fs` and `eam/he`, terminate explicitly rather than being interpreted as `eam/alloy`. See Section 11.7 for the functional form and table conventions.

6.1.3 Angular-dependent potential (ADP)

The ADP potential is selected with

```
<ipotential>
ADP
```

and uses the LAMMPS extended-setfl format:

```
units metal
atom_style atomic
read_data data.lammps
pair_style adp
pair_coeff * * potential.adp Fe Ni
```

The ordinary EAM embedding, density, and pair tables are followed by the ADP $u(r)$ and $w(r)$ tables in the standard LAMMPS ADP ordering. Type to element mapping follows the same rule as for `eam/alloy`.

6.1.4 Tabulated pair potential

A purely tabulated pair potential is selected with

```
<ipotential>
PAIR
```

and uses LAMMPS `pair_style table`. For example,

```
units metal
atom_style atomic
read_data data.lammps
pair_style table spline 1000
pair_coeff 1 1 pair.table P1_1
```

PIMD reads the distance and energy columns from the named table section and constructs its cubic spline from those nodes. An explicit `pair_coeff` record is required for every atom-type pair.

6.1.5 Tersoff potential

The Tersoff potential is selected with

```
<ipotential>
TERSOFF
```

and is configured with the LAMMPS-compatible `input.lammps` interface. For example,

```
units metal
atom_style atomic
boundary p p p
read_data data.lammps
pair_style tersoff
pair_coeff * * C.tersoff C
neighbor 2.0 bin
run 0
```

`units metal` is required. The supported style is the standard `pair_style tersoff` without style options (including `shift`). The `pair_coeff` record must use `* * FILE MAP...`, with one element mapping for every PIMD atom type. Standalone TERSOFF does not accept `NULL` mappings. The named potential file uses the

native LAMMPS Tersoff parameter format. PIMD accepts the standard ordered element triplets, including entries split across physical lines, and supports the standard $m = 1$ and $m = 3$ forms. All ordered triplets required by the mapped elements must be present. A `neighbor SKIN ...` command may be used to set the Verlet-list skin; if it is omitted, the native force path uses 2.0 Å. A `read_data` command and `data.lammps` may be retained so that the same file can be checked directly with LAMMPS, but PIMD obtains its coordinates, masses, and cell from the normal PIMD structure/input path and does not require `data.lammps` for the force calculation. See Section 11.8 for the parameter-file conventions.

6.1.6 GAL21 potential

For water-metal interfaces, the GAL21 potential can be used by specifying the keyword

```
<ipotential>
METALWATER
```

for seven metals: Pt, Ag, Au, Co, Cu, Ni, and Pd. The metal–metal EAM subpotential can be selected with a LAMMPS DYNAMO setfl file through `input.lammps`. For example, for the usual Pt–water ordering (Pt, O, H),

```
units metal
atom_style atomic
read_data data.lammps
pair_style eam/alloy
pair_coeff * * Pt_GAL21.eam.alloy Pt NULL NULL
run 0
```

The mapping list follows the PIMD atom-kind order. Metal kinds must map to matching element symbols in the setfl file, whereas all water/nonmetal kinds must be `NULL`; GAL21 and the built-in water model continue to provide those interactions. The setfl reader preserves independent `Nrho` and `Nr` grids and reads the standard `r*phi(r)` pair tables without resampling. `units metal` is required and unsupported EAM families such as `eam/fs` are rejected explicitly. The LAMMPS file is required and selects only the metal EAM subpotential; PIMD coordinates, masses, cell, GAL21 terms, and water terms remain controlled by the PIMD input path. Nonmetal water atom types must be mapped to `NULL` in the EAM/alloy `pair_coeff` record. METALWATER also supports **<xmpi>**="ON" for the XMPI-enabled methods in Section 11.4.228. The implementation keeps the GAL21, q-TIP4P/F, and EAM equations single-source with ordinary MPI: bead ownership switches to the XMPI contiguous bead blocks, force work is distributed over the XMPI force groups, and potentials, forces, and virials are completed with the XMPI sub/main communicators. Variable-cell PIMD NPT itself remains an ordinary-MPI integrator; use PIMD/NVT/MNHC, TESTFORCE, TESTVIRIAL, or another method listed in Section 11.4.228 when enabling XMPI. See reference [50] for details.

6.2 User-defined potentials

User-defined potentials can be used by designating the keyword

```
<ipotential>
USER
```

along with the execution command, input file, and output file, which are designated by the keywords **<user_command>**, **<user_input_file>**, and **<user_output_file>**, respectively. Refer to Section 11.4 for further details.

6.3 Born–Oppenheimer potentials

In the PIMD code, Born–Oppenheimer potentials are evaluated by calling external electronic-structure programs based on *ab initio*, density functional theory (DFT), or semiempirical methods. At present, the PIMD code can be coupled with the following packages:

- ABINIT-MP [71] (ab initio; fragment molecular orbital method).
- CP2K [56] (ab initio).
- DFTB+ [55] (density functional tight binding).
- MOPAC [54] (semiempirical).
- GAMESS [63] (ab initio, density functional theory, semiempirical).
- GAUSSIAN 98, 03, 09 [59, 60, 61] (ab initio, density functional theory, semiempirical).
- MOLPRO [65] (ab initio).
- OpenMX [57] (density functional theory).
- ORCA [67] (ab initio, density functional theory).
- PHASE/0 [58] (density functional theory).
- QUANTUM ESPRESSO [68] (density functional theory).
- SMASH [70] (ab initio, density functional theory).
- TURBOMOLE [64] (ab initio, density functional theory).
- VASP 6 [69] (density functional theory).

Among these, CP2K, DFTB+, OpenMX, QUANTUM ESPRESSO, SMASH, and VASP 6 can be internally linked to the PIMD code, enabling efficient and tightly coupled simulations.

6.3.1 SMASH

SMASH is an open-source electronic structure code released under the Apache License 2.0.

PIMD provides a CMake internally linked 3.0.2 interface selected with `-DSMASH_EMBEDDED=ON`. This interface requires setting the SMASH input parameters through the keyword **<smash_options>**. Select the embedded 3.0.2 force path with **<ipotential>** set to `SMASH`.

To build the 3.0.2 interface, configure CMake with `-DSMASH_EMBEDDED=ON` and `-DOPENMP=ON`. CMake fetches a pinned revision from `https://github.com/cmsi/smash`, compiles separate serial and MPI libraries, and links them into the matching PIMD executables. In the MPI path, PIMD passes each force-group communicator to SMASH for internal MPI parallelism. Set `OMP_NUM_THREADS` for threads and allow a stack of about 64 MiB. An insufficient stack (for example, the common 8 MiB default) can cause segmentation faults in HF gradients. In the same shell or batch job script that launches PIMD, use `ulimit -s unlimited` on Linux, subject to the permitted hard limit. On macOS use `ulimit -s 65520` instead. PIMD passes bead coordinates to SMASH and receives energy and gradients in memory. One shared SMASH input is generated per setup; checkpoints and diagnostic output remain separate for each bead. The `SMASH` force option supports isolated full-system calculations and isolated QM/MM electronic and mechanical embedding. Periodic calculations and QM/MM with XMPI are unsupported. For QM/MM set **<ipotential>** to `QMMM`, **<qmmm_potential>** to `SMASH`, and **<qmmm_embedding>** to `EE` or `ME`. Provide the usual CFF files and region/link definitions, plus **<smash_options>** containing electronic settings without geometry. PIMD supplies primary atoms, link atoms and MM point-charge sites for EE, and sets gradient mode and bohr units. The block ends at the next PIMD tag or end of file and is read only from `input.dat`. For example:

```
<smash_options>
job method=hf basis=sto-3g memory=256mb
control print=minimum precision=high
scf dconv=1e-11
```

The geometry is supplied by PIMD. TOML users can use `[smash.job]`, `[smash.control]`, and `[smash.scf]`.

For full-system calculations, **<smash_dip>** set to “1” (default) returns the HF/DFT SCF dipole to PIMD; “0” disables this contribution. Energy-only calls also support dipoles. MP2 requires **<smash_dip>** set to “0”. QM/MM dipoles are handled independently of this flag.

The **<smash_guess>** keyword selects `PREVIOUS` (default), which reuses each bead’s checkpoint after its first force call, or `DEFAULT`, which uses SMASH’s native initial guess on every call. The native guess defaults to extended Hückel orbitals and can be configured in **<smash_options>**; it is not a random guess. TOML users can set `guess = "DEFAULT"` in `[smash]`.

The explicit `lib/smash/smash-3.0.2.patch` targets upstream revision `602ed4055ccafc2ccafe11c7d5895f25026b97da`. CMake applies it only to a build-tree copy and refreshes that copy when the patch or source changes. Run `ctest -R '^smash-'` for full-system and QM/MM numerical checks.

Execution of PIMD.. The PIMD calculation with SMASH is executed in the usual way using:

```
> pimd.x
```

or, for parallel execution,

```
> pimd.mpi.x
```

6.3.2 ABINIT-MP

IMPORTANT: The user is solely responsible for all issues regarding the ABINIT-MP license. PIMD does not provide any ABINIT-MP license.

The ab initio ABINIT-MP code is linked externally at the subroutine `force_abinit_mp5.F90` of the PIMD code. The ABINIT-MP code is linked internally at the subroutine `force_abinit_mp_MPI.F90`. ABINIT-MP must be installed before this interface can be used. The source code is complete, but it has not been tested with a sufficient number of examples. Therefore, the user's manual for this part is incomplete. To use these methods, the user is encouraged to carefully read the source code.

NOTE: The two interfaces use different runtime-file contracts. For the embedded/internal **<ipotential>** = `ABINIT-MP` route, the input file name is selected by **<abinit_mp_input>** (default `input.ajf`). For the legacy system-call **<ipotential>** = `ABINIT-MP5` route, `force_abinit_mp5.F90` instead reads the fixed prototype `abinit-mp.dat` and generates `input.ajf` from that prototype for each force calculation. Therefore `abinit-mp.dat` is the required pre-run file for ABINIT-MP5, whereas `input.ajf` is generated and must not be diagnosed as a missing input for that route.

6.3.3 CP2K

IMPORTANT: The user is solely responsible for all issues related to the CP2K license. PIMD does not provide a CP2K license.

The CP2K code for quantum chemistry and solid-state physics [56] is linked directly to PIMD through the interface implemented in the subroutine `force_libcp2k.F90`, which calls the F77 CP2K API. To use this interface, the CP2K API library must be compiled beforehand. This can be done, for example, by invoking `make ARCH=psmp libcp2k` in the directory where CP2K was compiled. Please consult the CP2K documentation for compilation instructions.

Linking the CP2K API (CMake). For modern CP2K installations, use the CMake package exported by CP2K:

```
> cmake -S . -B build-cp2k \
>   -DCP2K=ON -DCP2K_LINK_MODE=CMAKE \
>   -DCP2K_DIR=/path/to/cp2k-install
> cmake --build build-cp2k --target pimd.mpi.x -j
```

For older CP2K installations, use the pkg-config option:

```
> cmake -S . -B build-cp2k-legacy \
>   -DCP2K=ON -DCP2K_LINK_MODE=PKGCONFIG \
>   -DCP2K_DIR=/path/to/cp2k-install -DCP2K_VER=local
> cmake --build build-cp2k-legacy --target pimd.mpi.x -j
```

Here `CP2K_VER` is the ARCH name used when building CP2K (`local` in this example).

When CP2K is built with OpenMP support, PIMD links the required OpenMP libraries automatically. Other linked packages may still require `-DOPENMP=ON`, so we recommend enabling this option when using an OpenMP-enabled CP2K build.

Linking the CP2K API (Old method). This method was tested in February 2020 using a development version of CP2K and the Linux-x86-64-intel-minimal.psmp architecture file. To link to the CP2K API, first ensure that CP2K is built with the API enabled. Then make a few changes to the PIMD makefile around the following lines.

```
#-------------------------------------------------------------------------
#     Usage of CP2K as static library - without MPI
#-------------------------------------------------------------------------

#     compile PIMD with CP2K as a static library
#CP2KFLAG = -Dlibcpk

#     compile PIMD without CP2K as a static library
CP2KFLAG =

ifdef CP2KLIBFLAG
   CP2KDIR = /home/g3/a184053/QM-Programs/cp2k_new
   CP2KVER = Linux-x86-64-intel-minimal/ssmp

   CP2KINC  = -I$(CP2KDIR)/obj/$(CP2KVER)/
   CP2KLIB  = -L$(CP2KDIR)/lib/$(CP2KVER)/ \
              -lcp2k -L$(CP2KDIR)/lib/$(CP2KVER)/exts/dbcsr/ -ldbcsr
endif

#-------------------------------------------------------------------------
#     Usage of CP2K as static library - with MPI
#-------------------------------------------------------------------------

#     compile PIMD with CP2K as a static library
#CP2KMPIFLAG = -Dlibcpkmpi

#     compile PIMD without CP2K as a static library
CP2KMPIFLAG =

ifdef CP2KMPIFLAG
   CP2KDIRMPI = /home/g3/a184053/QM-Programs/cp2k_new
   CP2KVER = Linux-x86-64-intel-minimal/psmp

   CP2KINCMPI  = -I$(CP2KDIRMPI)/obj/$(CP2KVER)/
   CP2KLIBMPI  = -L$(CP2KDIRMPI)/lib/$(CP2KVER)/ \
            -lcp2k -L$(CP2KDIRMPI)/lib/$(CP2KVER)/exts/dbcsr/ -ldbcsr
endif
```

Please note that support for CP2K must be enabled separately for the serial and MPI-parallel versions of the code. Do not attempt to link an MPI-parallel CP2K build to the serial PIMD code, or vice versa. To enable compilation with the CP2K API, comment out either the *CP2KFLAG*= or *CP2KMPIFLAG*= line and remove the comment marker from the line containing the actual flag. The *CP2KDIR* (or the corresponding MPI variable) should be set to the top-level directory where CP2K was compiled. The *CP2KVER* (and the corresponding MPI variable) should be set to the `architecture/parallel-version` corresponding to the *libcp2k* library you built.

CP2K input file.. Before running the PIMD code, the CP2K input files [56] should be prepared. The CP2K input must correspond to a static calculation of the energy and gradient for the system of interest. The order of atoms in the CP2K input file must match that in the PIMD input files.

The exchange-correlation functional should be specified explicitly in the `&XC` block. For example, PBE can be selected with

```
&XC
  &XC_FUNCTIONAL PBE
  &END XC_FUNCTIONAL
&END XC
```

The chosen functional should be consistent with the pseudopotentials used in the `&KIND` sections (for example, a GTH-PBE potential for a PBE calculation). **pimd-prep** checks common GTH naming conventions, but custom pseudopotentials must still be verified by the user.

For periodic PIMD calculations, the CP2K interface obtains the stress tensor after force evaluation. The `cp2k.dat` input should therefore request it, for example,

```
STRESS_TENSOR ANALYTICAL
```

in the `&FORCE_EVAL` section. This is especially important for variable-cell simulations. **pimd-prep** 0.9.7 also checks `RESTART OFF` and periodic stress-tensor availability during `check`.

Here is an example of the `cp2k.dat` file for a system of 32 water molecules:

```
&FORCE_EVAL
  METHOD QS
  &DFT
    BASIS_SET_FILE_NAME GTH_BASIS_SETS
    BASIS_SET_FILE_NAME BASIS_MOLOPT
    POTENTIAL_FILE_NAME POTENTIAL
    &PRINT
      &MOMENTS
       PERIODIC TRUE
      &END
    &END
    &MGRID
      CUTOFF 300
    &END MGRID
    &QS
      EPS_DEFAULT 1.0E-12
    &END QS
    &SCF
      SCF_GUESS ATOMIC
      EPS_SCF      1.0E-5
      &MIXING
        ALPHA 0.4
      &END
      &PRINT
        &RESTART OFF
        &END RESTART
      &END PRINT
    &END SCF
    &XC
      &XC_FUNCTIONAL Pade
      &END XC_FUNCTIONAL
    &END XC
  &END DFT
  &SUBSYS
    &CELL
      ABC 9.8528 9.8528 9.8528
    &END CELL
   # 32 H2O (TIP5P,1bar,300K) a = 9.8528
```

```
    &COORD
      O       2.280398       9.146539       5.088696
   ...
      H       8.687134       8.667252       2.448452
    &END COORD
    &KIND H
      BASIS_SET DZVP-GTH
      POTENTIAL GTH-PADE-q1
    &END KIND
    &KIND O
      BASIS_SET DZVP-GTH
      POTENTIAL GTH-PADE-q6
    &END KIND
  &END SUBSYS
  STRESS_TENSOR ANALYTICAL
&END FORCE_EVAL
&GLOBAL
  PROJECT H2O-32
  PRINT_LEVEL MEDIUM
&END GLOBAL
```

The coordinates have been omitted here but can be found in the **examples/02_path_integral_dynamics/water_32/pimd_dipole_cp2k** directory. Using this input file, you can test CP2K execution by running:

```
cp2k.psmp -i cp2k.dat
```

Note that the *RESTART OFF* option under *SCF* is essential to ensure that no unnecessary files are written. This is especially important if the input-suppression keyword **<cp2k_lib_output>** is used in the PIMD input file, as discussed below.

PIMD input file.. The following keywords should be set in the `input.dat` file:

```
<ipotential>
CP2KLIB
```

This keyword declares the use of the CP2K code. Another useful option is the ability to control how much output from CP2K is printed for each calculation:

```
<cp2k_lib_output>
1
```

This can be set to -1, 0, or 1. The option -1 suppresses all CP2K output. The option 0 prints only the output of the first bead in the directory 001/. Setting it to 1 causes all beads to write their outputs in the numbered directories created in the main PIMD simulation directory. Note that if CP2K output is suppressed, it is especially important to ensure that CP2K does not create any restart or other data files as part of the single-point calculation described in the previous paragraph. This is because all CP2K processes launched will attempt to write to the same restart files located in the main simulation directory, which can cause file locking and severely reduce performance. Finally, if you want PIMD to output dipole moments, the CP2K calculation must also compute dipole moments. For example, to output dipoles every 10 steps, set:

```
<iprint_dip>
10
```

Execution of PIMD.. Execution is performed in the usual way using **pimd.x** or **pimd.mpi.x**. Note that it may be necessary to set the environment variable *OMP_NUM_THREADS* to obtain correct parallelization of

CP2K. If you are running the MPI version of the code, for example, *OMP_NUM_THREADS* times the number of MPI processes should be less than or equal to the number of hardware threads available on your system.

6.3.4 DFTB – Internal

IMPORTANT: The user is solely responsible for any issues related to the DFTB+ license. PIMD does not provide any DFTB+ license.

The semiempirical DFTB+ code [55] can be linked directly to PIMD via `force_libdftb.F90` or `force_libdftb_MPI.F90`, using calls to the DFTB+ application programming interface (API). Before using this functionality, the DFTB+ API library must be compiled in advance. DFTB+ 25.1 is supported using its CMake build and public Fortran API. Both **`pimd.x`** and **`pimd.mpi.x`** support non-MPI DFTB+ libraries, including OpenMP-enabled builds. With a non-MPI library, **`pimd.mpi.x`** parallelizes over beads, but not within each bead. MPI-enabled DFTB+ libraries are also supported by **`pimd.mpi.x`**, using `<np_beads>` bead groups and `<np_force>` workers per group with `<xmpi>` set to `OFF`.

Linking the DFTB+ API. Use the same Fortran compiler for DFTB+ and PIMD and, for MPI-enabled libraries, the same MPI implementation. The installed library must provide the API (`WITH_API=ON`). From the PIMD repository root, run the following, replacing `/path/to/dftbplus/install` with the library installation prefix:

```
cmake -S . -B build/pimd-dftbplus \
  -DCMAKE_BUILD_TYPE=Release -DOPENMP=ON -DDFTB+=ON \
  -DDFTB+_INSTALLDIR="/path/to/dftbplus/install"
cmake --build build/pimd-dftbplus --parallel 8
```

The installation prefix may also be supplied through *CMAKE_PREFIX_PATH* when *DFTB+_INSTALLDIR* is unset. The exported DFTB+ target supplies the library, Fortran module include paths and any required OpenMP dependency. When linking an OpenMP-enabled DFTB+ library, we recommend configuring PIMD with `-DOPENMP=ON` to enable OpenMP consistently across PIMD targets. This option is not required for the DFTB+ interface itself: the exported DFTB+ target automatically supplies the necessary OpenMP compiler flags and runtime library, even when PIMD's `OPENMP` option is `OFF`.

DFTB+ input file. Before running PIMD, the DFTB+ input file must be prepared. The DFTB+ input should correspond to a static calculation of the energy and gradients for the system of interest. PIMD supplies the atomic species, atom ordering, initial coordinates, periodicity and lattice vectors through the API before initializing the DFTB+ calculator. The coordinates and cell are subsequently updated for each bead. No separate DFTB+ geometry file is required.

The following rules apply to the DFTB+ input file:

- The file name must be `dftb.dat`.
- The *Geometry* block can be omitted. If present, it is replaced by the geometry already read by PIMD.
- Element names in the Hamiltonian and parameter settings must match the chemical symbols supplied by PIMD.

An example `dftb.dat` file for a water molecule is shown below:

```
Driver = {}

Hamiltonian = DFTB {
  SCC = Yes
  SCCTolerance = 1.0e-5
  MaxSCCIterations = 1000
  Mixer = Broyden {
    MixingParameter = 0.2
  }
  SlaterKosterFiles = {
    O-O = "O-O.skf"
    O-H = "O-H.skf"
```

```
    H-O = "H-O.skf"
    H-H = "H-H.skf"
  }
  MaxAngularMomentum = {
    O = "p"
    H = "s"
  }
  Charge = 0.0
  SpinPolarisation = {}
  Filling = Fermi {
    Temperature [Kelvin] = 0.0
  }
}

Options = {
  WriteDetailedOut = No
  RestartFrequency = 0
}

Analysis = {
  PrintForces = Yes
  WriteBandOut = No
}

ParserOptions = {
  ParserVersion = 14
}
```

The required Slater–Koster files (`*.skf`), such as *O-O.skf*, *O-H.skf*, *H-O.skf*, and *H-H.skf*, must be present in the execution directory. Alternatively, their location can be specified using the *Prefix* keyword inside the *SlaterKosterFiles* block.

To test the DFTB+ input with the standalone executable, first add a *Geometry* block containing the corresponding molecular or periodic structure to a copy of the input. The standalone executable does not receive PIMD's API geometry. Then execute:

```
cp dftb.dat dftb_in.hsd
dftb+
```

Both the serial and MPI interfaces override the following DFTB+ input settings, including when linked to a non-MPI library:

- `Options`: `WriteDetailedOut = No`, `WriteResultsTag = No`, `WriteAutotestTag = No`, and `RestartFrequency = 0`.
- `Analysis`: `WriteBandOut = No`.
- `ParserOptions`: `WriteHSDInput = No`.

These settings are applied automatically, even if different values appear in `dftb.dat`, to prevent shared auxiliary files from being written by independent calculators. The calculator logs remain controlled by `<dftb_lib_output>`.

PIMD input file. To use DFTB+ via the internal API, set the following keyword in `input.dat`:

```
<ipotential>
DFTBLIB
```

The printing of DFTB+ output can be controlled using:

```
<dftb_lib_output>
```

```
0
```

This keyword may take the values `-1`, `0`, or `1`:

- `-1`: suppress DFTB+ calculator logs.
- `0`: print output from the first DFTB+ calculator only.
- `1`: print output from all DFTB+ calculators.

In the serial (non-MPI) version of PIMD, options `0` and `1` both cause output to be written to the directory *001*, since all beads share a single DFTB+ instance. With **`pimd.mpi.x`** linked to a non-MPI DFTB+ library, `0` prints output only for the first calculator in *001*, whereas `1` prints output from every calculator into its numbered directory. With a non-MPI DFTB+ library, each MPI process runs an independent instance of DFTB+. With an MPI library, the processes in each force group share a collective calculation and a log. Each calculator uses the directory numbered after its first assigned bead: **`001`**, **`002`**, and so on. When a calculator cycles over several beads, their output shares that log, as with CP2K.

Execution of PIMD. Execution proceeds in the usual manner using **`pimd.x`** or **`pimd.mpi.x`**. It may be necessary to set the environment variable *OMP_NUM_THREADS* appropriately to ensure correct parallelization of the DFTB+ code. In MPI runs, the product of *OMP_NUM_THREADS* and the number of MPI processes should not exceed the number of available hardware threads.

With a non-MPI DFTB+ library, the number of MPI processes must not exceed the number of beads. With an MPI-enabled library, only the number of bead groups must not exceed the number of beads; the rank count equals <np_beads> times <np_force>. If this condition is violated, PIMD will terminate with an error. The OpenMP library assigns a DFTB+ calculation to each MPI process. The MPI library instead assigns a collective DFTB+ calculation to each force group. Group leaders contribute each bead result once to the global PIMD reductions. MPI calculator logs use numbered bead directories, and shared DFTB+ auxiliary output files listed above are disabled by both adapters.

6.3.5 DFTB - External

IMPORTANT: The user is solely responsible for all issues related to the DFTB license. PIMD does not provide a DFTB license.

The semiempirical DFTB code [55] is linked in the subroutine `force_dftb.F90` via system calls. The DFTB code must be installed before this interface can be used.

DFTB input file.. Before running the PIMD code, the DFTB+ input files [55] should be prepared. The DFTB+ input files should correspond to a static calculation of the energy and gradient for the system of interest. The order of atoms in the DFTB+ input files must match that in the PIMD input files. There are some rules for the format of the DFTB+ input file:

- The file name should be `dftb.dat`.
- "GenFormat" should be used for the specification of geometry.
- The unit should be in Å.

Here is an example of a `dftb.dat` file for a water molecule:

```
Geometry = GenFormat {
3  C
 O H
     1    1    0.00000000000E+00  -0.10000000000E+01   0.00000000000E+00
     2    2    0.00000000000E+00   0.00000000000E+00   0.78306400000E+00
     3    2    0.00000000000E+00   0.00000000000E+00  -0.78306400000E+00
}

Driver = SteepestDescent{
```

```
  MaxSteps = 0
}

Hamiltonian = DFTB {
  SCC = Yes
  SCCTolerance = 1.0e-5
  MaxSCCIterations = 1000
  Mixer = Broyden {
    MixingParameter = 0.2
  }
  SlaterKosterFiles = {
    O-O = "O-O.skf"
    O-H = "O-H.skf"
    H-O = "O-H.skf"
    H-H = "H-H.skf"
  }
  MaxAngularMomentum = {
    O = "p"
    H = "s"
  }
  Charge = 0.0
  SpinPolarisation = {}
  Filling = Fermi {
    Temperature [Kelvin] = 0.0
  }
}

Options = {}

ParserOptions = {
  ParserVersion = 3
}
```

The Slater-Koster files (`*.skf`), in this case O-O.skf, O-H.skf, and H-H.skf, should be present in the execution directory. Using this input file, one can test the DFTB execution with the following command:

```
> dftb+ < dftb.dat
```

Note that the file “detailed.out” needs to be printed (which is the default option for DFTB+) in order for the code to work. Therefore, the setting

```
Options = {
  WriteDetailedOut = No
}
```

should NOT be used in the DFTB+ input.

PIMD input file.. The following keyword should be set in the file `input.dat`.

```
<ipotential>
DFTB
```

This keyword declares the usage of the DFTB+ code.

```
<dftb_exe_command>
'dftb+'
```

This keyword sets the DFTB+ execution command.

Execution of PIMD.. The execution is done in the usual way by the command `pimd.x` or `pimd.mpi.x`. When the subroutine `force_dftb.F90` is called, the following procedure is undertaken automatically. First, `dftb.dat` in the execution directory is read, then the lines where the geometry is specified are detected and replaced by the current geometry. Then, the DFTB calculation is executed using system calls. Finally, the energy and gradient values are read from the DFTB output, and the run continues.

6.3.6 GAMESS

IMPORTANT: The user is solely responsible for all issues related to the GAMESS license. PIMD does not provide any GAMESS license.

The ab initio GAMESS code [63] is linked in the subroutine `force_gamess.F90` via system calls. The GAMESS code must be installed before this interface can be used.

GAMESS input file.. Before running the PIMD code, the GAMESS input file [63] must be prepared. The GAMESS input file should correspond to a static calculation of the energy and gradient for the system of interest. The order of atoms in the GAMESS input file must match that in the PIMD input file. There are some rules for the format of the GAMESS input file.

- The file name should be `gamess.dat`.
- There should be no symmetry in the molecule: `C1`.
- The SCF convergence should be tight enough so that the error in the energy/gradient is small: `NCONV=9`.
- The geometry lines should start from the line immediately after `C1`.
- Cartesian coordinates should be used: `COORD=UNIQUE`.
- Atomic units should be chosen: `UNITS=BOHR`.
- The gradient calculation should be included: `RUNTYPE=GRADIENT`.

Here is an example GAMESS input file for a water molecule:

```
 $CONTRL SCFTYP=RHF MULT=1 RUNTYP=GRADIENT COORD=UNIQUE UNITS=BOHR $END
 $SYSTEM TIMLIM=1 $END
 $SCF    DIRSCF=.TRUE. NCONV=9 $END
 $BASIS  GBASIS=N31 NGAUSS=6 NDFUNC=1 NPFUNC=0 $END
 $GUESS  GUESS=HUCKEL $END
 $DATA
TEST H2O MOLECULE
C1
O   8.0  0.0000000000  0.0000000000  0.0000000000
H   1.0  1.4748637392  1.0778200416  0.0000000000
H   1.0 -1.4748637392  1.0778200416  0.0000000000
 $END
```

Using this input file, one can test the GAMESS execution:

```
> cp gamess.dat gamess.inp; rungms gamess 00 1
```

PIMD input file.. The following keywords should be set in the file `input.dat`:

```
<ipotential>
GAMESS
```

This keyword declares the use of the GAMESS code.

```
<gamess_command>
'rungms gamess 00 1'
```

This keyword sets the command to execute GAMESS.

Execution of PIMD.. The execution is done in the usual way using the `pimd.x` or `pimd.mpi.x` command. When the subroutine `force_gamess.F90` is called, the following procedure is performed automatically: First, `gamess.dat` in the execution directory is read, then the lines containing the geometry are detected and replaced by the current geometry. Next, the GAMESS calculation is executed using system calls. Finally, the energy and gradient values are read from the GAMESS output, and the run continues.

6.3.7 GAUSSIAN

IMPORTANT: The user is solely responsible for all issues related to the GAUSSIAN license. PIMD does not provide any GAUSSIAN license.

The ab initio GAUSSIAN 03 (98,09,16) code [59, 60, 61, 62] is linked in the subroutine `force_g03.F90` (`force_g98.F90`, `force_g09.F90`, `force_g16.F90`) via system calls. The GAUSSIAN code must be installed before this interface can be used.

GAUSSIAN input file.. Before running the PIMD code, the GAUSSIAN input file [59, 60, 61, 62] must be prepared. The GAUSSIAN input file should correspond to a static calculation of the energy and gradient for the system of interest. There are some rules for the format of the GAUSSIAN input file.

- The file name should be `g03.dat` (`g98.dat`, `g09.dat`, `g16.dat`).
- There should be no symmetry in the molecule: `Nosymm`.
- The SCF convergence should be tight enough so that the error in the energy/gradient is small: `SCF=Tight`.
- A check point file called *chk.g03* (*chk.g98*, *chk.g09*, *chk.g16*) should be produced: `%chk=chk.g03` (`%chk=chk.g98`, `%chk=chk.g09`, `%chk=chk.g16`).
- The geometry lines should start from the second line after the second blank line.
- Cartesian coordinates should be used.
- Atomic units should be chosen: `Units=au`.
- The memory size should be chosen appropriately: `%mem`.
  NOTE: This may affect the calculation time.
- The gradient calculation should be included: `Force`.
- It is recommended to restart MO coefficients if possible: `GUESS=TCHECK`.

Here is an example input file for a water molecule called `g03.dat`:

```
%mem=20MB
%chk=chk.g03
#p HF/6-31G* SCF=TIGHT UNITS=AU NOSYMM FORCE GUESS=TCHECK

comment

0   1
8  0.0000000000  0.0000000000  0.0000000000
1  1.4891196813  1.1634278031  0.0000000000
1 -1.4891196813  1.1634278031  0.0000000000
```

NOTE: Do not forget the last blank line, as required by the GAUSSIAN input format. Using this input file, one can test the execution of g03 (g98, g09, g16) and the *formchk* program [59, 60, 61, 62]:

```
> g03 < g03.dat

> formchk chk.g03

> cat chk.fchk
```

PIMD input file.. The following keywords should be set in the file `input.dat`:

```
<ipotential>
G03
```

```
<ipotential>
G98
```

```
<ipotential>
G09
```

```
<ipotential>
G16
```

These keywords declare the use of g03, g98, g09, and g16 codes.

```
<g03_command>
'g03'
'formchk'
```

```
<g98_command>
'g98'
'formchk'
```

```
<g09_command>
'g09'
'formchk'
```

```
<g16_command>
'g16'
'formchk'
```

These keywords set the execution commands for g03, g98, g09, g16, and the *formchk* program.

Execution of PIMD.. The execution is done in the usual way using the **`pimd.x`** or **`pimd.mpi.x`** command. When the subroutine `force_g98.F90` (`force_g03.F90`, `force_g09.F90`, or `force_g16.F90`) is called, the following procedure is performed automatically: First, `g98.dat` (`g03.dat`, `g09.dat`, or `g16.dat`) in the execution directory is read, then the lines containing the geometry are detected and replaced by the current geometry. Next, GAUSSIAN is executed using *g98* (*g03*, *g09*, or *g16*) and *formchk* commands via system calls. Finally, the energy and gradient values are read from the GAUSSIAN output, and the run continues.

6.3.8 MOLPRO

IMPORTANT: The user is solely responsible for all issues related to the MOLPRO license. PIMD does not provide any MOLPRO license.

The ab initio MOLPRO code [65] is linked via the subroutine `force_molpro.F90` using system calls. The MOLPRO code must be installed before this interface can be used.

MOLPRO input file.. Before running the MOLPRO code, the MOLPRO input file [65] must be prepared. The MOLPRO input file should correspond to a static calculation of the energy and gradient for the system of interest. There are several rules regarding the MOLPRO input file format:

- The file name should be `molpro.dat`.
- Cartesian coordinates must be used by setting `geomtyp=xyz`.
- The input must be set up such that a table of potential energy is created after the keyword **<energy>**.
- The input must be set up such that a table of forces is created after the keyword **<grad>**.
- The input must be set up such that a table of dipole moments is created after the keyword **<dipole>**.

Here is an example of a MOLPRO input file for a water molecule called `molpro.dat`:

```
geomtyp=xyz
geometry={
3

O   0.0000000000   0.0000000000  -0.1302052882
H   1.4891244004   0.0000000000   1.0332262019
H  -1.4891244004   0.0000000000   1.0332262019
}
hf
e(1)=energy
force
varsav

table,e                    ! print a table with results
save,table.molpro          ! save to file
title,<energy>             ! title for the table
format,(e24.16)            ! explicit format

table,gradx,grady,gradz    ! print a table with results
save,table.molpro          ! save to file
title,<grad> 1             ! title for the table
format,(3e24.16)           ! explicit format

table,dmx,dmy,dmz          ! print a table with results
save,table.molpro          ! save to file
title,<dipole>             ! title for the table
format,(3e24.16)           ! explicit format
```

Using this input file, one can test the MOLPRO execution:

```
> molpro < molpro.dat
```

PIMD input file.. The following keywords should be set in the file `input.dat`:

```
<ipotential>
MOLPRO
```

This keyword declares the use of the MOLPRO code.

```
<molpro_command>
'molpro'
```

This keyword sets the command used to execute MOLPRO.

Execution of PIMD.. The execution is performed in the usual way using the **pimd.x** or **pimd.mpi.x** command. When the subroutine `force_molpro.F90` is called, the wrapper code **run_molpro.x** is activated (**NOTE: run_molpro.x** should be placed in a directory accessible via PATH). The following procedure is undertaken automatically: First, `molpro.dat` in the execution directory is read, then the lines containing the geometry are

detected and replaced with the current geometry. Next, MOLPRO is executed via system calls. Finally, the energy and gradient values are read from the MOLPRO output, and the run continues.

Additional setup is necessary to perform nonadiabatic dynamics, i.e., surface hopping dynamics and mean field dynamics, with the MOLPRO code. See Section 5.9.1 for details.

6.3.9 MOPAC

IMPORTANT: The user is solely responsible for all issues related to the MOPAC license. PIMD does not provide any MOPAC license.

The semi-empirical MOPAC code [54] is linked via the subroutine `force_mopac.F90` using system calls. The MOPAC code must be installed before this interface can be used.

MOPAC input file.. Before running the PIMD code, the MOPAC input file [54] must be prepared. The MOPAC input file should correspond to a static calculation of the energy and gradients (negative of the forces) for the system of interest. There are several rules regarding the MOPAC input file format:

- The file name should be `mopac.dat`.
- Perform one SCF cycle then stop: `1SCF`.
- Include gradient calculation: `GRADIENTS`.
- Do not reorient the molecule: `NOREOR`.
- Do not use molecular symmetry: `NOSYM`.
- Ensure SCF convergence is tight enough to keep the error of energy/gradient values small: `PRECISE`.
- Use Cartesian coordinates: `XYZ`.
- The geometry lines should start from the fourth line.

Here is an example of a MOPAC input file for a water molecule:

```
PM3 1SCF XYZ PRECISE GRADIENTS NOSYM NOREOR GEO-OK
molecule
comment
8  0.0000000000 1  0.0000000000 1  0.0000000000 1
1  0.7804643317 1  0.5703578412 1  0.0000000000 1
1 -0.7804643317 1  0.5703578412 1  0.0000000000 1
```

Using this input file, one can test a MOPAC calculation:

```
> nohup /opt/mopac/mopac/MOPAC2007.out mopac.dat
```

PIMD input file.. The following keywords should be set in the file `input.dat`:

```
<ipotential>
MOPAC
```

This keyword declares the use of the MOPAC code.

```
<mopac_command>
'nohup /opt/mopac/mopac/MOPAC2007.out'
```

This keyword sets the command for executing MOPAC.

Execution of PIMD.. The execution is performed in the usual way using the `pimd.x` or `pimd.mpi.x` command. When the subroutine `force_mopac.F90` is called, the following procedure is executed automatically: First, `mopac.dat` in the execution directory is read, then the lines containing the geometry are detected and replaced

with the current geometry. Next, MOPAC is executed using system calls. Finally, the energy and gradient values are read from the MOPAC output, and the run continues.

6.3.10 NTCHEM

IMPORTANT: The user is solely responsible for all issues regarding the NTChem license. PIMD does not provide any NTChem license.

The MultiNTChem code is linked internally at the subroutine `force_ntchem_MPI.F90`. To make use of the internal link, the MultiNTChem code must be installed together with the PIMD code. Note that the user must legally possess permission to use MultiNTChem. The instructions for the internal link, the setup, and the execution are shown below.

Internal link of NTChem.. Follow the instructions below (an example for the H_2O molecule).
1) Compilation.
NTChem and MNTChem are binary codes. PIMD should be linked to the library of MNTChem, either without OpenMP (libmntchem_mpi.a) or with OpenMP (libmntchem_mpiomp.a).
In the makefile, set NTCHEM = -Dntchem for the former case, NTCHEM = -Dntchem -Domp for the latter case.
For Intel computers, sometimes one needs the following for both compilation and execution:

```
> module unload mpt; module load intel/cur intel-mpi/cur
```

2) Prepare INPUT - the NTCHEM input file.
To calculate the dipole moments, set:

```
&prop
Dipole_TxtOut=T
/
```

3) INPUT.MNTChem - the MNTCHEM input file.
Be sure to set the following values correctly: NCorePerIO = n and NCorePerReplica = n, where n is the number of processors (nprocs) divided by the number of beads (nbead). Also set DoProp = T.
4) INPUT.xyz - the MNTCHEM/PIMD interface input file.

```
3
O  0.000  0.000  0.000  8.0  8.0
H  1.489  1.163  0.000  1.0  1.0
H -1.489  1.163  0.000  1.0  1.0
```

5) To secure sufficient memory space, sometimes one needs the following on execution:

```
> ulimit -s unlimited; export OMP_STACKSIZE=1G
```

Setups of NTCHEM.. In the PIMD input file, `input.dat`, set the following keywords:

```
<ipotential>
NTCHEM
```

This keyword sets the usage of NTChem.

Execution of PIMD.. The execution is performed in the usual way using the `pimd.mpi.x` command.

6.3.11 ORCA

IMPORTANT: The user is solely responsible for all issues related to the ORCA license. PIMD does not provide any ORCA license.

The ab initio ORCA code [67] is linked to the PIMD code via system calls through the subroutine `force_orca.F90`. The ORCA program must be installed in advance in order to use this interface.

ORCA input file.. Before running PIMD with ORCA, an ORCA input file [67] must be prepared. The ORCA input file should correspond to a static calculation of the total energy and nuclear gradients for the system of interest. The order of atoms in the ORCA input file must exactly match that of the PIMD input files.

The following rules apply to the ORCA input file format:

- The file name must be `orca.dat`.
- The geometry specification must start on the line immediately following `*xyz`.
- Cartesian coordinates must be used.
- Atomic units must be selected by specifying `Bohrs`.
- Gradient calculation must be enabled using `ENGRAD`.

An example ORCA input file for a water molecule is shown below:

```
! HF def2-SV(P) ENGRAD Bohrs
%elprop
 Dipole true
end
*xyz 0 1
O 0.0 0.0 0.0
H 0.0 1.0 0.2
H 0.0 0.0 1.1
*
```

Using this input file, the ORCA installation can be tested by running:

```
> cp orca.dat orca_test.inp; orca orca_test.inp
```

NOTE: The dipole moment is currently not supported in the PIMD–ORCA interface.

PIMD input file.. The following keywords must be set in the file `input.dat` to use ORCA:

```
<ipotential>
ORCA
```

This keyword activates the ORCA interface.

```
<orca_command>
'orca'
```

This keyword specifies the command used to execute ORCA.

Execution of PIMD.. The execution is performed in the usual way using `pimd.x` or `pimd.mpi.x`. When the subroutine `force_orca.F90` is invoked, the following procedure is carried out automatically: First, the file `orca.dat` in the execution directory is read. The geometry section is detected and replaced with the current atomic coordinates provided by PIMD. Next, the ORCA calculation is launched via a system call. Finally, the total energy and nuclear gradients are extracted from the ORCA output files, and the PIMD simulation continues.

6.3.12 PHASE/0

IMPORTANT: The user is solely responsible for any issues related to the PHASE/0 license. PIMD does not provide any PHASE/0 license.

The electronic structure PHASE/0 code [58] is directly linked to the PIMD code via the subroutine `force_phase0_MPI.F90` by calling the PHASE/0 libraries, `libphase.a` and `libesm.a`. The PHASE/0 libraries must be compiled beforehand to use this link. This can be done by invoking `make libphase.a libesm.a` in the directory where PHASE/0 was compiled (*src_phase* or *src_phase_3d*), for example. Please refer to the PHASE/0 documentation on compilation for further instructions.

Currently, the library version of PHASE/0 does not support stress tensor calculations. Therefore, it cannot be used for BOXOPT calculations or NPT simulations.

Linking the PHASE/0 libraries. To link to PHASE/0, a few changes are needed in the PIMD makefile around the following lines.

```
#-----------------------------------------------------------------------
#     Usage of PHASE0
#-----------------------------------------------------------------------

#     compile PIMD with PHASE/0
PHASE0 = -Dphase0

ifdef PHASE0
LIBPHASE0 = -L../lib/phase0 -lphase -lesm #-L. -lfftw3
LIBMKL = -L../lib -L$(MKLROOT)/lib/intel64 -L$(HOME)/mkl \
 -lmkl_scalapack_lp64 -lmkl_blacs_intelmpi_lp64 -lmkl_intel_lp64 \
 -lmkl_sequential -lmkl_cdft_core -lfftw3x_cdft_lp64 -lmkl_core -lm -ldl
endif
```

PHASE/0 input file.. Before running the PIMD code, prepare the PHASE/0 input files [58]. The PHASE/0 input files must correspond to a static calculation of the energy and gradient for the system of interest. The order of atoms in the PHASE/0 input files must match the order in the PIMD input files.

Here is an example for ice. The file *file_names.data* contains the following

```
&fnames
F_POT(1) = '../../../../pp/H_ggapbe_paw_nc_01m.pp'
F_POT(1) = '../../../../pp/O_ggapbe_paw_us_02m.pp'
/
```

The file *nfinp.data* contains the following:

```
Control{
  cpumax = 5 day
  condition = initial
}

accuracy{
  cutoff_wf =   25.0 rydberg
  cutoff_cd =  225.0 rydberg
  ksampling{
    method = monk ! {mesh|file|directin|gamma}
    mesh{ nx = 2, ny =  2, nz =  2 }
    kshift{ k1 = 0.5, k2 = 0.5, k3 = 0.0 }
  }
}

structure{
  unit_cell{
    a_vector = 14.36865687014571 0.0 0.0
    b_vector = -7.184328435072851 12.443621867807988 0.0
    c_vector = 0.0 0.0 13.498240952339659
  }
  atom_list{
    coordinate_system = cartesian ! {cartesian|internal}
    atoms{
    #units angstrom
    #tag   element rx       ry       rz
H 2.5154346140648998 0 1.403077739736
H -1.2577173070324492 2.1784302773389084 1.403077739736
H 2.5440658429675516 4.406451297820341 1.403077739736
H 5.0881316859351005 0 4.9745587397360005
H -2.544065842967549 4.406451297820341 4.9745587397360005
H 1.2577173070324505 2.1784302773389084 4.9745587397360005
H 3.4394504051060997 0 0.076043973452
H -1.7197252025530492 2.9786514258785615 0.076043973452
H 2.0820579474469505 3.6062301492806883 0.076043973452
H 4.164115894893899 0 3.647524973452
H -2.0820579474469487 3.6062301492806883 3.647524973452
H 1.7197252025530505 2.9786514258785615 3.647524973452
H 1.710372816004051 5.778529901873124 6.991866924814
H -2.0577569495521484 5.176842932824522 6.991866924814
H 0.3473841335481005 2.2143903156208533 6.991866924814
H 5.51215596600405 0.806351673286126 6.991866924814
H 1.7440262004478506 1.4080386423347273 6.991866924814
H 4.149167283548101 4.3704912595383965 6.991866924814
H 2.09141033399595 0.806351673286126 3.420385924814
H 5.8595400995521505 1.4080386423347273 3.420385924814
H 3.454399016451901 4.3704912595383965 3.420385924814
H -1.7103728160040483 5.778529901873124 3.420385924814
H 2.057756949552151 5.176842932824522 3.420385924814
H -0.3473841335480995 2.2143903156208533 3.420385924814
O 2.4836441033646 0 0.39756297899599996
O -1.2418220516822993 2.1508988874731676 0.39756297899599996
O 2.5599610983177015 4.433982687686083 0.39756297899599996
O 5.1199221966354 0 3.969043978996
O -2.559961098317699 4.433982687686083 3.969043978996
O 1.2418220516823006 2.1508988874731676 3.969043978996
O 5.0229539156115 0 6.63991890115
O -2.5114769578057485 4.350005692958076 6.63991890115
O 1.2903061921942507 2.2348758822011736 6.63991890115
O 2.5806123843885 0 3.06843790115
O -1.2903061921942494 2.2348758822011736 3.06843790115
O 2.511476957805751 4.350005692958076 3.06843790115
```

```
    }
  }
}
```

The atom arrangement is required by PHASE/0 for allocating arrays, among other purposes, but the actual coordinate values are passed from PIMD. Therefore, the atoms must be aligned, and the atomic species must be the same, but any coordinate and attribute values are acceptable.

PIMD input file.. The following keywords should be set in the file *input.dat*.

```
<ipotential>
PHASE0
```

This keyword declares the use of the PHASE/0 code. Another useful option is to selectively print the output of a given PHASE/0 calculation.

```
<phase0_proc>
ne nk ng
```

The three integers, `ne nk ng`, correspond to the number of band parallels, k-point parallels, and G-point parallels, respectively. In the case of a two-dimensional version, 1 should be specified as the value corresponding to `ng`. Each value should be chosen so that the MPI parallel number (`np_force`) allocated to each replica by PIMD is equal to the product `ne nk ng`.

Execution of PIMD.. The execution is done in the usual way using the *pimd.mpi.x* command.

6.3.13 QUANTUM ESPRESSO

IMPORTANT: The user is solely responsible for all issues related to the Quantum ESPRESSO (QE) license. PIMD does not provide any QE license.

The QE code is internally linked through the subroutine `force_qe_MPI.F90`. To use this internal link, QE must be installed together with the PIMD code. Note that the user must legally possess the source code of QE version 6.2.1 or 6.3 and have permission to use and modify it. Instructions for the internal link, setup, and execution are provided below.

Setup using CMake.. Currently, only QE versions 6.2.1 and 6.3 are supported by this embedded interface. Configure a QE-enabled build from the top-level PIMD directory, for example

```
> cmake -S . -B build-qe \
>   -DMKLUSE=ON \
>   -DQE=ON \
>   -DQEVERSION=6.3
```

The option `QEVERSION` selects the supported QE version and therefore the matching PIMD patch. When internet access and `wget` are available at configuration/build time, the QE source archive can be obtained by the bundled QE subbuild. For an archive downloaded in advance, keep the archive packed and add, for example,

```
> cmake -S . -B build-qe \
>   -DMKLUSE=ON \
>   -DQE=ON \
>   -DQEVERSION=6.3 \
>   -DQEFILES=/path/to/q-e-qe-6.3.zip
```

Other PIMD backend options may be added to the same configuration. For example, -DAENET=ON may be combined with the QE options when the compiler, MPI, and linked libraries are compatible. After configuration, build the selected PIMD target using the common procedure in Sections 1.2 and 1.5. Compiler and MPI selection are described once in Section 1.3 and are not repeated here. The QE subbuild is incorporated into the CMake dependency graph, so a separate generic "run make" step is not required in this backend section.

Setups of QE.. In addition to the PIMD input files (input.dat, input_default.dat, structure.dat), the QE input file (qe.dat) is also required. Refer to the QE manual for instructions on preparing these input files at https://www.quantum-espresso.org/Doc/INPUT_PW.html. In the PIMD input file, input.dat, set the following keywords:

```
<ipotential>
QE
```

This keyword enables the use of QE.

```
<qe_input_file_name>
si.md8.in
```

This keyword specifies the QE input file name.

```
<qe_output>
1 10
```

This keyword controls the printing of QE output. The first number should be set to '0' or '1' (integer). If it is set to '0', QE output is printed for the first bead only; if it is set to '1', QE output is printed for all beads. The second number specifies the print interval; in this example, output is printed every 10 steps.

For PIMD force calls, set

```
nosym = .true.
```

rather than relying on the QE default. This avoids symmetry operations that are inconsistent with instantaneous bead coordinates.

For exchange-correlation handling, the safe **pimd-prep** default is AUTO: input_dft is left unset so QE reads the functional from the pseudopotential files. All pseudopotentials in one calculation should report a mutually consistent XC family. An explicit input_dft setting overrides that metadata and should be used only with compatible pseudopotentials. **pimd-prep** 0.9.7 checks these conditions and nosym=.true. during check.

When using pimd.toml, do not repeat the UPF filenames as [files].pseudopotential, pseudopotential_2, and so on. The ATOMIC_SPECIES card is the authoritative species-to-UPF mapping; **pimd-prep check** reads it directly (and respects pseudo_dir) to verify all required pseudopotential files.

Execution of PIMD.. Execution is performed in the usual way by running the **pimd.mpi.x** command.

Internal link with QE (Old method - Expert option).. Follow the directions described below.

- **Step 1: Prepare the PIMD makefile.** Make a copy of the source directory named "compile", where the PIMD code will be compiled.

  ```
  > cp -r ~/pimd/source ~/pimd/compile
  ```

  Activate the option "QE = -Dqe" in the PIMD makefile to link QE during compilation.

  ```
  > vi ~/pimd/compile/makefile
  ```

  Note that the libraries used must be consistent with those used to compile QE (see below).

- **Step 2: Prepare PIMD modules for QE** In the `~/pimd/compile` directory, generate the modules needed to compile the modified QE.

```
> make common_variables.o
```

- **Step 3: Download the QE source code.** Go to the QE library directory (`~/pimd/lib/qe`). Download and unzip the source code for QE version 6.2.1, *q-e-qe-6.2.1*.

```
> cd ~/pimd/lib/qe
> wget https://github.com/QEF/q-e/archive/qe-6.2.1.zip
> unzip q-e-qe-6.2.1.zip
```

Make sure the list of files is as follows:

```
> ls
README                    make_patch_qe.sh*         qe-6.2.1.zip
apply_patch_qe.sh*        q-e-qe-6.2.1/             qe-6.2.1_to_pimdlib.patch
```

Also ensure that an FFTW library (fftw3 or equivalent) required by QE is available (see below).

- **Step 4: Apply a patch to the QE code.** In the QE library directory, apply the patch file as follows:

```
> patch -p0 < qe-6.2.1_to_pimdlib.patch
```

The QE source code will be modified to link with PIMD.

- **Step 5: Prepare the QE makefile.** Configure QE in the *q-e-qe-6.2.1* directory.

```
> cd ~/pimd/lib/qe/q-e-qe-6.2.1
> ./configure
```

- **Step 6: Compile QE** Go to the QE directory and compile it.

```
> cd ~/pimd/lib/qe/q-e-qe-6.2.1
> make pw
```

- **Step 7: Compile and link PIMD and QE codes.** Go to the PIMD compile directory and edit the makefile as needed:

```
> vi ~/pimd/compile/makefile
```

After editing the makefile, compile the code.

6.3.14 OpenMX 4.0.1

OpenMX [57] can be used as an embedded electronic-structure backend in PIMD 2.8.0. The interface is designed for OpenMX 4.0 with the official 4.0.1 patch applied. OpenMX remains responsible for the electronic structure calculation, whereas PIMD controls the nuclear coordinates, thermostats, and, when applicable, the simulation cell.

The OpenMX source code is not distributed with PIMD. Download OpenMX from the official OpenMX website and follow the OpenMX license and installation requirements.

Installing and patching OpenMX.. First download the official OpenMX 4.0 source archive and the official `patch4.0.1.tar.gz`. The OpenMX 4.0.1 patch must be applied before the PIMD patch. A convenient directory layout is

```
PIMD/
  lib/openmx/
```

```
    openmx4.0/
      source/
      work/
      DFT_DATA19/
```

From the PIMD source directory, the preparation can be performed as follows:

```
cd lib/openmx
tar xzf /path/to/openmx4.0.tar.gz
cp /path/to/patch4.0.1.tar.gz .
./apply_official_4.0.1_patch.sh ./openmx4.0 ./patch4.0.1.tar.gz
./apply_pimd_patch.sh ./openmx4.0
```

The PIMD patch extends the OpenMX API with persistent initialization, energy/force evaluation, and finalization routines. It does not replace the standard OpenMX build configuration.

Edit `openmx4.0/source/Makefile` for the target computer in the same way as for a normal OpenMX build. Then build the OpenMX API library with

```
./build_api.sh ./openmx4.0
```

or, equivalently,

```
cd openmx4.0/source
make api
```

This produces `openmx4.0/source/libopenmx_api.a`.

Building PIMD with OpenMX.. The recommended build uses CMake. PIMD and OpenMX must use compatible MPI and compiler families. For example,

```
export CC=mpicc
export CXX=mpicxx
export FC=mpifort

cmake -S . -B build-openmx \
  -DOPENMX=ON -DOPENMP=ON \
  -DOPENMX_DIR="$PWD/lib/openmx/openmx4.0/source" \
  -DOPENMX_EXTRA_LIBS="<external libraries used by OpenMX>"
```

The global `OPENMP` CMake option defaults to OFF. Enable it with `-DOPENMP=ON` when building PIMD with OpenMX.

`OPENMX_EXTRA_LIBS` must contain the external libraries required by the OpenMX build, such as the ScaLAPACK/BLACS or MKL libraries and FFTW when they are present in the OpenMX `LIB` line. If `libopenmx_api.a` is kept outside `OPENMX_DIR`, its location can be given explicitly with `OPENMX_API_LIBRARY`.

Build **`pimd.mpi.x`** with the common target procedure in Section 1.5. The same OpenMX-enabled CMake tree can also build `polymers.x`. The serial executable `pimd.x` intentionally does not provide the embedded OpenMX backend; use `pimd.mpi.x` even for a one-MPI-rank test.

For the traditional Makefile build, enable

```
OPENMX = -Dopenmx
```

in `source/makefile`. The default layout assumes

```
OPENMXROOT = ../lib/openmx/openmx4.0
```

```
OPENMXDIR  = $(OPENMXROOT)/source
LIBOPENMX  = $(OPENMXDIR)/libopenmx_api.a
```

and OPENMXLIBS should reproduce the external library dependencies of the OpenMX build.

OpenMX input file.. Prepare a standard OpenMX input file named openmx.dat in the PIMD run directory before starting PIMD. PIMD copies this file into the bead directories when the OpenMX backend is initialized and creates one persistent OpenMX instance for each bead. The number of atoms, their order, and their species assignments in openmx.dat must agree with the PIMD structure; subsequent coordinates are supplied by PIMD, whereas the OpenMX species and basis definitions remain those established during initialization. OpenMX should be configured as an electronic-structure evaluator rather than as an independent molecular-dynamics driver. For example,

```
MD.Type       NOMD
MD.maxIter    1
scf.restart   off
```

For force and virial validation, scf.restart alloff and sufficiently tight SCF and numerical-grid settings are recommended.

Use an absolute DATA.PATH unless the relative path is deliberately constructed for the bead directories 001/, 002/, and so on. For an isolated calculation (<iboundary> = 0), the cell initialized by OpenMX is retained when PIMD does not supply a valid cell. For periodic or variable-cell calculations, PIMD supplies its cell to OpenMX on every force call.

The unit contract of the embedded interface is

coordinates and cell	bohr
energy	hartree
force	hartree/bohr
stress	hartree/bohr^3
OpenMX molecular dipole	Debye (converted by PIMD to e bohr)

No eV/Å conversion should be introduced in the PIMD–OpenMX interface. The dipole exposed by the present bridge is the ordinary OpenMX molecular dipole. Periodic Berry-phase polarization and polarization-branch unwrapping are not implemented by this interface.

PIMD input and parallel execution.. Select the OpenMX backend with

```
<ipotential>
OPENMX
```

The present implementation assigns one persistent OpenMX MPI subgroup to each bead and therefore requires

```
<np_beads>
<nbead>
```

i.e., numerically np_beads = nbead. The keyword <np_force> sets the number of MPI ranks used by OpenMX for each bead. The current Fortran adapter initializes OpenMX with one OpenMP thread per MPI rank, so parallelism in this interface is presently controlled by the MPI subgroup size. For example, 32 beads with 8 ranks per OpenMX calculation require 256 MPI ranks.

Run the calculation with the MPI executable, for example,

```
mpirun -np 256 ./pimd.mpi.x
```

PIMD keeps its own standard output separate from the OpenMX console stream. The root rank of each OpenMX subgroup writes `001/openmx.out`, `002/openmx.out`, and so forth. If an OpenMX build writes from non-root ranks, the additional streams are written as `openmx.rankNNNN.out` in the corresponding bead directory.

6.3.15 MBX

MBX supplies molecular energies, forces, virials, and dipoles to PIMD. MBX is installed separately and is not distributed with PIMD. PIMD does not provide a license to use MBX. See `https://paesanigroup.ucsd.edu/software/mbx.html` and the MBX papers listed there [93, 94]. Select `<ipotential> = MBXLIB`.

Build and parallel layout.. Build MBX and its FFTW and GSL dependencies first. Its installation prefix must contain `lib/pkgconfig/mbx.pc`. Configure from the PIMD root:

```
cmake -S . -B build-mbx -DMBX=ON -DMBX_DIR=/path/to/mbx/install \
  -DBUILD_TESTING=ON
cmake --build build-mbx --target pimd.mpi.x --parallel 4
```

Use the same compatible compiler and MPI installation for PIMD and MBX. The serial **`pimd.x`** does not contain the MBX backend. Set `-DMBX_MPI=ON` if the installed MBX library was built with MPI support. This setting matches the library interface but does not currently enable multiple MPI ranks to evaluate one bead. If MBX is compiled with OpenMP support, configure a compatible OpenMP build with `-DOPENMP=ON`.

Validation.. With `BUILD_TESTING=ON`, inspect and run the MBX checks using:

```
ctest --test-dir build-mbx -N -L mbx
ctest --test-dir build-mbx -L mbx --output-on-failure
```

The tests cover finite-difference forces and virials for isolated and periodic water, including PME cases. They require an MBX installation and an MPI launcher.

Input files and keywords.. Keep the PIMD structure, MBX JSON configuration, and monomer list together in the run directory, or provide paths for the latter two files. The three MBX-specific `input.dat` tags and their defaults are:

```
<ipotential>
MBXLIB
<mbx_json_file>
mbx.json
<mbx_input_file>
mbx.dat
<iprint_mbx_mol_dipoles>
0
<np_force>
1
```

The JSON file is an MBX-native configuration with a top-level `MBX` object. Its parameters and model choice follow the installed MBX version. The monomer file begins with the number of monomers, followed by that many monomer names and then that many positive atom counts. For three water molecules:

```
3
h2o h2o h2o
3 3 3
```

The atom counts must sum to the number of atoms in the PIMD structure; monomer and atom order must agree with that structure.

The TOML front end writes these tags from the following settings:

```
[potential]
type = "MBXLIB"

[parallel]
force = 1

[mbx]
json_file = "mbx.json"
monomer_file = "mbx.dat"
molecular_dipole_output_interval = 0
```

Use `pimd-prep init PIMD MBX` for an interactive starting point. `pimd-prep check` checks that both MBX files exist, that the JSON parses with an `MBX` object, and that the monomer list has the declared number of names and positive atom counts. It also checks that the atom-count sum matches the PIMD structure. The MBX bridge repeats these checks at runtime. The complete one-bead ice Ih NVT example is in **`examples/02_path_integral_dynamics/ice_ih/pimd_nvt_mbx`**. It contains `pimd.toml`, `structure.dat`, `mbx.json`, and `mbx.dat`. From the PIMD root, copy the files into a scratch directory and run them with the MBX-enabled executable:

```
run_dir=$(mktemp -d)
cp -R examples/02_path_integral_dynamics/ice_ih/pimd_nvt_mbx/. "$run_dir/"
PIMD_MPI_EXE="$PWD/build-mbx/pimd.mpi.x" \
  examples/run_example.sh mpi "$run_dir"
```

Molecular dipoles.. Set **`<iprint_mbx_mol_dipoles>`** to a positive step interval to append `mbx_molecular_dipoles.out`. Each row gives the step, bead and monomer indices, followed by the Cartesian permanent, induced, and total dipoles in atomic units (e bohr). A value of 0 produces no per-monomer dipole file.

6.3.16 TURBOMOLE

IMPORTANT: The user is solely responsible for all issues related to the TURBOMOLE license. PIMD does not provide any TURBOMOLE license.

The ab initio TURBOMOLE code [64] is linked to the subroutine `force_turbo.F90` via system calls. TURBOMOLE must be installed before this interface can be used.

TURBOMOLE input file.. Before running the PIMD code, the TURBOMOLE input files [64] should be prepared. The TURBOMOLE input files should correspond to a static calculation of the energy and gradient for the system of interest. The order of atoms in the TURBOMOLE input files must match that of the PIMD input files. In the TURBOMOLE code, the energy and the gradient are computed by two consecutive commands (`dscf`,`grad`), (`ridft`,`rdgrad`), (`dscf`,`mpgrad`), (`dscf`,`rimp2`), or (`dscf`,`ricc2`) [64]. Prepare the `control` file properly, as well as all other necessary files (`coord`, `mos`, `basis`, `auxbasis`, etc.). All of these can be obtained using the *define* command of the TURBOMOLE code [64]. Here is an example of the TURBOMOLE `control` file for a water molecule:

```
> cat control

$title
water molecule
$operating system unix
$symmetry c1
$coord    file=coord
$user-defined bonds    file=coord
$atoms
```

```
o  1                                                                          \
   basis =o def-SV(P)
h  2-3                                                                        \
   basis =h def-SV(P)
$basis    file=basis
$rundimensions
   dim(fock,dens)=211
   natoms=3
   nshell=10
   nbf(CAO)=19
   nbf(AO)=18
   dim(trafo[SAO<-->AO/CAO])=21
   rhfshells=1
$scfmo   file=mos
$closed shells
 a       1-5                                    ( 2 )
$scfiterlimit       30
$scfconv        8
$thize     0.10000000E-04
$thime        5
$scfdamp   start=0.300  step=0.050  min=0.100
$scfdump
$scfintunit
 unit=30       size=0        file=twoint
$scfdiis
$scforbitalshift  automatic=.1
$drvopt
   cartesian  on
   basis      off
   global     off
   hessian    on
   dipole     on
   nuclear polarizability
$interconversion  off
   qconv=1.d-7
   maxiter=25
$optimize
   internal   off
   cartesian  on
   global     off
   basis      off   logarithm
$coordinateupdate
   dqmax=0.3
   interpolate  on
   statistics    5
$forceupdate
   ahlrichs numgeo=0  mingeo=3 maxgeo=4 modus=<g|dq> dynamic fail=0.3
   threig=0.005  reseig=0.005  thrbig=3.0  scale=1.00  damping=0.0
$forceinit on
   diag=default
$energy    file=energy
$grad    file=gradient
$forceapprox    file=forceapprox
$lock off
$end
```

There are some rules regarding the format of TURBOMOLE `control` files.

- There should be no symmetry in the molecule.
- The SCF convergence should be tight enough so that the error in the energy and gradient values is small.

Let us give an example of the (`dscf`,`grad`) combination. One can test the TURBOMOLE execution of the `dscf` command with the `control` file as well as all other input files.

```
> dscf
```

If it works correctly, copy the `control` file to the *control.1* file. Then test the TURBOMOLE execution of the `grad` command using the `control` file as well as all other input files.

```
> grad
```

If it works correctly, copy the `control` file to the *control.2* file.

PIMD input file.. The following keywords should be set in the file `input.dat`.

```
<ipotential>
TURBOMOLE
```

This keyword declares the use of the TURBOMOLE code.

```
<turbo_command>
      dscf   control.1
      grad   control.2
```

This keyword sets the TURBOMOLE commands together with the control files. In the present case, the TURBOMOLE commands consist of the `dscf` and `grad` commands, and the names of the control files are *control.1* and *control.2*, respectively.

```
<turbo_guess>
PREVIOUS
```

This keyword specifies the option for the initial guess of molecular orbital coefficients. In the present case, those from the previous step are reused.

NOTE: Runtime-file diagnostics follow the effective **<turbo_command>**. The two control files explicitly named by its command lines (for example, `control.1` and `control.2`) are conditionally required. A pre-existing `coord` is also required as a TURBOMOLE prototype. At initialization, PIMD copies it to `coord.turbo`; before each force calculation, PIMD rewrites the `$coord` geometry block using the current PIMD coordinates while preserving the remaining prototype content. Thus the coordinates in the prototype are not authoritative, but the file itself must exist. The optional `turbo.dat` is only a secondary keyword source when the corresponding setting is absent from `input.dat`; it is not an unconditional runtime requirement.

Execution of PIMD.. The execution is carried out in the usual way using the **pimd.x** or **pimd.mpi.x** command. When the subroutine `force_turbo.F90` is called, the following procedure is performed automatically. First, the existing TURBOMOLE `coord` prototype is copied and its geometry block is rewritten using the current PIMD geometry. Then, the TURBOMOLE calculation is executed using system calls. Finally, the energy and gradient values are read from the TURBOMOLE output, and the run is continued. **NOTE:** When the subroutine `force_turbo.F90` is called, this keyword is searched primarily in the `input.dat` file, secondarily in the `turbo.dat` file, and finally in the `input_default.dat` file.

6.3.17 VASP6 Interface

IMPORTANT: The user is solely responsible for all issues related to the VASP6 license. PIMD does not provide a VASP license.

Linking the VASP6 code. Before configuring PIMD, compile the patched VASP6 source. The patch supplied under `lib/vasp6` currently supports VASP 6.4.0. PIMD does not provide the VASP source or a VASP compiler setup; follow the VASP documentation for the VASP build itself. Once the patched VASP build is available, configure PIMD from its top-level directory. For example,

```
> cmake -S . -B build-vasp6 \
>   -DVASP6=ON \
>   -DVASP6_DIR="$PWD/lib/vasp6/vasp.6.4.0.modified/build/std"
```

`VASP6_DIR` must identify the VASP build directory containing the module files and libraries required by the embedded interface. If that VASP build uses HDF5 and an explicit include path is required, also provide `-DHDF5_INC=/path/to/hdf5/include`. Build the configured PIMD tree using Sections 1.2 and 1.5. Compiler/MPI compatibility with VASP is governed by the common toolchain guidance in Section 1.3; those generic steps are not duplicated here.

VASP6 files.. Please refer to the VASP manual for details on how to set up the files mentioned here. The current interface requires `INCAR`, `KPOINTS`, and `POTCAR` in the main calculation directory. A pre-existing `POSCAR` is *not* a required runtime input for this embedded interface. During initialization, `force_vasp6_MPI.F90` writes a fresh `POSCAR` from the current PIMD bead coordinates and simulation cell and then copies `INCAR`, `KPOINTS`, the generated `POSCAR`, and `POTCAR` into the numbered directory for each bead. Consequently, the geometry and cell in the PIMD input are authoritative. A user-supplied `POSCAR` may still be useful before execution as a `pimd-prep` consistency/template check, but it is overwritten by the embedded VASP6 initialization and must not be treated as a missing-file blocker. The standard VASP output files for each bead calculation are also written in these directories. Finally, a file named `VASPSTDOUT` in each bead directory contains the output that VASP would normally write to standard output.

The embedded VASP6 interface uses the PIMD-specific ionic branch introduced by the supplied VASP patch. Therefore the following setting in `INCAR` is mandatory:

```
IBRION = 11
```

The interface checks this value after VASP initialization and aborts when a different `IBRION` is used. This must not be replaced by ordinary VASP relaxation settings such as `IBRION = 1` or `IBRION = -1`. For PIMD force calls, symmetry should also be disabled,

```
ISYM = 0
```

and `ISIF = 3` is recommended when stress is required, especially for variable-cell simulations. The VASP `NSW` value is not the number of PIMD steps; nuclear dynamics are driven by PIMD. It is also recommended to set `LCHARG = .FALSE.` to avoid unnecessary charge-density files. `pimd-prep` 0.9.7 writes/checks the mandatory force-call settings and checks a user-supplied POSCAR against POTCAR/`structure.dat` when it is available; absence of POSCAR is non-blocking because PIMD generates it.

PIMD input file.. The following keywords should be set in the file `input.dat`.

```
<ipotential>
VASP6
```

This keyword declares the use of the VASP6 interface.

6.3.18 VASP - General Words of Caution

IMPORTANT **NOTE:** The PIMD developers cannot provide specific support for VASP calculations. The advice given here is based on their experience and may not be sufficient in all cases.

As with all large-scale calculations, we recommend performing a test run, either as a single-point energy calculation or as a structural relaxation. A non-exhaustive list of things to check includes:

- Ensure that VASP is compiled correctly and that the executable is in your PATH.
- Ensure that the VASP input files are set up correctly.
- Ensure that the VASP output files are parsed correctly by the PIMD code.

One problem that can occur is that the order of ions in the POSCAR file does not match the order of ions in the POTCAR file. VASP may still be able to run the calculation, but the resulting relaxed structures can deviate significantly from what is expected. It is recommended to set *LCHARG = .FALSE.* to prevent VASP from writing charge-density output, since trajectory output is better handled through PIMD's internal trajectory output functions.

6.3.19 VASP5

IMPORTANT: The user is solely responsible for any issues related to the VASP license. PIMD does not provide any VASP license.

The VASP code is linked externally to the subroutine `force_vasp5.F90` of the PIMD code. VASP must be installed before the external interface can be used.

The VASP code is linked internally to the subroutine `force_vasp_MPI.F90`. To make use of the internal link, the VASP code must be installed together with the PIMD code. Note that the user must legally possess the source code of VASP version 5.3.5, as well as permission to use and modify it. The directions for the internal link, the setup, and execution are shown below.

Internal link of VASP.. Follow the directions below:

- **Step 1: Copy the source code.** Copy the source code of VASP version 5.3.5 and its library to the lib directory of PIMD.

```
> cp -r vasp.5.3.5 lib/vasp
> cp -r vasp.5.lib lib/vasp
```

- **Step 2: Compile the VASP library.** Starting from the main directory, go to the vasp.5.lib directory.

```
> cd lib/vasp/vasp.5.lib
```

Then, edit the Makefile appropriately with respect to the Fortran compiler, compilation options, and the LAPACK/BLAS libraries, depending on the machine.

```
> cp makefile.*** Makefile
> vi Makefile
```

Compile the VASP library.

```
> make
```

If successful, a new file called `libdmy.a` will be created. Copy it to the lib directory.

```
> cp libdmy.a ../../../lib
```

Return to the main directory.

```
> cd ../../../lib
```

- **Step 3: Compile the modified VASP code.** Go to the vasp directory.

```
> cd lib/vasp
```

Apply the patch to modify the VASP source:

```
> patch -u -p1 -d vasp.5.3.5/ < vasp.5.3.5_to_pimdlib.patch
```

If successful, the following output will appear:

```
patching file acfdt.F
patching file aedens.F
...
patching file xml.F
patching file zgemmtest.F
```

without any error message. Go to the vasp.5.3.5 directory.

```
> cd vasp.5.3.5
```

Then, edit the Makefile appropriately with respect to the Fortran compiler, compilation options, and the LAPACK/BLAS libraries, depending on the machine.

```
> vi Makefile
```

Here, do not overwrite the Makefile with the original makefile.***, because the Makefile is created by the patch. Compile the modified VASP code.

```
> make
```

If successful, a new file called `libvasp.a` will be created. Copy it to the lib directory.

```
> cp libvasp.a ../../../lib
```

Return to the main directory.

```
> cd ../../../
```

- **Step 4: Compile the PIMD code.** To compile the PIMD code, edit the `makefile` to activate the `-Dvasp` option.

```
> cp -r source compile
> cd compile
> vi makefile
> make
```

If successful, this generates **`pimd.mpi.x`**, an executable of PIMD internally linked to VASP.

Setups of VASP.. In addition to the PIMD input files (`input.dat`, `input_default.dat`, `structure.dat`), the VASP input files are necessary (`INCAR`, `POSCAR`, `POTCAR`, `KPOINTS`). See the VASP manual for the preparation of these input files. In the `INCAR` file, the following keywords must always be set:

```
IBRION = 1
NSW = 100
ISIF = 3
```

```
ISYM = 0
```

On the other hand, in `input.dat`, the following keyword sets the use of VASP:

```
<ipotential>
VASP5
```

for the external link, and

```
<ipotential>
VASP
```

for the internal link.

```
<vasp_output>
0 10
```

This keyword controls the VASP output. The first number should be set to '0' or '1' (integer). The VASP output is printed for the first bead only if '1', and for all beads if '0'. The second number is the print interval, in the present case, every 10 steps.

```
<vasp_reuse_wavefunction>
1
```

This keyword controls the initial guess for the Kohn–Sham orbitals. The Kohn–Sham orbitals are built from scratch if '0', and taken from the previous step if '1'.

Execution of PIMD.. PIMD can be executed using the command `pimd.x` or `pimd.mpi.x`.

6.4 Machine learning potentials

The PIMD code can also be combined with several types of machine learning interatomic potentials. These potentials enable efficient large-scale simulations with near *ab initio* accuracy, once properly trained. Currently, the following machine learning potentials are supported:

- AccelNet [86] (inference using supported AENET or N2P2 models; Section 6.4.2).
- AENET [87] (artificial neural network potential).
- ASE calculator interface (generic Python/ASE calculator bridge).
- MACE [92] (equivariant message-passing neural network potential).
- MTP [90, 91] (moment tensor potential).
- N2P2 [95] (high-dimensional neural network potential).

Each machine learning potential has its own requirements for training data, parameter files, and execution workflow. Details of the setup, input files, and execution procedures for each potential are described in the corresponding subsections.

6.4.1 AENET

IMPORTANT: The user is solely responsible for all issues related to the AENET license. PIMD does not provide any AENET license.

The artificial neural network code AENET [87] is linked via the subroutines `force_aenet.F90` and `force_aenet_MPI.F90`. To use this link, the AENET code must be installed properly.

XMPI. AENET force inference is supported with <xmpi>="ON". The historical on-the-fly AENET training workflow is not XMPI-enabled; use ordinary MPI when <istep_train_aenet>> 0. XMPI requests with AENET training enabled terminate explicitly rather than silently skipping training.

User-facing AENET configuration with pimd-prep. The TOML interface separates three different roles that were historically mixed together: general PIMD files, AENET model files, and AENET training controls. General run files remain in [files]. AENET networks and species-specific data belong under [aenet.species.N]; on-the-fly training controls belong under [aenet.training].

For prediction only, the input is intentionally short:

```
[aenet]
number_of_species = 2

[aenet.species.1]
name    = "Si"
network = "Si.15.ann"

[aenet.species.2]
name    = "O"
network = "O.15.ann"
```

The value of number_of_species must match the contiguous numbered tables [aenet.species.1] through [aenet.species.N]. pimd-prep converts these entries to the legacy <ntype_aenet> block.

For an SLHMC generate/train workflow, add only the information that is needed for training:

```
[aenet]
number_of_species = 2

[aenet.species.1]
name              = "Si"
network           = "Si.15.ann"
fingerprint       = "Si.fingerprint.stp"
reference_energy  = "0.0 eV"
hidden_layers     = ["15:tanh"]

[aenet.species.2]
name              = "O"
network           = "O.15.ann"
fingerprint       = "O.fingerprint.stp"
reference_energy  = "0.0 eV"
hidden_layers     = ["15:tanh"]

[aenet.training]
training_set             = "SiO2.train"
structure_output_interval = 1
training_interval        = 100
min_training_structures  = 50
iterations               = 500
max_energy               = "1.0 eV"
```

The structured interface uses test_percent=10, optimizer="bfgs", and timing output by default, so these routine settings need not be repeated in every example. Here fingerprint and reference_energy are used to build the AENET TYPES and SETUPS sections, while hidden_layers builds the NETWORKS section. The training table defines when retraining occurs and the common generate.x/ train.x settings. Thus species names, network files, and fingerprint files are written only once in the TOML input.

pimd-prep automatically generates the legacy <generate_aenet> and <train_aenet> blocks. The older flat

TOML keys `types_and_networks`, `generate_input`, and `train_input` remain accepted for compatibility, but should not be used in new inputs or distributed examples.

Internal link of AENET. An alternative PIMD-compatible CMake wrapper can be selected with `-DAENET_SOURCE_DIR=/path/` the default is `lib/aenet`.

For a new installation, use the bundled CMake integration. It prepares the supported AENET source/library subbuild and enables the PIMD interface with

```
> cmake -S . -B build-aenet -DAENET=ON
```

Then build the required PIMD target as described in Sections 1.2 and 1.5. The CMake command above is the AENET-specific part of the build; compiler, MPI, and common math library selection are documented in the central compilation section. The manual makefile-oriented procedure below is retained for compatibility with older installations in which the AENET sources and static libraries are prepared by hand.

- **Step 1: Copy the source code.** First, copy the source code of AENET version 2.0.3 and its libraries to the PIMD library directory.

  ```
  > cp -r aenet-2.0.3/src lib/aenet
  > cp -r aenet-2.0.3/lib lib/aenet
  ```

- **Step 2: Compile the AENET library.** Starting from the main directory, move to the L-BFGS library directory.

  ```
  > cd lib/aenet/lib
  ```

  Edit the Makefile as appropriate for your Fortran compiler.

  ```
  > vi Makefile
  ```

  Compile the AENET library.

  ```
  > make
  ```

  If successful, a file named `liblbfgsb.a` will be created. Copy it to the *lib* directory.

  ```
  > cp liblbfgsb.a ../../
  ```

  Return to the main directory.

  ```
  > cd ../../../
  ```

- **Step 3: Compile the modified AENET code.** Move to the AENET directory.

  ```
  > cd lib/aenet
  ```

  Create a copy of the source directory and apply the patch.

  ```
  > cp -r src src_modified
  > patch -u -p1 -d src_modified < src.patch
  ```

  Move to the modified source directory.

  ```
  > cd src_modified
  ```

  Edit the Makefile options according to your system (Fortran compiler, LAPACK/BLAS libraries, etc.).

```
> vi makefiles/Makefile.options
```

Compile the modified AENET code.

```
> make -f makefiles/Makefile.options lib
```

If successful, a file named `libaenet.a` will be created. Copy it to the *lib* directory.

```
> cp libaenet.a ../../
```

Return to the main directory.

```
> cd ../../../
```

- **Step 4: Compile PIMD with the legacy makefile.** Edit the `makefile` to activate the options `-Daenet2 -Dnlist`.

```
> cp -r source compile
> cd compile
> vi makefile
> make
```

If successful, the legacy procedure generates **`pimd.mpi.x`**, an executable of PIMD internally linked to AENET.

PIMD input file to create xsf files.. Use the PIMD, PIHMC, or REHMC method to create a set of xsf files. Set the following keyword in `input.dat`:

```
<iprint_xsf_aenet>
10
```

This keyword specifies the interval for printing AENET xsf files, in this case, every 10 steps.

If the PIHMC or REHMC method is used, the following keyword may also be set:

```
<ioption_xsf_aenet>
1
```

With this option, xsf files are printed for both accepted and rejected structures. The xsf files are created in the directory named *structures*.

PIMD input file to generate/train using xsf files.. Use the PIMD, PIHMC, or REHMC method to generate training data and train AENET networks using the xsf files. Set the following keywords in `input.dat`:

```
<istep_train_aenet>
100
```

This keyword specifies the interval for generating and training AENET. In this example, training is performed every 100 steps.

```
<lstep_train_aenet>
3000
```

This keyword sets the final step for generating and training AENET. Here, training stops at step 3000.

```
<minxsf_train_aenet>
```

```
300
```

This keyword specifies the minimum number of xsf files required to start training. Training is skipped until at least 300 xsf files are found in the *structures* directory.

```
<generate_aenet>
OUTPUT SiO2.train
TYPES
2
Si -5.420474   | eV
O  -4.939198   | eV
SETUPS
Si  Si.fingerprint.stp
O   O.fingerprint.stp
```

This block specifies the input for generating the AENET training set.

```
<train_aenet>
TRAININGSET SiO2.train
TESTPERCENT 10
ITERATIONS  5000
MAXENERGY 1.0
TIMING
METHOD
bfgs
NETWORKS
! atom   network         hidden
! types  file-name       layers  nodes:activation
Si     Si.15-15.ann    2      15:tanh 15:tanh
O      O.15-15.ann     2      15:tanh 15:tanh
```

This block specifies the training parameters for AENET.

```
<dir_train_aenet>
"trained_networks"
```

This keyword specifies the scratch directory for trained networks. After training is completed, the final network files are copied to the execution directory.

PIMD input file to run simulations (predict).. Set the following keywords in `input.dat`:

```
<ipotential>
AENET
```

This keyword specifies the use of the AENET potential.

```
<ntype_aenet>
2
Si  Si.15-15.ann
O   O.15-15.ann
```

This keyword defines the number of atomic types and the corresponding AENET network for each type.

PIMD input file to run simulations (predict/generate/train).. In the PIHMC or REHMC method with a dual potential, the generate/train process can be performed on the fly when AENET is used as the low-level po-

tential. For example:

```
<method>
PIHMC

<ipotential>
DUAL

<dual_potential>
VASP AENET
```

The following keywords are useful:

```
<istep_hmc>
8

<istep_max_hmc>
128

<istep_adjust_hmc>
100

<istep_mul_hmc>
2

<ratio_min_hmc>
0.05

<ratio_max_hmc>
0.20
```

These keywords define a variable number of MD steps per Metropolis step. In this example, the number of MD steps varies from 8 to 128 with a geometric progression of ratio 2. Every 100 Metropolis steps, the number of MD steps is increased if the acceptance ratio is below 0.05, and decreased if it is above 0.20.

Execution of PIMD.. The execution is performed in the usual way using the `pimd.x` or `pimd.mpi.x` command. When the subroutine `force_aenet.F90` is called, the input file *predict.in* is automatically generated.

6.4.2 AccelNet

AccelNet [86] evaluates energies, forces, and the full nine-component virial from supported AENET networks or N2P2 model directories. Select `<ipotential>` = `ACCELNET` to use the interface in `force_accelnet_MPI.F90`. The model format and the evaluation algorithm are selected independently. Source code and upstream documentation: `https://github.com/cometscome/AccelNet`. When publishing calculations using this interface, cite the AccelNet preprint [86] and the relevant model/descriptor publications, including AENET [87] or N2P2 [95] as applicable.

Method and advantages. The preprint [86] describes acceleration without retraining: descriptor definitions, scaling, and network weights are preserved, with energies and analytical forces agreeing to roundoff in its tested models. Its Cartesian moment factorization speeds up angular-descriptor evaluation while preserving the trained potential. Existing supported AENET and N2P2 models can therefore be reused directly with an accelerated inference engine.

CMake build and dependency updates. Use the common compiler, MPI, and BLAS/LAPACK setup in Sections 1.1–1.4. This interface requires AccelNet 1.1.1 or later. AccelNet additionally requires Git and network access for the initial clone. From the top-level PIMD source directory, configure and build with

```
> cmake -S . -B build-accelnet \
    -DCMAKE_Fortran_COMPILER=gfortran -DCMAKE_BUILD_TYPE=Release \
    -DACCELNET=ON -DACCELNET_PROVIDER=FETCH -DBUILD_TESTING=OFF
> cmake --build build-accelnet --target pimd.mpi.x --parallel 8
```

The resulting executable is **build-accelnet/pimd.mpi.x**. The **polymers.x** target also links the adapter and can be built explicitly. Use **pimd.mpi.x**, including for one-process runs; the serial **pimd.x** target does not include this backend. Neither `-DAENET=ON` nor `-DN2P2=ON` is needed to read those model formats through AccelNet. These options enable the separate original backends.

CMake clones AccelNet into **build-accelnet/_deps/accelnet-src** and follows `origin/main`. Re-run the configuration command to fetch the latest main branch, then rebuild. The build command alone does not check for upstream updates. CMake prints the checked-out commit; record it with calculation results. The default `ACCELNET_PROVIDER=FETCH` builds CPU shared libraries with the PIMD Fortran compiler. PIMD uses `ISO_C_BINDING` to call the batched C API for both CPU and GPU; it does not import AccelNet's compiler-specific Fortran modules. AccelNet 1.1.1 includes the required `AccelNetC` CMake package, `TargetC` install component and C constructors for N2P2 directories and versioned Chebyshev models. No local AccelNet modifications are needed. The CPU build does not require NVHPC or a GPU.

For an already populated build directory, disable updates for offline rebuilds:

```
> cmake -S . -B build-accelnet \
    -DFETCHCONTENT_UPDATES_DISCONNECTED=ON
```

Set this option back to `OFF` to resume fetching updates. To reproduce the official 1.1.1 release independently of future main updates, clone its tag and select that checkout:

```
> git clone --branch 1.1.1 --depth 1 \
    https://github.com/cometscome/AccelNet.git /path/to/AccelNet-1.1.1
> cmake -S . -B build-accelnet-1.1.1 -DACCELNET=ON \
    -DCMAKE_Fortran_COMPILER=gfortran -DCMAKE_BUILD_TYPE=Release \
    -DBUILD_TESTING=OFF \
    -DFETCHCONTENT_SOURCE_DIR_ACCELNET=/path/to/AccelNet-1.1.1 \
    -DACCELNET_DESCRIPTORS_SOURCE_DIR=/path/to/AccelNet-1.1.1/AccelNetDescriptors
> cmake --build build-accelnet-1.1.1 --target pimd.mpi.x --parallel 8
```

This override uses the specified source tree without automatically updating it. Use a fresh build directory when changing source trees and set both source paths, because the descriptor path is cached separately.

GPU library with a separate compiler. Only AccelNet needs to be compiled with NVHPC; PIMD can keep gfortran and its usual MPI installation. Put **nvfortran** from the NVIDIA HPC SDK on `PATH`, or use its full path in the CMake command. GPU execution also requires a compatible NVIDIA driver. GNU Fortran 11.4 with NVHPC 25.3 and an H100 NVL was verified with the official 1.1.1 sources. First clone the 1.1.1 tag as above. From the PIMD source directory, build and install only the AccelNet C library, then build PIMD against it:

```
> cmake -S /path/to/AccelNet-1.1.1 -B /path/to/accelnet-gpu-build \
    -DCMAKE_Fortran_COMPILER=nvfortran -DCMAKE_C_COMPILER=gcc \
    -DCMAKE_BUILD_TYPE=Release -DBUILD_SHARED_LIBS=ON \
    -DACCELNET_BUILD_OPENMP_TARGET=ON -DACCELNET_TARGET_SERIAL=OFF \
    '-DACCELNET_OPENMP_TARGET_FLAGS=-mp=gpu -gpu=cc90' \
    -DBUILD_TESTING=OFF
> cmake --build /path/to/accelnet-gpu-build \
    --target AccelNetTarget --parallel 8
> cmake --install /path/to/accelnet-gpu-build \
    --prefix /path/to/accelnet-gpu-install --component TargetC
> cmake -S . -B build-accelnet-gpu \
    -DCMAKE_Fortran_COMPILER=gfortran -DCMAKE_BUILD_TYPE=Release \
```

```
    -DACCELNET=ON -DACCELNET_PROVIDER=EXTERNAL -DBUILD_TESTING=OFF \
    -DAccelNetC_DIR=/path/to/accelnet-gpu-install/lib/cmake/AccelNetC
> cmake --build build-accelnet-gpu --target pimd.mpi.x --parallel 8
```

Adjust the GPU architecture for the device and compiler version. Use separate build directories for different compilers. The external provider links the installed C library without fetching or recompiling AccelNet. The matching NVHPC runtime libraries must remain available when running PIMD, including on compute nodes. The installed `AccelNetC` package records the shared-library path; retain the installation after linking. If compiler runtime paths differ on a compute node, load its NVHPC environment or set `LD_LIBRARY_PATH` there. Setting `ACCELNET=ON` alone builds the CPU provider; a GPU input tag requires the separately built offload library and the external provider.

In the run directory, keep `structure.dat`, the model files and the usual model format, units and evaluation settings described below. In `input.dat`, select **<ipotential>** = `ACCELNET` and add:

```
<accelnet_backend>
GPU
<accelnet_device>
0
```

The default backend is `CPU`; the default device index is 0. The device is a nonnegative index among GPUs visible to each MPI rank. A GPU request fails if the library or device cannot perform GPU execution; it never silently falls back to the CPU. The same GPU-enabled library can also run with **<accelnet_backend>** = `CPU`. For example:

```
> CUDA_VISIBLE_DEVICES=0 OMP_TARGET_OFFLOAD=MANDATORY OMP_NUM_THREADS=1 \
    mpiexec -n 1 /path/to/build-accelnet-gpu/pimd.mpi.x
```

Each MPI/XMPI rank owns a model and workspace. PIMD does not automatically assign ranks to GPUs: use the scheduler's per-rank binding or a launcher wrapper exposing one GPU per rank, with device index 0. Multiple ranks may share a device. Neighbor construction, unit conversion, and MPI reductions stay on the CPU; each owned bead's centers are passed together to AccelNet for inference. Start with one MPI rank per visible GPU. When using a scheduler, preserve its GPU visibility settings instead of overriding them with a global device list. In XMPI the process count must still equal **<np_beads>**×**<np_force>**. Rank zero prints `AccelNet C API backend: GPU; device: 0` for the example above; missing GPU support or an invalid device produces an error.

MPI bead parallelism and GPU assignment. For **<method>** = `PIMD` with ordinary MPI, a starting configuration is one MPI rank per GPU, **<np_beads>** equal to that rank count and **<np_force>** equal to 1. There is no requirement to have one GPU per bead. For example, 32 beads on four GPUs can use:

```
<method>
PIMD
<nbead>
32
<xmpi>
OFF
<np_beads>
4
<np_force>
1
<accelnet_backend>
GPU
<accelnet_device>
0
```

Keep the usual ensemble, integrator, model and unit settings as well. Rank 0 evaluates beads 1, 5, 9, ..., 29; rank 1 evaluates 2, 6, 10, ..., 30, and similarly for ranks 2 and 3. Each rank evaluates its eight beads sequentially on its assigned GPU. <np_beads> partitions beads; it does not assign GPU devices. With <np_force> greater than 1, ranks also split the atomic centers of each bead, and each rank still uses its own selected GPU.

If the scheduler already exposes one GPU per rank, launch normally and keep <accelnet_device> equal to 0. Otherwise, for Open MPI, the supplied lib/accelnet/mpi_gpu_rank.sh wrapper selects the GPU at the node-local rank's position in CUDA_VISIBLE_DEVICES. On a single node where GPUs 0–3 are allocated to this job, for example:

```
> CUDA_VISIBLE_DEVICES=0,1,2,3 OMP_TARGET_OFFLOAD=MANDATORY \
    OMP_NUM_THREADS=1 mpiexec -n 4 \
    /path/to/pimd-master/lib/accelnet/mpi_gpu_rank.sh \
    /path/to/build-accelnet-gpu/pimd.mpi.x
```

The wrapper accepts GPU indices or UUIDs and prints each rank's binding. It rejects more local ranks than entries in the GPU list. For a two-GPU node, use two ranks, <np_beads>=2, <np_force>=1 and a two-entry device list; each rank then evaluates 16 of the 32 beads. Use only devices allocated to the job, and do not use this wrapper on top of a scheduler's existing one-GPU-per-rank visibility. A shared four-GPU visibility list without rank-specific selection would make all ranks with device index 0 use the same first GPU.

AENET model input. AccelNet reads both native aenet binary networks and ASCII networks. Existing binary .ann and .nn files can be used directly; conversion to ASCII is not required. The following blocks are added to input.dat, together with the usual method settings and structure.dat. This interface currently uses the tagged input directly; a dedicated structured AccelNet TOML interface has not been added to pimd-prep. This example loads the binary SiO_2 networks:

```
<ipotential>
ACCELNET
<accelnet_model_format>
AENET
<ntype_accelnet>
2
Si models/Si.15-15.ann
O  models/O.15-15.ann
<accelnet_energy_unit>
EV
<accelnet_length_unit>
ANGSTROM
<accelnet_chebyshev_version>
1
<accelnet_chebyshev_evaluation>
AUTO
```

The first line of <ntype_accelnet> is the number of model elements, followed by one element symbol and network filename per line. Include all elements declared by the model, even if some are absent from the structure. File-list order and atom order may differ: the adapter maps elements by name and orders the loaded networks using their embedded metadata. Paths may be absolute or relative to the run directory; quoted network paths are accepted. All MPI ranks must be able to read the files.

The reader is selected by the filename: .ascii (case sensitive) selects ASCII, and all other suffixes select Fortran native binary. To use ASCII exports, replace the network filenames above with the corresponding .ascii filenames. The SiO_2 .ann and PdH .nn example models can be loaded directly without conversion. To reproduce PIMD's bundled aenet 2.0.3, set <accelnet_chebyshev_version> to 1; the angular convention is not inferred from binary model metadata. ASCII files are preferable for portability across compiler environments.

N2P2 model input. For an N2P2 model directory, replace the format and file-list settings with

```
<ipotential>
ACCELNET
<accelnet_model_format>
N2P2
<accelnet_model_dir>
models/n2p2
<accelnet_energy_unit>
EV
<accelnet_length_unit>
ANGSTROM
<accelnet_g5_evaluation>
MOMENT
```

The directory contains `input.nn`, the element-specific weight files (for example, `weights.001.data` for H), and `scaling.data` when required by the model. Coordinates and the simulation cell come from PIMD; no N2P2 `input.data` is needed. The reader supports the short-range 2G model subset implemented by AccelNet; 4G/charge models and weighted/compact descriptors are outside this supported subset.

Units and evaluation controls. `<accelnet_model_format>` is required and accepts `AENET` or `N2P2`. `<accelnet_energy_unit>` is required and accepts `EV` or `HARTREE`. `<accelnet_length_unit>` is required and accepts `ANGSTROM` or `BOHR`. These units describe physical model inputs and outputs, not the internal N2P2 `conv_energy`/`conv_length` normalization. AccelNet handles that normalization and atomic energy offsets.

The following settings are optional:

`<accelnet_chebyshev_version>` Accepts 0, 1, or 10; default 0. Select the descriptor convention used by the model. This is distinct from the algorithm selection below.

`<accelnet_chebyshev_evaluation>` Accepts `AUTO`, `DIRECT`, or `MOMENT`; default `AUTO`.

`<accelnet_g5_evaluation>` Accepts `AUTO`, `DIRECT`, `MOMENT`, or `MOMENT_FORCE`; default `AUTO`. This setting also applies to N2P2 symmetry-function type 9.

`<neighbor_list_skin>` Nonnegative distance in Bohr, independently of model length units; default 1.0. Zero rebuilds the neighbor list on every evaluation.

Enumeration values are case insensitive. Evaluation modes affect applicable descriptors independently of whether the model was loaded in AENET or N2P2 format. Input settings are read from `input.dat`, falling back to `input_default.dat` when a tag is absent.

PIMD converts coordinates to model units and converts energy, forces, and virial back to atomic units. The virial is an energy tensor, not stress: the adapter does not divide it by volume. PIMD performs the bead average once. Periodic image handling includes triclinic cells and multiple images within the cutoff. Each owned bead has a neighbor cache, rebuilt on cell changes or displacements of at least half the skin. The rebuild uses a quadratic atom-pair search, which can limit performance for large systems.

Execution and scope. From a run directory containing the input files and model, for example:

```
> mpiexec -n 4 /path/to/PIMD/build-accelnet/pimd.mpi.x
```

Ordinary MPI and the supported XMPI methods are available; see `<xmpi>` in Section 11.4.228 for the method/ensemble restrictions. For example, add the following settings to the model input to validate forces with four beads and four MPI ranks:

```
<method>
TESTFORCE
<nbead>
4
```

```
<xmpi>
ON
<np_beads>
2
<np_force>
2
```

Launch this example with `mpiexec -n 4`. The two bead groups each contain two force ranks; the total MPI size must equal **<np_beads>**×**<np_force>**. Use **<method>** = `TESTVIRIAL` with a periodic structure to check all nine cell derivatives. For dynamics, the current XMPI support includes PIMD/NVT with MNHC, CMD/NVE, RPMD/NVE, and BCMD/NVE or NVT, as well as GEOOPT. Variable-cell PIMD (NPT/NPH/NTT) uses ordinary MPI with **<xmpi>** = `OFF`; the existing XMPI integrator restrictions apply to AccelNet as they do to other force backends. The interface uses one fixed model per MPI rank. On-the-fly training, SLHMC model updates, and heat-flux output are not implemented.

CTest validation. Tests additionally require Python 3 and an MPI launcher capable of starting up to eight local ranks. Enable and build the fixture generator before running CTest:

```
> cmake -S . -B build-accelnet -DACCELNET=ON -DBUILD_TESTING=ON
> cmake --build build-accelnet --parallel 8 \
    --target pimd.mpi.x pimd_accelnet_fixtures
> ctest --test-dir build-accelnet -L accelnet --output-on-failure
```

The `accelnet` label groups model loading and units (`loading`), evaluation algorithms (`modes`), force and nine-component virial finite differences (`derivatives`), and rank/bead decompositions including XMPI (`mpi`).

Additional tests compare the actual original AENET and N2P2 backends with AccelNet using the same model files. To register both comparison groups, prepare the original backend dependencies as in Sections 6.4.1 and 6.4.4, including PIMD-patched N2P2 libraries, Eigen, and GSL:

```
> cmake -S . -B build-reference -DACCELNET=ON \
    -DAENET=ON -DN2P2=ON -DBUILD_TESTING=ON \
    -DN2P2_LIBRARY_DIR=/path/to/patched-n2p2/lib
> cmake --build build-reference --parallel 8 \
    --target pimd.mpi.x pimd_accelnet_fixtures
> ctest --test-dir build-reference -L reference --output-on-failure
```

The groups are named `accelnet.reference_aenet` and `accelnet.reference_n2p2`. Each is registered only when its original backend is enabled. They compare energies, forces, and virials in free, orthorhombic, and triclinic geometries, and check parallel derivatives. `N2P2_LIBRARY_DIR` defaults to the PIMD **`lib`** directory. If GSL is installed outside standard system paths, make its shared libraries visible to the runtime loader as well.

The groups above use synthetic regression fixtures, not physical models. Inputs and logs are retained under the build directory's **`tests/external/AccelNet/runs`**. Use `ctest -test-dir build-accelnet -N` to list registered tests. `BUILD_TESTING=OFF` omits the test targets.

The additional groups `accelnet.examples_aenet` and `accelnet.examples_n2p2`, selected by the `examples` label, compare the trained SiO_2/PdH binary aenet models and the SiO_2/ice N2P2 models stored in **`tests/external/AccelNet/fixtures`**. Runnable AccelNet inputs are in **`examples/02_path_integral_dynamics`** under those systems. See `tests/external/AccelNet/README.md` for the comparison script; it can also measure PIMD integration time.

For an external library, synthetic tests use the fixture generator from a separate CPU PIMD build. Set `ACCELNET_TEST_FIXTURES` to that build's `tests/external/AccelNet/pimd_accelnet_fixtures` executable and `ACCELNET_TEST_MODEL_DIR` to the AccelNet source directory **`AccelNetPredictor/test/data/n2p2-virial-angular`**. Enable `ACCELNET_TEST_GPU=ON` to register additional `accelnet.gpu.*` tests; select them with `ctest -L gpu`. The `cpu` label selects host execution of the same library. Without the fixture

paths, external builds register only the trained-model comparisons whose original backends are enabled. Full configure and test commands are in `lib/accelnet/README.md`.

6.4.3 MTP

IMPORTANT: The user is solely responsible for all issues related to the MLIP code [90, 91], which is distributed via the website

```
https://mlip.skoltech.ru
```

PIMD does not provide any license for the MLIP code.

In the PIMD code, the subroutines `force_mtp.F90` and `force_mtp_MPI.F90` have been reimplemented in `Fortran 90` based on the original MLIP code written in `C++`.

The potential data file `pot.mtp`, in the native MLIP format, must be present in the execution directory. The file `pot.mtp` should be generated by the MTP training procedure using the MLIP code. Manually modifying the `pot.mtp` file is strongly discouraged, as it may lead to unexpected errors when the PIMD code reads the potential parameters.

To use the MTP potential, the following keyword must be specified in the file `input.dat`:

```
<ipotential>
MTP
```

This keyword activates the MTP potential in the PIMD code.

The execution is performed in the usual manner using either **`pimd.x`** or **`pimd.mpi.x`**.

As an example, a set of input files for a path integral simulation of hydrogen in bcc iron using the MTP potential is provided in the directory

```
examples/02_path_integral_dynamics/FeH/pimd_nvt_mtp
```

6.4.4 N2P2

IMPORTANT: The user is solely responsible for all issues related to the N2P2 license. PIMD does not provide any N2P2 license.

The high-dimensional neural network N2P2 code [95] is linked through the subroutines `force_n2p2_MPI.F90` and `predict_n2p2_MPI.F90`. The N2P2 code must be properly installed in order to use this interface. Currently, only version `2.2.0` is supported; support for version `2.3.0` is planned for a future release. This means that, at present, only second-generation NNPs generated by N2P2 are supported in PIMD.

Internal link of N2P2.. Follow the directions below.

- **Step 1: Obtain the N2P2 source code and apply the patch.** Go to the N2P2 directory.

  ```
  > cd lib/n2p2/
  ```

  Run the bash script to download the source code from GitHub and apply the patch.

  ```
  > ./getandapply_patch.sh
  ```

  This script downloads the source files for N2P2 version `2.2.0` and applies the patch required to interface with PIMD.
- **Step 2: Compile the modified N2P2 code.** Go to the directory containing the modified N2P2 source code.

```
> cd n2p2-2.2.0.modified
```

Compile the modified N2P2 code. Note that when using the Intel compiler, the `-ipo` option must be removed from the `PROJECT_CFLAGS` variable in **`./src/makefile.intel`** to avoid runtime errors. The following commands overwrite the `PROJECT_CFLAGS` variable by passing arguments directly to the make command.

```
> make -C ./src/libnnp COMP=intel PROJECT_CFLAGS="-O3 -xHost -std=c++11"
> make -C ./src/libnnptrain COMP=intel PROJECT_CFLAGS="-O3 -xHost -std=c++11"
> make -C ./src/libnnpif COMP=intel PROJECT_CFLAGS="-O3 -xHost -std=c++11"
```

If successful, new files named `libnnp.a`, `libnnptrain.a`, and `libnnpif.a` will be created. Copy them to the lib directory.

```
> cp ./lib/libnnp.a ../../
> cp ./lib/libnnptrain.a ../../
> cp ./lib/libnnpif.a ../../
```

Return to the main directory.

```
> cd ../../
```

- **Step 3: Configure PIMD with the N2P2 interface.** The recommended CMake configuration is

```
> cmake -S . -B build-n2p2 -DN2P2=ON
```

Build the configured tree as described in Section 1.2. If N2P2 was compiled with OpenMP, add `-DOPENMP=ON` to the configuration command. Otherwise, leave this option at its default, OFF. Use `-DN2P2_LIBRARY_DIR=/path/to/n2p2/lib` if the patched archives are outside the default **`lib`** directory. N2P2 and PIMD must use compatible compiler toolchains; see Section 1.3.

Legacy makefile method: To compile the PIMD code using the makefile, edit the file `makefile` to activate the `-Dn2p2` option.

```
> cp -r source compile
> cd compile
> vi makefile
> make
```

If successful, this process generates **`pimd.mpi.x`**, the PIMD executable internally linked to N2P2.

N2P2 input files.. In addition to the PIMD input files (`input.dat`, `input_default.dat`, `structure.dat`), the N2P2 input files are also required (*input.nn*, *scaling.data*, *weights.***.data*). See the N2P2 manual for instructions on preparing these input files.

For the TOML front end, these conventional N2P2 filenames are not duplicated in the generic `[files]` table. In particular, numbered manifest keys such as `n2p2_model_2` and `n2p2_weights_2` are unnecessary. **`pimd-prep check`** reads the `elements` line from `input.nn`, requires `scaling.data`, and checks the corresponding `weights.ZZZ.data` files automatically.

PIMD input file to create xsf files.. Use the PIMD, PIHMC, or REHMC method to create a set of xsf files. The following keyword should be set in the file `input.dat`.

```
<iprint_xsf_n2p2>
10
```

This keyword specifies the step interval for printing N2P2 xsf files; in this case, every “10” steps.

If the PIHMC or REHMC method is selected, one can also set

```
<ioption_xsf_n2p2>
1
```

This keyword controls the printing option for xsf files. In this case, “1” indicates that xsf files are printed for both accepted and rejected structures. The xsf files are created in the directory named “structures”.

PIMD input file to generate/train using xsf files.. Use the PIMD, PIHMC, or REHMC method to generate input data and train networks using a set of xsf files. The following keywords should be set in the file `input.dat`.

```
<istep_train_n2p2>
100
```

This keyword specifies the step interval for generating and training N2P2. In this case, “100” means that generation and training are performed every 100 steps.

```
<lstep_train_n2p2>
3000
```

This keyword sets the final step for N2P2 generation and training. In this case, “3000” means that generation and training stop at step 3000.

```
<minxsf_train_n2p2>
300
```

This keyword specifies the minimum number of xsf files required to train N2P2. In this case, “300” means that the generation and training process is skipped until 300 xsf files are found in the “structures” directory.

PIMD input file to run simulations (predict).. The following keywords should be set in the file `input.dat`.

```
<ipotential>
N2P2
```

This keyword declares the use of the N2P2 code.

PIMD input file to run simulations (predict/generate/train).. In the PIHMC or REHMC method with a dual potential, where the low-level potential is N2P2, generation and training can be performed on the fly. For example, use the following combination:

```
<method>
PIHMC

<ipotential>
DUAL

<dual_potential>
VASP N2P2
```

The following keywords are useful.

```
<istep_hmc>
8
```

```
<istep_max_hmc>
128

<istep_adjust_hmc>
100

<istep_mul_hmc>
2

<ratio_min_hmc>
0.05

<ratio_max_hmc>
0.20
```

These keywords define a variable number of MD steps per Metropolis step. In this example, the number of MD steps per Metropolis step can vary from “8” (initial value) to “128” following a geometric progression with a common ratio of “2”. The following adjustment is performed every “100” Metropolis steps: The number of MD steps per Metropolis step is increased if the acceptance ratio is lower than “0.05”, and decreased if the acceptance ratio is higher than “0.20”.

Execution of PIMD.. Execution is performed in the usual way using the `pimd.mpi.x` command.

6.4.5 MACE

IMPORTANT: The user is solely responsible for all issues related to the MACE license. PIMD does not provide any MACE license.

The MACE machine-learning potential [92] is linked through the subroutine `force_mace_MPI.F90`. The current interface is designed for the MPI version of PIMD. The recommended executable for production calculations is `pimd.mpi.x`. Two inference backends are available:

- **Python backend**: force evaluation is carried out through the Python helper runtime in `lib/mace`.
- **libtorch backend**: force evaluation is carried out directly in the linked C++/libtorch interface in `lib/mace/libtorch`. Python is still used for helper tasks such as model export and on-the-fly training.

Both backends require the helper scripts in `lib/mace`.

XMPI. MACE inference is supported with `<xmpi>`=“ON”. The Python transport and the libtorch backend use the XMPI bead/force decomposition and reconstruct per-bead force/virial results within each subgroup. On-the-fly MACE training is not yet supported under XMPI; use inference only (`<istep_train_mace>` ≤ 0), or use ordinary MPI for training. For helper scripts and on-the-fly training, a Python environment with `torch`, `mace`, `ase`, and `PyYAML` is required.

Compilation and Python runtime.. Follow the directions below.

- **Step 1: Configure the PIMD code with the MACE interface. CMake method:** Create a build directory and invoke CMake with the MACE option enabled. The backend is selected with `MACE_INFER_BACKEND`. The commands below show only the MACE-specific configuration. Build the configured tree using the common procedure in Section 1.2. For the Python backend:

```
> cmake -S . -B build-mace-python \
>   -DMACE=ON \
>   -DMACE_INFER_BACKEND=python
```

  For the libtorch backend:

```
> cmake -S . -B build-mace-libtorch \
>   -DMACE=ON \
>   -DMACE_INFER_BACKEND=libtorch \
>   -DLIBTORCH_ROOT=/path/to/libtorch
```

Here, `LIBTORCH_ROOT` must point to a prebuilt libtorch distribution containing **`share/cmake/Torch/TorchConfig.cmake`**. The libtorch package should be compatible with the PyTorch version used in the helper Python environment. If a dedicated Python virtual environment should also be prepared automatically, the following option can be added to either backend:

```
> cmake -S . -B build-mace-python \
>   -DMACE=ON \
>   -DMACE_INFER_BACKEND=python \
>   -DMACE_INSTALL_VENV=ON
```

In this case, the virtual environment is created below **`build/lib/mace/venv`**. The MACE backend can be combined with other CMake options. For example, to build the QE + MACE/libtorch executable:

```
> cmake -S . -B build-mace-libtorch-qe \
>   -DMACE=ON \
>   -DMACE_INFER_BACKEND=libtorch \
>   -DLIBTORCH_ROOT=/path/to/libtorch \
>   -DQE=ON \
>   -DQEVERSION=6.3
```

Makefile method: The current MACE interface is maintained primarily through CMake. The makefile-based build is not the recommended method for MACE.

- **Step 2: Select the Python executable used by MACE.** The recommended method is to export the environment variable `PIMD_MACE_PYTHON` before running PIMD:

```
> export PIMD_MACE_PYTHON=/path/to/python
```

If `MACE_INSTALL_VENV=ON` was used at configuration time, the generated helper file may be sourced:

```
> source build/lib/mace/use_mace_python.sh
```

If `PIMD_MACE_PYTHON` is not set, the helper scripts fall back to `sys.executable` or the script shebang, depending on the helper. This Python environment is required not only for the Python inference backend, but also for libtorch export and on-the-fly MACE training.

- **Step 3: Prepare the MACE input files.** For inference, the file `mace_infer_options.yaml` is required. This file must define the model used in production calculations. For example:

```
mace:
  model: SiO2_MACE.model
  device: cuda
  device_assignment: bead_local_rank
  gpus_per_node: 4
  device_id: 0
  default_dtype: float64
  r_max: 6.0
```

Here, `device_assignment: fixed` assigns all local bead ranks to the same `device_id`. `device_assignment: bead_local_rank` assigns the local bead-owning MPI ranks on a node to GPU indices in local order. When `gpus_per_node` is smaller than the number of local bead-owning ranks, the GPU indices are reused cyclically. For on-the-fly learning, the file `mace_train_options.yaml` is also required. This file specifies the finetuning options and the foundation model. For example:

```
mace:
  foundation_model: mace_mp_0_small.model
  device: cuda
  train_gpus_per_node: 1
  max_num_epochs: 50
  patience: 10
```

Here, `train_gpus_per_node` specifies how many GPUs are used for on-the-fly training on the node that contains PIMD world rank 0. If `device: cuda` and `train\_gpus\_per\_node > 1`, PIMD launches single-node distributed MACE training with `python -m torch.distributed.run --standalone`. If `train\_gpus\_per\_node = 1`, training is carried out on a single GPU. Both YAML files may either use a top-level `mace:` mapping, or place the options directly at the top level. The files `model` and `foundation_model` are resolved relative to the directory containing the corresponding YAML file, unless absolute paths are given.

If successful, this procedure generates **`pimd.mpi.x`**, the PIMD executable internally linked to MACE.

MACE input files.. In addition to the standard PIMD input files (`input.dat`, `input_default.dat`, `structure.dat`), the MACE interface uses the following files:

- `mace_infer_options.yaml`: inference settings.
- `mace_train_options.yaml`: training settings (only for on-the-fly learning).
- `*.model`: the active MACE model and, optionally, the foundation model.
- *mace_libtorch.pt*: libtorch-exported model generated automatically for the libtorch backend.
- *mace_libtorch.runtime*: libtorch runtime settings generated automatically from `mace_infer_options.yaml`.

The keywords **<mace_infer_options_file>** and **<mace_train_options_file>** are resolved relative to the current run directory unless absolute paths are given. In contrast, **<dir_train_mace>** is used as a working directory for generated runtime and training files. PIMD creates this directory automatically when needed. During on-the-fly learning, PIMD automatically creates *structures*, `mace_training.xyz`, `mace.ini`, `mace_count_1.out`, `mace_count_2.out`, and the helper output directories below **<dir_train_mace>**.

PIMD input file to run simulations (predict).. The following keyword should be set in `input.dat`:

```
<ipotential>
MACE
```

This keyword declares the use of the MACE potential. The following optional keywords are also available:

```
<mace_infer_options_file>
mace_infer_options.yaml

<dir_train_mace>
mace_work
```

The first keyword selects the YAML file used for inference. The second keyword sets the working directory where generated MACE runtime files are created. For the libtorch backend, the runtime control parameters should be set in `mace_infer_options.yaml` through `device`, `device_assignment`, `gpus_per_node`, `device_id`, and `r_max`. The legacy input keywords **<mace_use_cuda>**, **<mace_device_id>**, and **<mace_rmax>** remain as deprecated fallbacks and should not be used in new input files.

PIMD input file to create xsf files.. Use the PIMD, PIHMC, or REHMC method to create a set of xsf files for MACE learning. The following keyword may be set in `input.dat`:

```
<iprint_xsf_mace>
10
```

This keyword specifies the interval for writing MACE xsf files; in this case, every 10 steps.

If the PIHMC or REHMC method is selected, the following keyword may also be set:

```
<ioption_xsf_mace>
1
```

This keyword controls whether xsf files are written only for accepted structures or for all trial structures. The xsf files are created in the directory named *structures* below <dir_train_mace>.

PIMD input file to generate/train using xsf files.. Use the PIMD, PIHMC, or REHMC method to generate training data and train MACE using the xsf files. The following keywords are available in input.dat:

```
<istep_train_mace>
100
```

This keyword specifies the interval for MACE training.

```
<lstep_train_mace>
3000
```

This keyword sets the final step for MACE training.

```
<minxsf_train_mace>
300
```

This keyword specifies the minimum number of xsf files required to start training.

```
<mace_train_options_file>
mace_train_options.yaml
```

This keyword selects the YAML file used for training. **NOTE:** The legacy tag <mace_generate_command> is not read by the current PIMD 2.8.0 MACE source and must not be used in new inputs.

```
<istep_save_mace>
30

<dir_save_mace>
saved_models
```

These keywords save snapshot models periodically.

The active production model is taken from the model entry in mace_infer_options.yaml. When training succeeds, this active model is replaced by the updated model, optional snapshots are written to <dir_save_mace>, and the libtorch backend automatically regenerates *mace_libtorch.pt* from the updated active model.

PIMD input file to run simulations (predict/generate/train).. In the PIHMC or REHMC method with a dual potential, MACE can be used as the low-level potential while a higher-level potential such as QE or VASP is used to generate reference data. For example:

```
<method>
PIHMC

<ipotential>
DUAL

<dual_potential>
```

```
QE MACE
```

or

```
<dual_potential>
VASP MACE
```

In this setup, MACE is used for inexpensive trial moves, while the high-level potential is used for acceptance tests and for generating updated training data.

Remarks.. For the Python backend, the MACE interface launches one Python daemon per MPI rank that is responsible for the local bead calculations. Therefore, the Python environment specified by `PIMD_MACE_PYTHON` must be available on all compute nodes. This is particularly important when running batch jobs on supercomputers. For the libtorch backend, force evaluation itself is performed in the linked C++/libtorch code. Python is still used for model export and on-the-fly training. The libtorch backend supports two GPU assignment modes configured in `mace_infer_options.yaml`. With `device_assignment: fixed`, all local bead ranks use the same `device_id`. With `device_assignment: bead_local_rank`, the bead-owning ranks on the same node are assigned to GPU indices in local bead order. In this mode, `gpus_per_node` specifies the number of visible GPUs on each node. If `gpus_per_node` is smaller than the number of local bead-owning ranks, the GPU indices are reused cyclically (for example, 4 beads on 2 GPUs are assigned as 0, 1, 0, 1).

6.4.6 ASE calculator interface

The ASE calculator interface is linked through the subroutine `force_ase_MPI.F90`. This interface allows PIMD to evaluate energies, forces, and virials through Python calculators provided by ASE or ASE-compatible packages. The current implementation is intended for the MPI version of PIMD. The recommended executable is therefore **`pimd.mpi.x`**. The serial executable does not support the ASE calculator interface.

Configure an ASE-enabled build with

```
> cmake -S . -B build-ase -DASE=ON
```

Use the common build procedure in Section 1.2 after configuration. Compiler, MPI, and general build-directory choices are covered in Sections 1.3–1.8 and are not repeated here. If ASE and the target calculator are installed in a specific Python environment, set the environment variable `PIMD_ASE_PYTHON` before running PIMD:

```
> export PIMD_ASE_PYTHON=/path/to/python
```

If `PIMD_ASE_PYTHON` is not set, the helper scripts fall back to the local file *.pimd_ase_python* when it is present, and otherwise to the Python specified by the script shebang.

PIMD input file.. To use the ASE calculator interface, set in `input.dat`:

```
<ipotential>
ASE

<ase_infer_options_file>
ase_infer_options.yaml
```

The keyword **`<ase_infer_options_file>`** selects the YAML file that describes which calculator is constructed in Python.

ASE inference YAML file.. The file `ase_infer_options.yaml` must contain a `calculator:` section. Relative paths given in the YAML file are resolved relative to the directory containing that YAML file. The optional section `properties:` controls how the virial is constructed. The following values are available:

- `virial_mode: stress`: use the stress tensor returned by the calculator.
- `virial_mode: forces`: construct the virial from $F^T R$.
- `virial_mode: auto`: use stress when available and otherwise fall back to $F^T R$.

For calculators that provide stress, `virial\_mode: stress` is recommended. In particular, the current MACE implementation through ASE has been checked against `TESTVIRIAL` in this mode.

Examples of ASE calculators.. The ASE interface supports several styles of calculator specification.

- **MACE through ASE.** The dedicated MACE interface remains available through `<ipotential> = MACE`. However, the same model can also be used through the ASE interface:

```
calculator:
  type: mace
  kwargs:
    model: mace_mp_0_small.model
    device: cpu
    default_dtype: float64

properties:
  virial_mode: stress
```

- **CHGNet through ASE.** CHGNet is used as an ASE-compatible calculator through the generic `custom` type:

```
calculator:
  type: custom
  module: chgnet.model
  class: CHGNetCalculator
  kwargs:
    use_device: cpu

properties:
  virial_mode: stress
```

- **NequIP through ASE.** NequIP is used through a compiled model prepared in advance:

```
calculator:
  type: nequip
  kwargs:
    compile_path: mir-group__Allegro-MP-L__0.1.cpu.nequip.pth
    device: cpu

properties:
  virial_mode: stress
```

Files and examples.. The ASE interface uses the standard PIMD input files (`input.dat`, `input_default.dat`, `structure.dat`) plus the calculator YAML file specified by `<ase_infer_options_file>`. Current distributed examples that exercise the ASE interface include:

```
examples/02_path_integral_dynamics/BaZrO3_H_gb/pimd_mace
examples/02_path_integral_dynamics/SiO2/pimd_npt_chgnet
examples/02_path_integral_dynamics/TiO2/pimd_npt_chgnet
```

The interface can also be configured for NequIP through an ASE calculator YAML file, but PIMD 2.8.0 does not distribute a separate curated NequIP example directory. Model files and Python packages remain external to PIMD. For foundation or generic models distributed outside this repository, the example directories provide helper scripts such as *setup_env.sh* and *download_model.sh*. For external ASE models, a typical workflow is to prepare the Python environment and model explicitly and then run the MPI-enabled PIMD executable. GPU

settings can be stored in a separate calculator YAML file; the device selection belongs to the ASE calculator configuration rather than to the PIMD trajectory integrator itself.

6.4.7 PFP

IMPORTANT: The user is solely responsible for all matters related to the PFP license. PIMD does not provide any PFP license.

The machine learning potential of Matlantis PFP [96] is linked through the subroutine `force_pfp.F90`. Note that `force_pfp.F90` does not support bead-parallel calculations, but single GPU acceleration is available. The MPI version (`force_pfp_MPI.F90`), which will support multiple GPU acceleration, is currently under development. The PFP code must be properly installed in order to use this interface. This service is provided by the commercial cloud platform of The Matlantis Corporation: `https://matlantis.com/en/product/about-pfp/`. Please contact them for any inquiries.

Optional model-selection file.. The file `pfp_model.dat` is optional. If it is present, the first two lines select the PFP version and model passed to the interface. If it is absent (or contains no usable selection), the PIMD interface passes blank/default selectors to PFP. Therefore a missing `pfp_model.dat` must not be reported as a pre-run error by `pimd-prep`; installation, credentials, and license availability are separate runtime prerequisites that cannot be validated from the calculation directory alone.

Part IV

Hybrid Methods, Analysis, and Utilities

7 Hybrid potentials

The hybrid potentials based on QM/MM and ONIOM methods can be used in the PIMD code.

7.1 QM/MM

The following QM/MM methods are implemented in the PIMD code.

- The mechanical embedding QM/MM method.
- The electronic embedding QM/MM method with SMASH.
- The multiple time scale method with mechanical embedding QM/MM method.

These methods can be used in combination with the link atom. The following methods are NOT implemented, although they might be future targets.

- The electronic embedding QM/MM method with codes other than SMASH.
- The multiple time scale method with electronic embedding QM/MM method.

The theoretical background of the QM/MM methods is briefly explained in Section 13.18. With <xmpi>="ON", QM/MM is currently supported for mechanical embedding only; ONIOM is also XMPI-enabled. QM/MM electronic embedding (EE) continues to require ordinary MPI. The XMPI wrappers use bead-group ownership and subgroup communication so the QM/MM and ONIOM results are reconstructed within the corresponding force subgroup; ONIOM external jobs are launched by the bead-group leader.

IMPORTANT: The user is solely responsible for all the license issues of external software used to compute hybrid potentials, e.g., SMASH, ABINIT-MP, DFTB, GAMESS, GAUSSIAN, MOLPRO, MOPAC, ORCA, TURBOMOLE. PIMD does not provide any of these licenses.

7.1.1 QM/MM setup

In the two-layer QM/MM method, the whole system is composed of two layers, A and B. Layer A is the primary region (e.g., the solute), while layer B is the secondary region (e.g., solvent). If the boundary between layers A and B is chosen such that a bonded pair is separated between layers A and B, the link atom method should be used. The link atom L caps the bonds broken in layer A when layer A is separated from layer B. The following files are required in the QM/MM method.

- All files required for QM calculations
- `input.lammps`, `data.lammps`, and `parm.lammps` for the CFF part
- `structure.dat`
- `input.dat`

Prepare all files required for QM calculations.. The data files required for QM calculations are:

- `abinit-mp.dat` for ABINIT-MP
- `dftb.dat` for DFTB
- `gamess.dat` for GAMESS
- `g98.dat` for GAUSSIAN 98
- `g03.dat` for GAUSSIAN 03
- `g09.dat` for GAUSSIAN 09
- `g16.dat` for GAUSSIAN 16
- `molpro.dat` for MOLPRO
- `mopac.dat` for MOPAC
- `orca.dat` for ORCA
- `turbo.dat`, *control.1* and *control.2* for TURBOMOLE

They should be placed in a subdirectory `./dat`, corresponding to the QM calculations of the subsystem (A) or (A+L). The atoms should be exactly in the same order as defined in `structure.dat`, i.e., the list of atoms in the primary region followed by the list of link atoms. Note that the files are not required for SMASH calculations since the information is already given by the keyword <smash_input> in the file `input.dat`.

Prepare the CFF files.. The classical part is specified with `input.lammps`, `data.lammps`, and `parm.lammps`, using the same CFF format as for <ipotential> = `CFF`. Double counting of interactions belonging to the QM region is handled by the QM/MM CFF coupling code; the user should not create a separate legacy `mm.dat` file.

Prepare structure.dat.. The file `structure.dat` must be in the following format for a water dimer:

```
6
BOHR
O   -1.13151   1.10684  -0.67760   1 A
H   -2.30412  -0.29737  -0.45427   2 A
H    0.45453   0.27505  -0.94620   2 A
O   -5.29474  -1.87202   0.09742   1 B
H   -5.90817  -0.13639   0.18675   2 B
```

```
H   -6.79303  -2.84927   0.37973   2 B
```

where “A” and “B” are the primary and secondary regions, respectively. Otherwise, the region “B” can be abbreviated as:

```
6
BOHR
O   -1.13151   1.10684  -0.67760   1 A
H   -2.30412  -0.29737  -0.45427   2 A
H    0.45453   0.27505  -0.94620   2 A
O   -5.29474  -1.87202   0.09742   1
H   -5.90817  -0.13639   0.18675   2
H   -6.79303  -2.84927   0.37973   2
```

The file `structure.dat` must be in the following format for ethane:

```
8
ANGSTROM
C     2.14200   1.39500  -8.93200   1 AL 5 0.70 H
H     1.60400   0.76000  -8.26000   2 A
H     1.74500   2.38800  -8.88000   2 A
H     2.04300   1.02400  -9.93000   2 A
C     3.63100   1.41600  -8.53700   1 B
H     4.16900   2.05100  -9.21000   2 B
H     3.73100   1.78800  -7.53900   2 B
H     4.20300  -0.01200  -8.61200   2 B
```

“AL” means the atom in the primary region (C_1) connected to a link atom (L). “5” means the atom in the secondary region (C_5) which is linked to C_1. “0.70” is the equilibrium bond length ratio between C_1–L and C_1–C_5. “H” is the atomic element of the link atom, L=H. The region “B” can be abbreviated in this case as well. The file `structure.dat` must be in the following format for butane:

```
14
ANGSTROM
C     2.14200   1.39500  -8.93200   1 B
H     1.60400   0.76000  -8.26000   2 B
H     1.74500   2.38800  -8.88000   2 B
H     2.04300   1.02400  -9.93000   2 B
C     3.63100   1.41600  -8.53700   1 AL2 1 0.70 H 8 0.70 H
H     4.16900   2.05100  -9.21000   2 A
H     3.73100   1.78800  -7.53900   2 A
C     4.20300  -0.01200  -8.61200   1 B
H     3.66500  -0.64700  -7.94000   2 B
H     4.10400  -0.38400  -9.61000   2 B
C     5.69100   0.00900  -8.21800   1 B
H     6.08800  -0.98300  -8.27000   2 B
H     5.79100   0.38100  -7.22000   2 B
H     6.23000   0.64400  -8.89000   2 B
```

“AL2” means the atom in the primary region (C_4) connected to two link atoms (L2).

Prepare input.dat.. The details are explained in the next sections.

7.1.2 QM/MM with SMASH

The following keywords are important in the QM/MM method using an internal link with the SMASH code.

```
<ipotential>
QM/MM
```

This keyword specifies the use of the QM/MM method.

```
<qmmm_potential>
SMASH
```

This keyword specifies the use of the SMASH code.

```
<qmmm_embedding>
EE
```

This keyword specifies the embedding scheme, in the present case, electronic embedding. The use of mechanical embedding can be specified in the following way:

```
<qmmm_embedding>
ME
```

7.1.3 QM/MM with other codes

The following keywords are important in the QM/MM method using an external link with any code other than SMASH.

```
<ipotential>
QM/MM
```

This keyword specifies the use of the QM/MM method.

```
<qmmm_potential>
ORCA
```

This keyword specifies the use of an external code for QM calculations, in the present case, ORCA.

```
<qmmm_embedding>
ME
```

This keyword specifies the use of mechanical embedding.

```
<qmmm_dat_dir>
dat
```

This keyword specifies the input directory of QM calculations.

```
<qmmm_scr_dir>
scr
```

This keyword specifies the temporary scratch directory for QM calculations.

7.1.4 QM/MM with BEST method

The following keywords are important in the QM/MM method with the Boundary based on Exchange Symmetry Theory (BEST).

```
<ioption_best>
1
```

This keyword specifies the use of BEST.

```
<iobest>
1
```

This keyword specifies the central atom of BEST, in the present case, the first atom.

```
<ikind_best>
1
```

This keyword specifies the atomic kind of BEST, in the present case, the first kind.

```
<fc_best>
1.0
```

This keyword specifies the force constant of BEST, in the present case, 1.0 hartree/bohr^2.

```
<eps_best>
1.e-16
```

This keyword specifies the energy cutoff of BEST, in the present case, 10^{-16} hartree.

```
<iprint_best>
1
```

This keyword specifies the print interval of BEST in `best.out`.

7.1.5 Multiple time scale QM/MM method

The multiple time scale (MTS) molecular dynamics and path integral molecular dynamics methods are implemented in the PIMD code. The massive Nosé-Hoover chain thermostat is used to generate the NVT ensemble. The method can be used either with the ONIOM potential or the QM/MM potential. The method can be used in combination with the BEST method to deal with systems under open boundary conditions. The theoretical background of the MTS method is briefly described in Section 13.19.

To run the MTS simulations, the following keywords need to be specified:

```
<method>
MTS
```

This keyword sets the simulation method to the MTS method.

```
<ensemble>
NVT
```

This keyword sets the statistical ensemble to NVT.

```
<bath_type>
MNHC
```

This keyword sets the type of thermostat to massive Nosé-Hoover chain.

```
<ipotential>
QMMM
```

This keyword sets the potential, in the present case, QM/MM.

```
<dt>
0.25d0
```

This keyword sets the timestep, in the present case, 0.25 femtoseconds.

```
<nstep>
10
```

This keyword sets the number of steps, in the present case, ten steps.

```
<nmulti>
100
```

This keyword controls the time increment and the mass scaling factor for layer B. In the present case, the time increment for updating the forces with respect to layer B and the interaction between layers A and B is set to $1/100$ of <dt>, which means 0.0025 fs. In addition, the masses of all atoms in layer B are set to $1/10000$ of the physical masses.

7.2 ONIOM

The two-layer ONIOM method is implemented in the PIMD code. It can be used in combination with the link atom method. The three-layer ONIOM method is NOT implemented, although it may be a future target. The theoretical background of the ONIOM method is briefly explained in Section 13.18.

IMPORTANT: The user is solely responsible for all license issues related to external software used to compute hybrid potentials, e.g., SMASH, ABINIT-MP, DFTB, GAMESS, GAUSSIAN, MOLPRO, MOPAC, ORCA, TURBOMOLE. PIMD does not provide any of these licenses.

7.2.1 ONIOM setup

In the two-layer ONIOM method, the whole system is composed of two layers, A and B. Layer A is the primary region (e.g. the solute), while layer B is the secondary region (e.g. the solvent). If the boundary between layers A and B is chosen such that a bonded pair is separated into layers A and B, the link atom method is used. The link atom L caps the bonds broken in layer A when layer A is separated from layer B.

The following files are required for the ONIOM method:

- All files required for high- and low-level calculations
- `structure.dat`
- `input.dat`

Prepare all files required for high- and low-level calculations.. The data files required for high- and low-level calculations are

- `abinit-mp.dat` for ABINIT-MP
- `dftb.dat` for DFTB
- `input.lammps` plus its referenced setfl file for EAM
- `gamess.dat` for GAMESS
- `g98.dat` for GAUSSIAN 98
- `g03.dat` for GAUSSIAN 03
- `g09.dat` for GAUSSIAN 09
- `g16.dat` for GAUSSIAN 16

- `input.lammps`, `data.lammps`, and `parm.lammps` for CFF
- `molpro.dat` for MOLPRO
- `mopac.dat` for MOPAC
- `orca.dat` for ORCA
- `turbo.dat`, *control.1*, and *control.2* for TURBOMOLE

They should be placed in the three subdirectories named by **<oniom_dat_dir>**; the distributed CFF example uses **./dat_hi**, **./dat_lo_model**, and **./dat_lo_real**, corresponding to the calculations, respectively, as follows:

- The calculation of the subsystem (A) or (A+L) using a high-level potential. The atoms must be listed in exactly the same order as defined in `structure.dat`, i.e., the list of atoms in the primary region followed by the list of link atoms.
- The calculation of the subsystem (A) or (A+L) using a low-level potential. The atoms must be listed in exactly the same order as defined in `structure.dat`, i.e., the list of atoms in the primary region followed by the list of link atoms.
- The calculation of the system (A+B) using a low-level potential. The atoms must be listed in exactly the same order as defined in `structure.dat`, irrespective of the region.

Prepare structure.dat.. The file `structure.dat` must be in the following format for a water dimer:

```
6
BOHR
O   -1.13151   1.10684  -0.67760   1 A
H   -2.30412  -0.29737  -0.45427   2 A
H    0.45453   0.27505  -0.94620   2 A
O   -5.29474  -1.87202   0.09742   1 B
H   -5.90817  -0.13639   0.18675   2 B
H   -6.79303  -2.84927   0.37973   2 B
```

where “A” and “B” denote the primary and secondary regions, respectively. Alternatively, region “B” can be abbreviated as follows:

```
6
BOHR
O   -1.13151   1.10684  -0.67760   1 A
H   -2.30412  -0.29737  -0.45427   2 A
H    0.45453   0.27505  -0.94620   2 A
O   -5.29474  -1.87202   0.09742   1
H   -5.90817  -0.13639   0.18675   2
H   -6.79303  -2.84927   0.37973   2
```

The file `structure.dat` must be in the following format for ethane:

```
8
ANGSTROM
C     2.14200   1.39500  -8.93200   1 AL 5 0.70 H
H     1.60400   0.76000  -8.26000   2 A
H     1.74500   2.38800  -8.88000   2 A
H     2.04300   1.02400  -9.93000   2 A
C     3.63100   1.41600  -8.53700   1 B
H     4.16900   2.05100  -9.21000   2 B
H     3.73100   1.78800  -7.53900   2 B
H     4.20300  -0.01200  -8.61200   2 B
```

“AL” indicates the atom in the primary region (C_1) connected to a link atom (L). “5” denotes the atom in the secondary region (C_5) that is linked to C_1. “0.70” is the equilibrium bond-length ratio between C_1–L and C_1–C_5. “H” is the atomic element of the link atom, L = H. Region “B” can also be abbreviated in this case. The file `structure.dat` must be in the following format for butane:

```
14
ANGSTROM
C     2.14200   1.39500  -8.93200   1 B
H     1.60400   0.76000  -8.26000   2 B
H     1.74500   2.38800  -8.88000   2 B
H     2.04300   1.02400  -9.93000   2 B
C     3.63100   1.41600  -8.53700   1 AL2 1 0.70 H 8 0.70 H
H     4.16900   2.05100  -9.21000   2 A
H     3.73100   1.78800  -7.53900   2 A
C     4.20300  -0.01200  -8.61200   1 B
H     3.66500  -0.64700  -7.94000   2 B
H     4.10400  -0.38400  -9.61000   2 B
C     5.69100   0.00900  -8.21800   1 B
H     6.08800  -0.98300  -8.27000   2 B
H     5.79100   0.38100  -7.22000   2 B
H     6.23000   0.64400  -8.89000   2 B
```

“AL2” indicates the atom in the primary region (C_4) connected to two link atoms (L2).

Prepare input.dat.. Details are explained in the following sections.

7.2.2 ONIOM with any codes

The following keywords are important for the ONIOM method using external links with any codes other than SMASH.

```
<ipotential>
ONIOM
```

This keyword specifies the use of the ONIOM method.

```
<oniom_hi_potential>
ORCA
```

This keyword specifies the use of an external code for high-level calculations; in the present case, ORCA.

```
<oniom_lo_potential>
CFF
```

This keyword specifies the use of an external code for low-level calculations; in the present case, CFF.

```
<oniom_dat_dir>
dat_1 dat_2 dat_3
```

This keyword specifies the input directories for the high-level calculation of the subsystem, the low-level calculation of the subsystem, and the low-level calculation of the system.

```
<oniom_scr_dir>
scr_1 scr_2 scr_3
```

This keyword specifies the temporary scratch directories for the high-level calculation of the subsystem, the low-level calculation of the subsystem, and the low-level calculation of the system.

7.2.3 ONIOM with BEST method

The following keywords are important in the ONIOM method with the Boundary based on Exchange Symmetry Theory (BEST).

```
<ioption_best>
1
```

This keyword specifies the use of BEST.

```
<iobest>
1
```

This keyword specifies the central atom of BEST, in the present case, the first atom.

```
<ikind_best>
1
```

This keyword specifies the atomic kind of BEST, in the present case, the first kind.

```
<fc_best>
1.0
```

This keyword specifies the force constant of BEST, in the present case, 1.0 hartree/bohr^2.

```
<eps_best>
1.e-16
```

This keyword specifies the energy cutoff of BEST, in the present case, 10^{-16} hartree.

8 Data analysis

The PIMD code prints the trajectory file, `trj.out`. The code **`calc.x`** can perform the following analyses using the information from `trj.out`.

- Visualization of trajectory: Transform the data into the xyz format that can be read by visualization software, such as Visual Molecular Dynamics (VMD): `http://www.ks.uiuc.edu/Research/vmd`.
- Statistical analysis: Create lists and distributions of the geometrical variables specified by the user. The variables include bond lengths (interatomic distances), bond angles, dihedral angles, out-of-plane distances, etc. These variables can be specified either by a set of specific atoms or by a set of specific atomic species.

The code reads three files, which should be placed in the execution directory: the trajectory file, `trj.out`; the input file to be edited by the user, `calc.dat`; and the input file that contains the list of all default parameters, `calc_default.dat`. The file `calc_default.dat` can be copied from the directory *examples*.

8.1 Visualization of trajectory

The trajectory file `trj.out` can be transformed into xyz format that can be read by visualization software. Executing **`calc.x`** writes the converted trajectory to a file named `calc.xyz`. **NOTE:** The data in `trj.out` are in bohrs, while those in `calc.xyz` are in angstroms.

As an example, a set of input files for a water molecule is available. Copy it to the directory *run*:

```
> cp -r examples/01_basics/H2O/calc_pimd run
```

and enter that directory:

```
> cd run/calc_pimd
```

Let us examine the keywords present in `calc.dat` one by one.

```
<iboundary>
0
```

This keyword sets the boundary condition, as in `input.dat`. In the present case, "0" means a free boundary condition.

```
<nbead>
32
```

This keyword sets the number of beads, as in `input.dat`. In the present case, 32 beads are chosen.

Now come the keywords specific to `calc.dat`.

```
<iconf_ini_calc>
1
```

This keyword sets the first configuration to be used for the analysis. In this case, "1" means the first configuration of `trj.out`.

```
<iconf_fin_calc>
-1
```

This keyword sets the last configuration to be used for the analysis. In this case, "-1" means the last configuration of `trj.out`.

```
<iprint_std_calc>
10
```

This keyword sets the standard output. In this case, the standard output is printed every 10 configurations.

```
<jprint_xyz>
1
```

This keyword sets the print interval for xyz data. In this case, "1" means every step.

```
<jorigin_xyz>
2
```

This keyword sets the origin of the system. In this case, "2" means the origin is fixed in space.

```
<jformat_xyz>
2
```

This keyword sets the format of the xyz file. In this case, "2" means that all bead configurations are printed together.

To run, type

```
> calc.x
```

then the xyz trajectory will be printed to the file `calc.xyz`.

NOTE: When using the **`calc.x`** command, the file `calc.xyz` is always overwritten if it already exists.

8.2 Statistical analysis

For statistical analysis using **`calc.x`**, all the keywords in Sec. 8.1 are important. Additionally, the following keywords are important.

```
<iprint_calc>
1
```

This keyword sets the print interval of the statistical analysis. In the present case, "1" means every step.

```
<ncalc>
5
  lin.atom.dens   1  2         'calc.12a.dens'
 angl.atom.dens   2  1  3      'calc.213.dens'
  lin.atom.list   1  2         'calc.12a.list'
  lin.spec.list   1  2         'calc.12s.list'
 angl.atom.list   2  1  3      'calc.213.list'
```

This keyword sets the data to be analyzed. In this case, five types of analyses are specified.

- The density with respect to the distances between atoms #1 and #2 is printed to the file `calc.12a.dens`.
- The density with respect to the angles defined by atoms #1, #2, and #3 is printed to the file `calc.213.dens`.
- The list of interatomic distances of atoms #1 and #2 is printed to the file `calc.12a.list`.
- The list of interatomic distances with respect to atomic species #1 and #2 is printed to the file `calc.12s.list`.
- The list of bond angles with respect to atoms #1, #2, and #3 is printed to the file `calc.213.list`.

To run, type

```
> calc.x
```

After the run, the five output files `calc.12a.dens`, `calc.213.dens`, `calc.12a.list`, `calc.12s.list`, and `calc.213.list` will be created.

NOTE: When using the **`calc.x`** command, all output files specified in the keyword **<ncalc>** are always overwritten if they already exist.

9 Codes

9.1 Main codes

The main executable programs are:

- **`pimd.x`**: Serial version.
- **`pimd.mpi.x`**: Parallel (MPI) version.

The parallel version can be used for the following methods, which involve multiple replicas, beads, or images:

- Static calculations (STATIC) with multiple structures.
- Classical molecular dynamics (MD) for NVE and NVT ensembles with multiple independent trajectories.
- Normal mode analysis (NMA) with multiple geometric displacements.
- Phonon calculations (PHONON) with multiple geometric displacements.
- Path integral simulations (PIMD, CMD, RPMD, PIHMC) with multiple beads.
- String method (STRING) with multiple images.
- Metadynamics (MTD) with multiple walkers.
- Replica exchange hybrid Monte Carlo (REHMC) with multiple replicas.

A hierarchical parallelization scheme is implemented, in which force parallelization and bead parallelization can be used simultaneously. This scheme is available for first-principles potentials:

```
<ipotential> = ABINIT-MP, CP2K, NTCHEM, OPENMX, PHASE0, QE, SMASH, VASP
```

for semiempirical potentials:

```
<ipotential> = DFTB
```

and for machine-learning potentials:

```
<ipotential> = AENET, ASE, MACE, MTP, N2P2
```

Hierarchical parallelization is also applicable to all empirical potentials:

```
<ipotential> = CFF, EAM, POL, ADP, PAIR, TERSOFF
```

By default, hierarchical parallelization follows this rule. Let P be the number of beads and C the number of processors.

- If $P < C$, all P beads are computed in parallel, each bead using C/P processors.
- If $P > C$, C beads are computed in parallel, and the calculation is repeated sequentially P/C times.

These behaviors can be controlled in `input.dat` using the keywords **<np_beads>** (number of beads computed simultaneously) and **<np_force>** (number of processors assigned to each bead).

Only bead parallelization is available for the following first-principles potentials:

```
<ipotential> = GAMESS, G98, G03, G09, G16, MOLPRO, ORCA, TURBOMOLE
```

and for the semiempirical potential:

```
<ipotential> = MOPAC
```

In these cases, the number of processors must not exceed the number of beads specified by **<nbead>**.

The MPI execution command depends on the MPI implementation. Typically, the calculation can be started as follows:

```
> mpirun -np 4 -machinefile host.list pimd.mpi.x >> monitor.out &
```

where the file `host.list` contains a list of host names:

```
machine_name_for_process_1
machine_name_for_process_2
machine_name_for_process_3
machine_name_for_process_4
```

Please refer to the documentation of your MPI implementation for further details.

9.2 Codes for post-processing

The following code is provided for post-processing:

- `calc.x`: Statistical analysis of simulation results.

See Section 8.2 for details.

9.3 Subsidiary codes

The subsidiary codes are:

- `run_molpro.x`: Interface to run the MOLPRO code.
- `run_vasp.x`: Interface to run the VASP code.

These programs are called automatically from the main codes only when required.

9.4 Utility codes

The following utility programs are included:

- `convert_tinker.x`: Converts TINKER force-field files into the PIMD input format.
- `prep_liquid.x`: Generates random initial configurations for molecular liquids and creates TINKER xyz files.

See Section 10 for details.

9.5 External calls

The PIMD code can be invoked externally from other programs. For example, the `external` keyword in GAUSSIAN can be used to obtain partially optimized structures or transition-state geometries. In this case, the program `gau.x` acts as an interface between GAUSSIAN and `pimd.x` (the serial version only). A simple example for a water molecule is provided in the directory `examples/99_utilities/gau_external_h2o`.

Part V

Classical Force Fields, Input, and Output

10 Classical force field (CFF)

The <ipotential> = CFF potential is the supported classical force-field interface in the PIMD code. It replaces the former standalone MM input path and reads a LAMMPS-compatible description of the system and force-field coefficients. A CFF calculation normally uses the following files in the execution directory:

- input.lammps: LAMMPS-style commands and style selection.
- The data file selected by read_data in input.lammps (default data.lammps): LAMMPS atom types/charges and bonded topology; LAMMPS-compatible coordinates, masses, and box metadata may also be present but do not replace the PIMD structure/masses/cell.
- parm.lammps: coefficient commands, typically included from input.lammps.
- cmap.dat: CMAP grid data when a CMAP term is used.
- structure.dat and input.dat: ordinary PIMD structure and control input.

The old mm.dat runtime format is no longer read. If <ipotential> = MM is supplied, the program terminates with a migration error and instructs the user to use CFF.

10.1 LAMMPS-compatible input

The CFF interface is intentionally a *LAMMPS-compatible force-field reader*, not a complete LAMMPS input-script interpreter. The same input.lammps can contain ordinary LAMMPS run-control commands so that it remains convenient for independent LAMMPS checks, but PIMD reads only the subset that is needed to define the force field. In particular, the trajectory, ensemble, masses, coordinates, and PIMD boundary treatment remain under control of the ordinary PIMD files. An unsupported command that would change the force field is rejected rather than silently ignored.

10.1.1 Files and division of responsibility

A typical CFF calculation separates the information as follows.

File	Information used by PIMD/CFF
input.dat / pimd.toml	PIMD method, ensemble, temperature, bead number, time step, parallelization, boundary condition, and other simulation controls.
structure.dat	PIMD atomic coordinates, atomic masses, and the simulation cell used by the PIMD dynamics.
input.lammps	CFF style selection, cutoffs, electrostatics, exclusions/scaling, read_data, and included coefficient files.
data file from read_data (default data.lammps)	Atom types and charges, bonded topology, optional CMAP topology, type counts, and optionally coefficient sections. Coordinates and Masses present for LAMMPS compatibility are not used as the PIMD coordinates or PIMD masses.
parm.lammps	A convenient included file containing pair_coeff, bond_coeff, angle_coeff, dihedral_coeff, and improper_coeff.

The filename is not required to be literally data.lammps; read_data structure.data, for example, makes structure.data the required CFF topology file. If no read_data command is present, the CFF reader uses its default name data.lammps. Diagnostics therefore follow the command rather than hard-coding one filename.

Thus, for example, changing a LAMMPS mass command does not change the mass used by PIMD. Likewise, boundary, timestep, velocity, fix nvt, and run do not replace the corresponding PIMD controls. They may be left in a shared LAMMPS input file, but the PIMD calculation is controlled by input.dat (or the TOML source used to generate it) and structure.dat.

10.1.2 Commands interpreted by CFF

The following commands are currently interpreted by the CFF reader. Command names are case-insensitive. Recursive include is supported up to 16 nested levels, and a physical line may be continued with &. LAMMPS

variable substitution (for example `${name}`) is not performed in an active CFF command; use explicit numerical values there.

Command	CFF behavior
`units real`	Required. Other LAMMPS unit styles are rejected.
`atom_style full|charge|atomic`	Selects how atom type and charge are read from the `Atoms` section of `data.lammps`.
`read_data FILE`	Selects the topology/type file. The number of atoms must equal the number in the PIMD structure.
`include FILE`	Recursively reads commands and coefficients from another file.
`bond_style`, `angle_style`, `dihedral_style`, `improper_style`	Select the bonded functional forms listed below. Supported `hybrid` combinations are selected by giving the sub-style in each coefficient line.
`pair_style`	Selects the non-bonded form and global cutoff(s).
`pair_modify`	Supports `mix arithmetic` and `mix geometric`. `shift no` and `tail no` are accepted; `yes` is rejected.
`special_bonds`	Defines 1–2, 1–3, and 1–4 LJ/Coulomb weights. The `amber` and `charmm` presets and explicit `lj/coul`, `lj`, and `coul` triplets are supported.
`kspace_style ewald|pppm ACCURACY`	Selects long-range electrostatics for periodic CFF. The first numerical argument is used as the requested accuracy.
`neighbor SKIN ...`	The first numerical argument is used as the CFF short-range Verlet-list skin. The LAMMPS neighbor style token is retained for compatibility but is not a separate CFF algorithm selection.
`neigh_modify ...`	Neighbor-list-only options may be present. `exclude` and `include` are explicitly rejected because they would change the physical interaction graph.
`fix ID all cmap FILE`	Enables CMAP and reads the CMAP parameter grid from `FILE`. Other active force-field fixes are not supported.
`pair_coeff`, `bond_coeff`, `angle_coeff`, `dihedral_coeff`, `improper_coeff`	Define force-field coefficients in `input.lammps`, an included file, or the corresponding coefficient section of `data.lammps`.

The coefficient type field may be a single type or a LAMMPS-style range such as `*`, `3*`, `*8`, or `3*8`. For LJ styles, all diagonal `pair_coeff i i` values must be available; missing unlike pairs are generated with the selected mixing rule. An explicitly supplied unlike `pair_coeff i j` overrides mixing for that pair. Morse and Buckingham styles require all type pairs explicitly.

10.1.3 Supported styles and coefficient forms

The implemented functional forms and accepted coefficient records are shown below. Energies and distances follow LAMMPS `units real`.

Category	Style	Coefficient form after type index/indices
bond_style	harmonic	K r0
	class2	r0 K2 K3 K4
	hybrid	sub-style followed by one of the forms above
angle_style	harmonic	K theta0
	charmm	K theta0 KUB rUB
	hybrid	sub-style followed by one of the forms above
dihedral_style	harmonic	K d n
	fourier	m K1 n1 d1 ... Km nm dm (up to 16 terms)
	opls	K1 K2 K3 K4
	charmm	K n d weight
	hybrid	sub-style followed by one of the forms above
improper_style	harmonic	K chi0
	cvff	K d n
	hybrid	sub-style followed by one of the forms above
pair_style	lj/cut	epsilon sigma [cut]
	lj/cut/coul/cut	epsilon sigma [cutLJ [cutCoul]]
	lj/cut/coul/long	epsilon sigma [cutLJ]
	lj/charmm/coul/long	epsilon sigma [epsilon14 sigma14]
	morse	D0 alpha r0 [cut]
	buck	A rho C [cut]
	buck/coul/long	A rho C [cut]
kspace_style	ewald, pppm	requested accuracy in the style command

For periodic calculations, a coul/long pair style requires kspace_style ewald or pppm. For a shared input that is also intended to run in LAMMPS, the per-pair coefficient of lj/cut/coul/long should contain at most the optional LJ cutoff; the Coulomb cutoff is global in the pair_style command. The current lj/cut/coul/cut implementation is restricted to free-boundary CFF. The availability of a style in this table means that the PIMD CFF evaluator implements that force form; it does not imply support for arbitrary LAMMPS suffixes, pair_style hybrid, or other LAMMPS packages.

Many-body potentials remain separate PIMD potential interfaces rather than being evaluated by the molecular CFF pair kernel. However, <ipotential> = EAM, <ipotential> = ADP, and <ipotential> = PAIR now obtain their active style and potential file from input.lammps; EAM/alloy setfl, ADP extended-setfl, and LAMMPS pair tables are read directly. The separate <ipotential> = METALWATER interface is an exception for its metal–metal subpotential: it can read pair_style eam/alloy and pair_coeff * * FILE ELEMENT... from input.lammps, with nonmetal/water atom kinds mapped to NULL. This does not route GAL21 itself through the CFF pair evaluator.

10.1.4 Commands accepted for a shared LAMMPS file but not executed

The following commands can remain in the file so that the same input is useful for LAMMPS validation. CFF accepts them but does not use them to control the PIMD trajectory:

```
dimension  boundary  thermo  thermo_style  thermo_modify
log  print  variable  reset_timestep  mass  timestep  newton
processors  comm_modify  echo  group  region
compute  compute_modify  uncompute
dump  dump_modify  undump
restart  write_restart  write_data  write_dump
velocity  run_style  min_style  min_modify  minimize
suffix  package  timer  info
```

The common run-control fixes nve, nvt, npt, nph, nvt/sllod, langevin, temp/berendsen, temp/csvr, temp/rescale, press/berendsen, and momentum are also accepted but ignored by CFF. run N is checked only for a non-negative numeric step count; PIMD does not execute those LAMMPS steps. This distinction is important: *the PIMD ensemble and integrator are never selected by a LAMMPS fix or run command.*

10.1.5 data.lammps: topology actually used by CFF

The reader uses the header counts for atoms, bonds, angles, dihedrals, impropers, and their type counts. It reads the following sections: Atoms, Bonds, Angles, Dihedrals, Impropers, and the PIMD CMAP extension section CMAP. The atom records are interpreted as

```
# atom_style full
id  molecule-ID  atom-type  charge  x  y  z  ...

# atom_style charge
id  atom-type  charge  x  y  z  ...

# atom_style atomic
id  atom-type  x  y  z  ...
```

CFF uses the atom ID, type, charge, and bonded topology from these records. The coordinate fields are deliberately not used to replace the PIMD coordinates. Likewise, a Masses section and LAMMPS box bounds may be present for LAMMPS compatibility, but PIMD masses and the PIMD cell come from the PIMD input/structure path. The atom count in data.lammps must match the PIMD structure exactly.

Coefficients may alternatively be stored in LAMMPS Pair Coeffs, PairIJ Coeffs, Bond Coeffs, Angle Coeffs, Dihedral Coeffs, and Improper Coeffs sections. Coefficients read later from input.lammps or an included file overwrite values read from data.lammps.

10.1.6 1–2, 1–3, and 1–4 exclusions/scaling

Intramolecular non-bonded exclusions are controlled by special_bonds, using the bond graph in data.lammps. The three numbers are the weights for 1–2, 1–3, and 1–4 atom pairs, respectively. LJ and Coulomb weights may be given together or independently, for example

```
special_bonds lj/coul 0.0 0.0 0.5
special_bonds lj 0.0 0.0 0.5
special_bonds coul 0.0 0.0 0.833333333333
special_bonds amber
special_bonds charmm
```

The CFF amber preset uses LJ weights 0 0 0.5 and Coulomb weights 0 0 1/1.2; charmm sets the ordinary special-pair weights to zero because CHARMM 1–4 LJ treatment can be carried by the CHARMM dihedral/pair path. The topology modifiers angle, dihedral, and one/five are accepted only as ... no; options that would redefine the special-pair graph are rejected.

10.1.7 PIMD-only CFF extensions

Features without a direct command in the supported LAMMPS subset are written as comments, so LAMMPS itself ignores them while PIMD/CFF reads them during the coefficient pass:

```
# PIMD_CFF tip4p O H gamma
# PIMD_CFF water_model NAME
# PIMD_CFF lj_switch rin rout
# PIMD_CFF skin_lj value
# PIMD_CFF skin_ewald value
# PIMD_CFF polarization induced
# PIMD_CFF polarization_solver direct|bicgstab [eps [maxiter]]
# PIMD_CFF polarizability type alpha [kind]
# PIMD_CFF damping CC|CD|DD kind1 kind2 none|zero|lin|exp|gau|oss [parameters]
# PIMD_CFF ewpol accuracy ratio
```

The tip4p line defines the O label, H label, and the M-site position parameter γ. lj_switch supplies a switching interval when one is needed by a converted force field. The polarization records select the induced-

dipole model, solver, polarizabilities, damping, and polarizable-Ewald parameters. Unsupported extension names terminate with an explicit error.

10.1.8 Example: water and the auto-generated q-TIP4P/F files

The built-in `WATER` potential writes `input.lammps`, `data.lammps`, and `parm.lammps` automatically on its first setup. For periodic q-TIP4P/F the generated `input.lammps` has the following structure:

```
units real
atom_style full
boundary p p p
bond_style class2
angle_style harmonic
pair_style lj/cut/coul/long 9.6442553630
pair_modify mix arithmetic
special_bonds lj 0.0 0.0 1.0 coul 0.0 0.0 1.0
kspace_style ewald 1.0e-8
# PIMD_CFF tip4p O H 0.73612000
# PIMD_CFF lj_switch 8.1440378621 9.6442553630
read_data data.lammps
include parm.lammps
neighbor 2.0 bin
thermo 1
run 0
```

This example illustrates the separation between standard LAMMPS syntax and PIMD-only metadata. `bond_style class2` is used to represent the q-TIP4P/F quartic O–H stretching potential, while the angle term is harmonic. The 1–2 and 1–3 LJ/Coulomb weights are zero, so intramolecular non-bonded interactions implied by the O–H bonds and H–O–H angle are excluded; 1–4 weights are one. The virtual charge site is not inserted as an extra atom in `data.lammps`; instead the `PIMD_CFF tip4p` record gives the O label, H label, and $\gamma = 0.73612$ needed to construct the M site from the real atomic coordinates. The LJ switching window is likewise carried in a PIMD-only comment.

SPC/F and RWK use the same generated-file workflow. Their model-specific terms that have no direct representation in the supported LAMMPS subset are marked with `PIMD_CFF water_model` metadata and evaluated by the built-in `WATER` force routine. Thus the generated files are useful for topology, exclusions, electrostatics, and parameter inspection, but the selected WATER model is still controlled by the PIMD `WATER` interface.

A separate generic CFF water example is distributed in

```
examples/02_path_integral_dynamics/water_64/pimd_nvt_cff
```

and demonstrates the same `read_data`/`include`/ `special_bonds`/`kspace_style` workflow for a converted molecular force field.

10.1.9 Example: alanine dipeptide (dialanine)

The distributed alanine-dipeptide example

```
examples/01_basics/dialanine/md_cff
```

shows why style and coefficient support matters for a molecular force field:

```
units real
atom_style full
boundary f f f
bond_style harmonic
angle_style hybrid charmm harmonic
dihedral_style fourier
improper_style harmonic
```

```
pair_style lj/cut/coul/long 13.229431225
# PIMD_CFF lj_switch 10.58354498 13.229431225
pair_modify mix arithmetic
special_bonds lj 1.0 1.0 1.0 coul 0 0 1
read_data data.lammps
include parm.lammps
neighbor 2.0 bin
thermo 1
run 0
```

The topology is read from `data.lammps`. In `parm.lammps`, a hybrid angle coefficient states the sub-style explicitly, for example

```
angle_coeff 1 harmonic 79.9999319316351 121.0
angle_coeff 3 charmm 32.9999714583012 109.5 29.999974315927 2.16299999296395
```

while the Fourier torsions use records such as

```
dihedral_coeff 1 3 1.59999862272 1 0 2.499997848 2 180 0 0 0
```

where `3` is the number of Fourier terms and each term is `K n d`. The first two shown terms are non-zero; a zero-amplitude term is retained by the converter so that the generated representation follows its source force field exactly. This example also demonstrates a PIMD-only LJ switching window while keeping the rest of the file valid LAMMPS syntax.

10.1.10 Example: EAM/alloy input for a Ni–H system

The LAMMPS-style input workflow is also used by the separate PIMD `EAM` backend. This is not a CFF pair style: the many-body EAM energy is evaluated by the PIMD EAM routines, while `input.lammps` selects the native DYNAMO setfl file and maps PIMD atom kinds to elements. A complete Ni–H example is distributed in

```
examples/02_path_integral_dynamics/NiH/pimd_npt_eam
```

Its `input.lammps` is

```
units metal
atom_style atomic
boundary p p p
read_data data.lammps
pair_style eam/alloy
pair_coeff * * potential.eam.alloy Ni H
neighbor 2.0 bin
run 0
```

The potential file is a native multi-element setfl file whose element line contains `Ni H`. The trailing element names in `pair_coeff` map PIMD/LAMMPS atom type 1 to Ni and atom type 2 to H; the mapping order is therefore significant. Standalone PIMD `EAM` and `ADP` calculations require every active atom kind to map to an element present in the potential file, and do not accept `NULL` mappings. The `NULL` placeholder is used only by the METALWATER/GAL21 route for the water types, whose interactions are supplied by GAL21 and the water model rather than by EAM.

The corresponding TOML selects the EAM backend explicitly, for example

```
[files]
structure   = "structure.dat"
force_field = "input.lammps"

[potential]
type = "EAM"
```

The setfl radial-density and radial-pair meshes are read independently, so `Nrho` and `Nr` need not be equal. Before a run, `pimd-prep check` verifies `units metal`, the `eam/alloy` style, the number and order of element mappings, the setfl element list, and consistency of the shared structure/data files. Thus a swapped mapping such as `H Ni`, a `NULL` standalone-EAM type, or a missing setfl element is rejected before the dynamics starts.

10.1.11 Unsupported input is fail-closed

"LAMMPS-compatible" therefore means the documented subset above. CFF does not silently reinterpret unsupported active physics. Examples that terminate with an explicit error include an unknown `pair_style`, an unsupported active `fix`, `pair_modify shift yes`, `pair_modify tail yes`, `neigh_modify exclude/include`, an active command containing unresolved LAMMPS variables, missing required coefficients, or a periodic `coul/long` calculation without a supported `kspace_style`. This fail-closed behavior is intentional so that a LAMMPS input file cannot silently produce a different physical model in PIMD.

10.2 Static preflight and cross-file consistency with pimd-prep

With `pimd-prep` 0.9.7, CFF input is checked before PIMD starts. The front end mirrors the supported CFF parser rather than attempting to be a general LAMMPS parser. It validates style names and coefficient schemas, LAMMPS type selectors, hybrid sub-styles, `pair_modify`, `special_bonds`, k-space syntax, and coefficient coverage. Coefficient sections in `data.lammps` are combined with command-file coefficients; later definitions are reported as overrides.

The same check validates topology header/section counts, atom IDs, atom and bonded type ranges, and bonded/CMAP atom references. When `structure.dat` is present, the atom count, atom order/coordinates, boundary condition, and for periodic systems the triclinic cell are checked across files. Periodic-image shifts and one common rigid translation are allowed, but a stale geometry or atom-ID/order mismatch is rejected. Optional species annotations and Masses records are also checked. H, D, T, and Mu are deliberately treated as one hydrogen chemical family for these force-field checks; PIMD still assigns their distinct dynamical masses.

The parsed state can be inspected with

```
pimd-prep show pimd.toml --force-field
pimd-prep explain cff-styles
```

so style/coeff errors and cross-file inconsistencies can be found before a Fortran run is launched.

10.3 CFF with XMPI

CFF can be used with `<xmpi>`="ON" for the XMPI-enabled methods listed in Section 11.4.228. Free-boundary CFF and periodic direct Ewald are supported, including the shared TIP4P path. The following restrictions remain in the present implementation:

- CFF PME (`kspace_style pppm`) is not yet supported under XMPI; use direct Ewald or ordinary MPI.
- Induced-polarization CFF under XMPI currently supports only the free-boundary direct polarization solver. Periodic/Ewald polarization and the BiCGSTAB polarization solver require ordinary MPI.
- The CFF XMPI wrapper preserves the exact total virial, but the historical CFF-MPI kernel does not expose a native per-bead virial. Variable-cell PIMD integrators are therefore not XMPI-enabled.

These cases fail explicitly rather than falling back to a different physical model.

10.4 TINKER conversion

The utility `convert_tinker.x` converts a TINKER xyz file (`*.txyz`) and a TINKER parameter file (`*.prm`) directly to the CFF files described above. It is the supported migration path for TINKER-style force fields; it

does not create mm.dat. The command-line interface is

```
> convert_tinker.x INPUT PARAM RIN ROUT
```

where RIN and ROUT are the inner and outer Lennard–Jones cutoff distances in bohr. The converter writes input.lammps, data.lammps, parm.lammps, structure.dat, and input.dat; cmap.dat is also written when CMAP data are present. See Section 10.5 for a complete example.

10.5 How to convert TINKER files

IMPORTANT: The user is solely responsible for all issues related to the TINKER license. PIMD does not provide any TINKER license.

The utility **convert_tinker.x** converts TINKER xyz and parameter files directly to the LAMMPS-compatible CFF input used by **<ipotential>** = CFF. A compact water example is distributed in **examples/99_utilities/convert_tinker_h2o**:

```
> cp -r examples/99_utilities/convert_tinker_h2o run
> cd run
> convert_tinker.x h2o.txyz water_min.prm 20.0 25.0
```

The four arguments are

- the TINKER xyz-format input file (*.txyz);
- the TINKER parameter file (*.prm);
- the inner Lennard–Jones switching distance in bohr;
- the outer Lennard–Jones cutoff distance in bohr.

On success, the converter writes

- input.lammps,
- data.lammps,
- parm.lammps,
- structure.dat,
- input.dat, and
- cmap.dat when the parameter set contains CMAP data.

The old eight-argument invocation is accepted only as a transition aid; its legacy mm.dat/centroid output arguments are ignored. New scripts should use the four-argument form above. The generated input.lammps selects the required LAMMPS-compatible styles and includes parm.lammps; the topology and atomic data are stored in data.lammps. The files can therefore be inspected and version-controlled in the same style as ordinary LAMMPS input. The example's run.sh also performs a small CFF regression run.

10.6 Preparation of initial structure

The **prep_liquid.x** code is a utility code for the generation of random structures of molecular liquids and the TINKER xyz file.

This code creates the geometry files for the PIMD code in three different forms: centroid.dat (for initiations with **<input_style>** = OLD), structure.dat (for initiations with **<input_style>** = NEW), and geometry.ini (for restarts). It also creates the TINKER xyz file, structure.txyz, to prepare the classical force field.

This code reads the keywords from the file input.dat, which should be prepared by the user. The following is an example of the file input.dat to prepare the simulation of aqueous hexanediol.

```
<natom>
```

```
280
```

This keyword sets the number of atoms, in this case, “280” atoms, with 86 water molecules (3 atoms each) and one hexanediol molecule (22 atoms).

```
<nspec>
8
O     15.99900  86   63  2
H      1.00800 172   64  1
H      1.00800   2   97  1
O     15.99900   2   96  2
C     12.01100   2   81  4
C     12.01100   2  100  4
C     12.01100   2   80  4
H      1.00800  12   85  1
```

This keyword sets the number of species, followed by the atomic symbol, the atomic mass, the number of atoms of each species, the atomic type of the classical force field (in the present case `opls_aa.prm`), and the bond order. The atomic type and the bond order affect only the TINKER input file (txyz). In this case, there are “8” species.

```
<iboundary>
1
  26.589    0.000    0.000
   0.000   26.589    0.000
   0.000    0.000   26.589
```

This keyword sets the boundary condition and the simulation box. In this case, periodic boundary conditions are applied, and the simulation box is cubic with a side length of 26.589 bohr. For new TOML input, define the rigid reference geometry and molecule instances in `[molecular_components]` and `[molecular_components.component.N]`; see Section 2. The following **<components>** and **<molecules>** records are the legacy engine representation.

```
<components>
 2
 MOL    22     GENERAL
 -0.93099  -4.50041   4.00409
  5.14070   5.24963   3.62029
 -2.29140  -3.36393   3.96554
  3.33204   5.23867   3.69512
  0.36177   1.74551   2.75309
 -0.00389  -0.45356   0.96768
 -2.44787  -1.91602   1.71015
  2.74906   3.32740   1.96319
 -3.38480  -3.57747  -0.43337
  2.37017   4.79893  -0.48344
  0.69625   5.79104  -0.37705
  3.89386   6.10662  -0.98025
  1.86864   3.71234  -2.06514
  0.89071   1.18086   4.72407
 -1.27549   2.94303   2.68714
 -0.25390   0.24564  -0.88274
  1.53510  -1.80849   1.13095
 -3.80829  -0.46565   1.98775
  4.20755   1.93864   1.77308
 -5.47767  -3.92499  -0.29825
 -3.16181  -2.45213  -2.13450
```

```
 -2.49738  -5.43979  -0.67147
 H2O     3     GENERAL
  0.00000   0.00000   0.00000
  1.48911   1.16342   0.00000
 -1.48911   1.16342   0.00000
```

This keyword sets each molecular component. In this case, there are two components in the system: "MOL" and H2O. "MOL" is a non-linear molecule composed of 22 atoms, while "H2O" is a non-linear molecule composed of 3 atoms. The atomic coordinates in the molecular frame are given after "MOL" and "H2O".

```
<molecules>
87
MOL   259  260  261  262  263  264  265  266  267  268
      269  270  271  272  273  274  275  276  277  278
      279  280
H2O     1   87  173
H2O     2   88  174
H2O     3   89  175
...    ..  ...  ...
...    ..  ...  ...
...    ..  ...  ...
H2O    84  170  256
H2O    85  171  257
H2O    86  172  258
```

This keyword gives the list of atoms in the molecules. The numbering should be consistent with that given by the keyword <nspec>.

To run, type

```
> prep_liquid.x
```

```
-------------------------------------------------
  atom 1  atom 2  bond order   dist/au  r1+r2/au
-------------------------------------------------
       1      87     1     2   1.88971   3.45000
       1     173     2     2   1.88971   3.45000
       2      88     1     2   1.88971   3.45000
       .     ...     .     .   .......   .......
       .     ...     .     .   .......   .......
       .     ...     .     .   .......   .......

Created: structure.dat.

Created: centroid.dat.

Created: geometry.ini.

Created: structure.dat.

Normal termination of PREP_LIQUID.
```

Confirm that the files are created properly. The file `geometry.xyz` is in xyz format and can be visualized. The file `structure.txyz` can be passed to **convert_tinker.x** together with a TINKER parameter file to create the CFF files `input.lammps`, `data.lammps`, and `parm.lammps` as described in Section 10.5.

NOTE: If one gets the following message, there are ways to fix it.

```
Warning - Number of bonds do not match:  4 and  3 for atom:   265.
```

This warns that the number of neighboring atoms of atom 265 (= 3) did not match the bond order of atom 265 (= 4). Therefore, one can

- check if the bond order is correct.
- tune the atomic radius in <symbol> in input_default.dat, which changes the values of the cutoff radii $(r1 + r2)$ and thus affects the number of neighboring atoms.
- change manually the TINKER input file (txyz).

11 Keywords

This section documents the input tags used by PIMD. For version 2.8.0, the manual was checked against the active literal tags read by the Fortran source, including tags in potential-specific files and LAMMPS-compatible input.lammps, plus hob.dat and oss.dat. The source-audited supplement at the end of this section records tags that were not described explicitly in the previous manual.

Earlier editions divided the main input.dat tags into "necessary", "important", "specific", and "optional" groups. Those labels mixed two different ideas—whether a tag is logically required and how frequently a user is likely to set it—and became increasingly misleading as defaults and the structure.dat input format evolved. The version 2.8.0 manual therefore uses a functional classification instead:

- **Core run-definition keywords**: select the method, ensemble, potential, and legacy/special-case system-size information.
- **Common simulation-control keywords**: time step, temperature, trajectory length, bead count, output intervals, thermostat/barostat controls, and other settings shared by many workflows.
- **Method-, potential-, and interface-specific keywords**: controls that become relevant only for a particular algorithm, potential, external program, or machine-learning backend.
- **Conditional and auxiliary keywords**: structured blocks or secondary controls that are required only when the corresponding feature is enabled.

The historical LaTeX labels necessary, important, specific, and optional are retained internally so that existing cross-references remain valid; they no longer imply a strict priority ranking.

NOTE: Each keyword must start from the first column of the line. Do not place any spaces or characters before a keyword. If a keyword is indented or preceded by extra characters, it will not be recognized correctly by the code.

11.1 Run-definition keywords in input.dat

These keywords define the basic identity of a run. Not every entry in this group must appear explicitly in every modern input file: in particular, <natom> and <nspec> are normally obtained from structure.dat when <input_style> = NEW is used. This grouping therefore describes function, not unconditional necessity.

11.1.1 <ensemble>

Used in ALL methods. This keyword specifies the type of thermodynamic ensemble. Input: one character string on the first line. The available options are as follows.

- <ensemble> = NONE: unspecified.
- <ensemble> = NVE: constant volume and energy.
- <ensemble> = NVT: constant volume and temperature.
- <ensemble> = NPT: constant pressure and temperature.
- <ensemble> = NTT: constant tension and temperature.

- <ensemble> = NPH: constant pressure and enthalpy.
- <ensemble> = NTH: constant tension and enthalpy.

The applicable conditions are as follows.

- <method> = MD: NVE or NVT.
- <method> = PIMD: NVE, NVT, NPH, NTH, NPT, or NTT.
- <method> = PIHMC: NVT, NPT, or NTT; the source explicitly rejects NVE, NPH, and NTH.
- <method> = CMD: NVE.
- <method> = RPMD: NVE or NVT.
- <method> = TRPMD: NVE or NVT.
- <method> = BCMD: NVE or NVT.
- <method> = MTS, MTD, TASS: NVT.
- <method> = REHMC: NVT or NPT.
- <method> = REMC: NVT.
- <method> = ROTOR: NVE or NVT.
- <method> = AFED: NVT.

For NVE and NVT the simulation cell is fixed and no barostat is used. Pressure/barostat controls apply to NPH/NPT, while tension/barostat controls apply to NTH/NTT. The TOML front end enforces this ensemble dependence and maps its single [barostat].cell_mode setting to the corresponding legacy box-type keyword. Default: “NONE”.

11.1.2 <ipotential>

Used in ALL methods. This keyword specifies the potential. Input: one character string on the first line. Choose one of the following standard potentials.

- <ipotential> = ABINIT-MP: ABINIT-MP.
- <ipotential> = ACCELNET: AccelNet inference with AENET or N2P2 models (Section 6.4.2).
- <ipotential> = ADP: angular dependent potential.
- <ipotential> = AENET: AENET.
- <ipotential> = ASE: ASE calculator interface.
- <ipotential> = CP2KLIB: CP2K (internal link).
- <ipotential> = DFTB: DFTB (external link).
- <ipotential> = DFTBLIB: DFTB (internal link).
- <ipotential> = EAM: embedded atom method.
- <ipotential> = G98: GAUSSIAN 98.
- <ipotential> = G03: GAUSSIAN 03.
- <ipotential> = G09: GAUSSIAN 09.
- <ipotential> = GAMESS: GAMESS.
- <ipotential> = METALWATER: GAL21.
- <ipotential> = MACE: MACE.
- <ipotential> = CFF: LAMMPS-compatible classical force field.
- <ipotential> = MOLPRO: MOLPRO.
- <ipotential> = MTP: MTP.
- <ipotential> = N2P2: N2P2.
- <ipotential> = MOPAC: MOPAC.
- <ipotential> = NTCHEM: NTCHEM.
- <ipotential> = ONIOM: ONIOM (two-layer).
- <ipotential> = OPENMX: OpenMX 4.0.1 (embedded MPI interface).
- <ipotential> = ORCA: ORCA.

- <ipotential> = OSS: Ojamae–Shavitt–Singer (OSS2) water.
- <ipotential> = PFP: Matlantis PFP.
- <ipotential> = POL: molecular mechanics (polarizable).
- <ipotential> = PAIR: tabulated pair potential.
- <ipotential> = QE: QUANTUM ESPRESSO.
- <ipotential> = QMMM: QM/MM (mechanical embedding).
- <ipotential> = SMASH: SMASH.
- <ipotential> = TIP4P: water (test).
- <ipotential> = TERSOFF: Tersoff potential.
- <ipotential> = TURBOMOLE: TURBOMOLE.
- <ipotential> = USER: user-defined.
- <ipotential> = VASP: VASP (internal link).
- <ipotential> = VASP5: VASP (external link).
- <ipotential> = VASP6: VASP version 6.
- <ipotential> = WATER: built-in q-TIP4P/F (default), SPC/F, or RWK water.
- <ipotential> = XTB: (extended tight binding).
- <ipotential> = ABINIT-MP5: ABINIT-MP version-5 interface.
- <ipotential> = CART: Cartesian model/restraint potential.
- <ipotential> = ECKART: Eckart model potential.
- <ipotential> = G16: GAUSSIAN 16.
- <ipotential> = HELIUM: helium model potential.
- <ipotential> = HOB: harmonic-plus-one-dimensional-barrier potential.
- <ipotential> = HYDROGEN: hydrogen model potential.
- <ipotential> = LIBNNP: LIBNNP neural-network interface.
- <ipotential> = MULLER: Mueller model potential.
- <ipotential> = PHASE0: PHASE/0 interface.
- <ipotential> = QUARTIC: quartic model potential.
- <ipotential> = TULLY: Tully model potential.
- <ipotential> = QEGIT: Quantum ESPRESSO git/MPI interface; recognized in the MPI force dispatcher.

Default: not defined.

11.1.3 <method>

Used in ALL methods. This keyword specifies the simulation method. Input: one character string on the first line. Choose one of the following options:

- <method> = STATIC: static calculation.
- <method> = TESTFORCE: test forces by numerical differentiation.
- <method> = TESTVIRIAL: test the virial by numerical differentiation.
- <method> = GEOOPT: geometry optimization using the limited-memory BFGS method.
- <method> = NMA: normal mode analysis.
- <method> = SD: steepest descent method.
- <method> = BOXOPT: box optimization using the limited-memory BFGS method.
- <method> = FULLOPT: simultaneous geometry and box optimization using the limited-memory BFGS method.
- <method> = ELASTIC: static elastic constants.
- <method> = MD: classical molecular dynamics.
- <method> = PIMD: path integral molecular dynamics.
- <method> = BCMD: Brownian chain molecular dynamics.
- <method> = CMD: centroid molecular dynamics.

- <method> = RPMD: ring polymer molecular dynamics.
- <method> = TRPMD: thermostatted ring polymer molecular dynamics.
- <method> = MTD: metadynamics.
- <method> = MTS: multiple time-scale path integral molecular dynamics.
- <method> = TASS: temperature accelerated sliced sampling.
- <method> = PIHMC: path integral hybrid Monte Carlo.
- <method> = REHMC: replica-exchange hybrid Monte Carlo.
- <method> = STRING: string method.
- <method> = PHONON: phonon calculation.
- <method> = TFS: nonadiabatic surface hopping dynamics (Tully's fewest switches).
- <method> = MFE: nonadiabatic mean-field dynamics (Ehrenfest mean field).
- <method> = ROTOR: molecular dynamics of rigid-body molecules.
- <method> = SCAN: consecutive static calculations.
- <method> = GAD: gentlest ascent dynamics.
- <method> = AFED: adiabatic free energy dynamics.
- <method> = TESTEWALD: developer validation of Ewald-related terms.
- <method> = TESTEWPOL: developer validation of polarizable-Ewald terms.
- <method> = TESTOM: developer validation for Onsager–Machlup machinery.
- <method> = PIOPT: path-integral optimization.
- <method> = DVR: discrete-variable-representation calculation.
- <method> = REMC: replica-exchange Monte Carlo.
- <method> = OMOPT: Onsager–Machlup path optimization.
- <method> = RPTFS: ring-polymer extension of fewest-switches dynamics.
- <method> = RPMFE: ring-polymer mean-field Ehrenfest dynamics.
- <method> = TPS: transition-path sampling.
- <method> = SHS: surface-hopping/string-related specialized dynamics implemented by the source.
- <method> = PIMD-1D: one-dimensional PIMD driver.
- <method> = HMC-1D: one-dimensional hybrid Monte Carlo driver.
- <method> = BCMD-1D: one-dimensional BCMD driver.
- <method> = RPMD-1D: one-dimensional RPMD driver.
- <method> = TRPMD-1D: one-dimensional TRPMD driver.
- <method> = CMD-1D: one-dimensional CMD driver.
- <method> = MATSUBARA-1D: one-dimensional Matsubara-dynamics driver.

Default: "STATIC".

11.1.4 <natom>

Used in ALL methods. This keyword specifies the number of atoms. Input: one integer on the first line. Default: 0 (not defined).

NOTE: This keyword is only used for <input_style> = OLD. For <input_style> = NEW, this keyword is obsolete. See Section 3 for details.

11.1.5 <nspec>

Used in ALL methods. This keyword specifies the atomic species present in the system. On the first line, the number of species should be given. On the following lines, the parameter set for each atomic species, namely the atomic symbol (character), the atomic mass in amu (real number), and the number of atoms (integer) for that species, should be provided. Default: 0 (not defined). **NOTE:** The same atomic symbol may be used more than once.

NOTE: This keyword is only used for <input_style> = OLD. For <input_style> = NEW (default), this keyword

is obsolete. See Section 3 for details.

11.2 Primary simulation-control keywords in input.dat

These controls are used across many simulation methods. Most have parser defaults, so membership in this group does not mean that every keyword must be written in every input.dat. Set them explicitly when the default is not appropriate for the chosen workflow.

11.2.1 <bath_type>

Used with <method> = MD, PIMD, MTS, CMD, MTD. This keyword sets the type of Nosé-Hoover thermostats. Input: one character string on the first line.

- <bath_type> = NONE: no thermostat is used.
- <bath_type> = NHC: a Nosé-Hoover chain thermostat attached to the whole system.
- <bath_type> = NHCS: Nosé-Hoover chain thermostats attached independently to x, y, and z.
- <bath_type> = MNHC: massive Nosé-Hoover chain thermostats attached to each degree of freedom and to each bead.

The conditions are as follows.

- For <method> = MD with <ensemble> = NVT, one must choose MNHC.
- For <method> = CMD, MTD, MTS, one must choose MNHC.

Default: "NONE" (no thermostat).

11.2.2 <dt>

Used in ALL cases except when <method> is STATIC, TESTFORCE, TESTVIRIAL, GEOOPT, NMA, PHONON, BOXOPT, or FULLOPT. This keyword sets the step size. Input: one real number on the first line.

- For <method> = SD, STRING, GAD: the initial step size in hartree$^{0.5}$ femtoseconds.
- For <method> = MD, PIMD, PIHMC, BCMD, CMD, RPMD, TRPMD, REHMC, TFS, MFE, MTS, MTD, or ROTOR: the step size in femtoseconds.

Default: "0.25" (either femtoseconds or hartree$^{0.5}$ femtoseconds, depending on the method chosen).

NOTE: In <method> = CMD, it is recommended that the <dt> value be set smaller than usual by a factor of <igamma>, to ensure adiabatic separation between the centroid and the non-centroid modes.

11.2.3 <iboundary>

Used in ALL cases. This keyword sets the boundary condition. Input: one integer on the first line. Choose one of the following options:

- <iboundary> = 0: free boundary (isolated system).
- <iboundary> = 1: periodic boundary; trajectory folded within the box; box matrix $\mathbf{h}$ in bohr.
- <iboundary> = 2: periodic boundary; trajectory not limited within the box; box matrix $\mathbf{h}$ in bohr.
- <iboundary> = BOHR: periodic boundary; trajectory folded within the box; box matrix $\mathbf{h}$ in bohr (same as 1).
- <iboundary> = ANGSTROM: periodic boundary; trajectory folded within the box; box matrix $\mathbf{h}$ in angstrom.

Default: "0" (free boundary).

For iboundary = 1, the box information should be specified. By default, the last step and the box matrix, $\overleftrightarrow{\mathbf{h}}$, are read from the file box.ini. The file box.ini should include three lines

$i,\ h(1,1),\ h(1,2),\ h(1,3)$

$i,\ h(2,1),\ h(2,2),\ h(2,3)$
$i,\ h(3,1),\ h(3,2),\ h(3,3)$

where the first column is the last step (integer), and the second, third, and fourth columns (real numbers) correspond to the box-matrix elements in bohrs, $(h(1,1), h(2,1), h(3,1)) = (a_x, a_y, a_z)$, $(h(1,2), h(2,2), h(3,2)) = (b_x, b_y, b_z)$, and $(h(1,3), h(2,3), h(3,3)) = (c_x, c_y, c_z)$.

If the file box.ini is not found, the box matrix is read from the three lines after the keyword <iboundary> in the file input.dat as:

<iboundary>
1
$h(1,1),\ h(1,2),\ h(1,3)$
$h(2,1),\ h(2,2),\ h(2,3)$
$h(3,1),\ h(3,2),\ h(3,3)$

11.2.4 <iprint_rest>

This keyword sets the print interval for generating and overwriting the restart files, *.ini. Input: one integer on the first line. Default: "10" (every 10 steps).

11.2.5 <iprint_std>

This keyword sets the print interval of the standard output standard.out. Input: one integer on the first line. Default: "10" (every 10 steps). Set to "-1" to switch off.

11.2.6 <iprint_trj>

This keyword sets the print interval of the standard trajectory file trj.out. Input: one integer on the first line. Default: "-1" (do not print).

11.2.7 <iread_exit>

This keyword sets the interval for checking the existence of the exit.dat file for a soft exit. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off.

NOTE: Reading the file exit.dat from the disk will slightly slow down the computation, which can be noticeable when running test calculations with a small molecular system using the CFF potential. In such a case, set <iread_exit> either to a larger value so that it is checked less frequently, or to "-1" so that it is not checked at all.

11.2.8 <nbead>

Used in ALL cases except <method> = STATIC, SD, GEOOPT. This keyword sets the number of beads.

- For <method> = PIMD, BCMD, CMD, RPMD, TRPMD, PIHMC, MTS: the number of beads.
- For <method> = TESTFORCE, TESTVIRIAL: the number of structures.
- For <method> = MTD: the number of walkers.
- For <method> = STRING: the number of images.
- For <method> = REHMC: the number of replicas.
- For <method> = NMA, PHONON: the number of shifted geometries calculated in parallel.
- For <method> = MD with <ncons> = 0, MFE, TFS: the number of independent trajectories.
- For <method> = MD with <ncons> = 1: the number of independent trajectories with different constraints.

Input: one integer on the first line. Default: "1".

11.2.9 <nstep>

Used in ALL cases except <method> = STATIC, TESTFORCE, TESTVIRIAL, NMA. This keyword sets the number of steps.

- For <method> = GEOOPT, BOXOPT, FULLOPT: the maximum number of optimization steps.
- For <method> = STRING: the maximum number of string updates.
- For <method> = SD: the number of steps along the steepest-descent trajectory.
- For other methods: the number of time steps of the trajectory.

Input: one integer on the first line. Default: "4".

11.2.10 <temperature>

Used in ALL cases except when <method> is STATIC, TESTFORCE, TESTVIRIAL, GEOOPT, NMA, PHONON, BOXOPT, FULLOPT, or ELASTIC. This keyword sets the temperature of the system in kelvin. Input: one real number on the first line.

- For <method> = MD, PIMD, BCMD, CMD, RPMD, TRPMD, MTD, MTS, PIHMC, REHMC, TFS, MFE, or ROTOR: the temperature of the Maxwell–Boltzmann distribution used to generate the initial random velocities.
- For <method> = MD, PIMD, BCMD, CMD, TRPMD, MTD, MTS, PIHMC, REHMC, or ROTOR, with <ensemble> = NVT, NPT, or NTT: the temperature of the Nosé–Hoover thermostats.

Default: "300" kelvin.

11.3 Additional common simulation-control keywords in input.dat

11.3.1 <beadspread>

Used in the cases of <method> = PIMD, PIHMC, BCMD, CMD, RPMD, TRPMD, MD, MTS, REHMC, MTD.

- For <method> = PIMD, PIHMC, MTS, BCMD, CMD, RPMD and TRPMD, this keyword controls the generation of the initial set of bead configurations. After the centroid configuration is read from the file centroid.dat, the initial position of each bead is generated randomly around the centroid, with the distribution being that of a free quantum atom. This keyword sets the temperature, in kelvin, that characterizes the bead spread of a free quantum atom. Input: one real number on the first line. As the temperature becomes higher, the bead spread shrinks, and all beads are generated closer to the centroid.
- The same idea is inherited for <method> = MD, REHMC, MTD. This keyword sets the temperature, in kelvin, that characterizes the spread used to generate the initial set of configurations for each trajectory in MD, each replica in REHMC, and each walker in MTD.

Default: "500" kelvin.

NOTE: The temperature in the keyword <beadspread> has nothing to do with the physical temperature of the system designated by the keyword <temperature>. If one prefers the initial configurations to be more different from one another, a smaller value should be chosen.

11.3.2 <corrections>

Used in the cases of <method> = PIMD, PIHMC, MTS, BCMD, CMD, RPMD and TRPMD. This keyword sets the translational and rotational corrections applied to path-integral simulations. Input: two integers on the first line.

- The first integer: the translational-correction option.
  - If "0", the correction is not applied.
  - If "1", the correction is applied at the initial step only.
  - If "2", the correction is applied at every step.

- The second integer: the rotational-correction option.
  - If “0”, the correction is not applied.
  - If “1”, the correction is applied at the initial step only.
  - If “2”, the correction is applied at every step.

The default values are “0, 0”, which mean that both translational and rotational corrections are not applied.

NOTE: In the PIMD code, there is a limitation in using the rotational correction for diatomic systems. In this specific case, the atoms must be aligned initially on the x axis.

11.3.3 <input_style>

Used in ALL cases. This keyword sets the input style for the initial molecular structure. Input: one character string on the first line.

- OLD: Old format using the file centroid.dat.
- NEW: New format using the file structure.dat.

Default: “NEW”.

11.3.4 <iprint_dcd>

This keyword sets the print interval of several binary files, *.dcd. Input: one integer on the first line. Default: “-1” (do not print).

The formats of the files box.dcd, dipoles.dcd, force.dcd, potential.dcd, trj.dcd, trj_unfolded.dcd, and vel.dcd are described in the output-files section.

The files box.dcd and trj_unfolded.dcd are only printed if periodic boundary conditions are used.

Note that in some cases the files box.dcd, trj.dcd, trj_unfolded.dcd, vel.dcd and force.dcd will be printed for each bead and will contain the bead number as part of their filename (e.g., trj.001.dcd).

In the case of dual potentials, the file force.dcd will be replaced by force_low.dcd and force_high.dcd, which contain the forces of the low- and high-accuracy potentials, respectively.

11.3.5 <iprint_xsf>

This keyword sets the print interval of the trajectory file in xsf format, *trj.xsf*. For a path-integral simulation, each bead is written as an independent XSF animation frame, so its energy, cell, coordinates, and forces remain grouped together. Input: one integer on the first line. Default: “-1” (do not print).

11.3.6 <iprint_xyz>

This keyword sets the print interval of the trajectory file in xyz format, trj.xyz. Input: one integer on the first line. Default: “-1” (do not print).

11.3.7 <iprint_extxyz>

This keyword sets the print interval of the trajectory file in extended xyz format, trjext.xyz. The extended xyz output contains coordinates, forces, energies, box information, and frame dipoles in units suitable for machine-learning potential workflows. See the description of trjext.xyz in Section 12.2. Input: one integer on the first line. Default: “-1” (do not print).

11.3.8 <ncolor>

Used with <method> = MD, PIMD, CMD, MTD. This keyword is only necessary when one wants to control the temperature very strongly using massive Nosé-Hoover chain thermostats. Nosé-Hoover chain thermostats are multiply attached with different masses; see Section 13.10.3 for details. This keyword sets the multiplicity M. Input: one integer on the first line. Default: “1” (single).

11.3.9 <nnhc>

Used with <method> = MD, PIMD, CMD, MTD. This keyword sets the chain length of the thermostat. Input: one integer on the first line. Default: "4". This keyword is neglected if a thermostat is not used.

11.3.10 <np_beads>

Number of bead groups computed at the same time by the XMPI decomposition. Each bead group contains <np_force> MPI ranks and owns a contiguous subset of the ring-polymer beads. This keyword is used when <xmpi>="ON"; ordinary MPI retains its historical bead distribution. The default is -1 (automatically defined). For an explicit XMPI layout, <np_beads>×<np_force> must equal the number of MPI ranks used to launch pimd.mpi.x. In TOML this is [parallel].beads.

11.3.11 <np_force>

Number of MPI ranks assigned to each XMPI bead group for force evaluation. The default is -1 (automatically defined). For an explicit XMPI layout, <np_beads>×<np_force> must equal the number of MPI ranks used to launch pimd.mpi.x. Increasing <np_force> is useful only for force backends that actually distribute work within a bead group; external backends may instead execute once on the subgroup leader and broadcast the result. In TOML this is [parallel].force.

11.3.12 <nref>

Used in the cases of <method> = MD, PIMD, CMD, MTS, MFE, TFS, AFED. This sets the number of updates for fast-varying forces per unit step <dt>.

- For <method> = MD with <ensemble> = NVE or NVT: the number of updates for the harmonic forces of constraints per step <dt>.
- For <method> = PIMD and CMD: the number of updates for the harmonic forces in path integrals, i.e., the forces between the beads, per step <dt>.
- For <method> = MTS: the number of updates for the harmonic forces in path integrals, i.e., between the beads, per step in the multiple time-scale method, <dt>/<nmulti>.
- For <method> = MFE and TFS: the number of updates of the electronic-state coefficients per step <dt>.
- For <method> = AFED with <afed_type> = TAMD or AFED: the number of updates for the harmonic forces of constraints per step <dt>.

In principle, the accuracy can be improved, but more computation is required as the <nref> value becomes larger. Input: one integer on the first line. Default: "1".

11.3.13 <nsymbol>

Used in ALL methods when <input_style> = NEW. Usually, this keyword is not used in the file input.dat by keeping the file input_default.dat unchanged. This keyword sets the atomic species followed by the atomic number, atomic mass in amu, and atomic radius in au. The atomic radius is not used in the pimd.x and pimd.mpi.x codes; it is used specifically in the prep_liquid.x code to set the cutoff distance when identifying the bond topology of a classical force field.

This keyword was slightly modified after version 2.1.

The canonical 2.8.0 fallback table begins with "249", i.e. 249 built-in species records, followed by the atomic-number/mass/radius/species table stored in input_default.dat. When <nsymbol> is omitted from input.dat, that complete canonical table is used.

11.3.14 <nys>

Used with <method> = MD, PIMD, CMD, MTD. This keyword sets the number of updates of the NHC, NHCS, and MNHC thermostats per unit step <dt>. The accuracy is usually improved by increasing the number of updates for the thermostats. Input: one integer on the first line.

- <nys> = 3 or 5: the Suzuki–Yoshida time-step decomposition method is employed.
- Other values of <nys>: the step size <dt> is divided equally into <dt>/<nys>.

Default: “5”.

11.3.15 <pimd_command>

Used with <ipotential> = ONIOM, QMMM, ALCHEM, DUAL. This is the execution command for the PIMD code (serial version). This information is only required when the PIMD code is called recursively via system calls. The default is “pimd.x”.

11.3.16 <pressure>

Used with <ensemble> = NPH, NPT. The external isotropic pressure is read from the first line (real value) in megapascals. Default: “0.101325” megapascals, which is equal to 1 atm.

11.3.17 <time_bath>

Used with <method> = MD, PIMD, CMD, MTD. This keyword controls the masses of thermostats. The thermostat masses are related to the characteristic time scale of the system. This keyword sets this time scale in femtoseconds. Input: one real number on the first line. Default: “10” femtoseconds.

11.4 Method-, potential-, and interface-specific keywords in input.dat

The keywords in this group are activated by a particular method, potential, external electronic-structure code, machine-learning backend, or analysis feature. Consult the corresponding method or potential section first; it usually provides the smallest working set of controls for that workflow.

11.4.1 <abinit_mp_exe_command>

Used with <ipotential> = ABINIT-MP5. This keyword sets the execution command of the ABINIT-MP5 code (the system call version). The first line gives the ABINIT-MP5 execution command. The default is “abinitmp”.

11.4.2 <abinit_mp_input>

Used with <ipotential> = ABINIT-MP. This keyword sets the input file of the internal/library ABINIT-MP interface. The legacy system-call <ipotential> = ABINIT-MP5 route instead reads the fixed prototype abinit-mp.dat and generates input.ajf. The name of the input file should be provided in the first line. The default is “input.ajf”.

11.4.3 <abinit_mp_output>

Used with <ipotential> = ABINIT-MP. This keyword sets the log file of the internal/library ABINIT-MP interface. Three data items should be provided in the first line.

- The name of the log file (character).
- The option, “0” or “1” (integer).
  - If “0”, only the information for the first bead is printed.
  - If “1”, the information for all beads is printed.
- The print interval (integer) of the ABINIT-MP log file.

The default is “output.abi 0 1”, which means that the information for the first bead at every step is printed to the ABINIT-MP log file, *output.abi*.

11.4.4 <abinit_mp_reuse_mo>

Used with <ipotential> = ABINIT-MP. This keyword controls the initial guess of monomer, dimer, and trimer molecular orbitals calculated in the ABINIT-MP code. Three integers should be provided in the first line.

- The option, "0" or "1" (integer).
  - If "0", the guess of monomer molecular orbitals is built from scratch at each step.
  - If "1", the monomer molecular orbitals of the last step are reused as the guesses for the next step.
- The option, "0" or "1" (integer).
  - If "0", the guess of dimer molecular orbitals is built from scratch at each step.
  - If "1", the dimer molecular orbitals of the last step are reused as the guesses for the next step.
- The option, "0" or "1" (integer).
  - If "0", the guess of trimer molecular orbitals is built from scratch at each step.
  - If "1", the trimer molecular orbitals of the last step are reused as the guesses for the next step.

The default values are "1, 1, 0", which means that the monomer and dimer molecular orbitals are reused while the trimer molecular orbitals are built from scratch.

NOTE: When the orbital information is reused (with "1"), the memory space should be sufficient to preserve the molecular orbitals. The trimer molecular orbitals require the largest memory size.

11.4.5 AccelNet model and evaluation keywords

Used with <ipotential> = ACCELNET. The required <accelnet_model_format>, <accelnet_energy_unit>, and <accelnet_length_unit> select the model format and its physical units. For AENET models, <ntype_accelnet> supplies the element/network list; for N2P2 models, <accelnet_model_dir> supplies the model directory. Optional <accelnet_chebyshev_version>, <accelnet_chebyshev_evaluation>, and <accelnet_g5_evaluation> select the descriptor convention and evaluation algorithms. <neighbor_list_skin> controls the neighbor cache, with a default of 1.0 Bohr. See Section 6.4.2 for the accepted values, defaults, and input examples.

11.4.6 <mbx_json_file>

Used with <ipotential> = MBXLIB. Names the MBX JSON configuration file, relative to the run directory unless an absolute path is supplied. Default: mbx.json. See Section 6.3.15.

11.4.7 <mbx_input_file>

Used with <ipotential> = MBXLIB. Names the monomer list containing the monomer count, names, and atom counts in PIMD structure order. Default: mbx.dat. See Section 6.3.15.

11.4.8 <iprint_mbx_mol_dipoles>

Used with <ipotential> = MBXLIB. A positive integer writes per-monomer permanent, induced, and total dipoles every specified number of steps to mbx_molecular_dipoles.out; zero disables this output. Default: 0. See Section 6.3.15.

11.4.9 <alchem_dat_dir>

Used with <method> = PIHMC with <irem_type> = HX. This keyword sets the input directories of alchemical potential which include input files for the two potentials specified by the keyword <alchem_potential>. Two character strings should be provided in the first line. The default values are "dat_1" and "dat_2".

11.4.10 <alchem_potential>

Used with <method> = PIHMC with <irem_type> = HX. This keyword sets the two potentials used for alchemical transformations. Two character strings should be provided in the first line. The default values are “CFF” and “CFF”.

11.4.11 <alchem_scr_dir>

Used with <method> = PIHMC with <irem_type> = HX. This keyword sets the temporary scratch directories of alchemical potential. Two character strings should be provided in the first line. The default values are “scr_1” and “scr_2”.

11.4.12 <auto_string>

Used with <method> = STRING. This keyword sets the option to control the step size in string method. When <auto_string> = OFF, the step size is a constant value set by the keyword <dt>. When <auto_string> = ON, the step size, initially set by the keyword <dt>, is controlled during the run. In this case the step size decreases or increases automatically at a given step interval by monitoring the change in the highest energy value among the images. Default: “ON”.

11.4.13 <box_anis>

This option is used for the cases of <npt_type>, <nph_type>, <ntt_type>, <nth_type> = PPHEX, when <method> = PIMD with <ensemble> = NPT, NTT, NPH, NTH; when <method> = PIHMC with <ensemble> = NPT, NTT; and when <method> = REHMC with <ensemble> = NPT. It controls whether anisotropic volume changes of the simulation box are allowed. Default: <box_anis> = ON (fully flexible). When <box_anis> = OFF, all off-diagonal elements of the simulation box matrix are frozen.

11.4.14 <cells_phonon>

Used with <method> = PHONON. This keyword sets the unit cell within the box. In the first line, three integers, n_i, n_j and n_k are read, which correspond to the number of unit cells replicated in the $\vec{a}$, $\vec{b}$ and $\vec{c}$ directions, respectively. In other words, the unit cell is composed of the Bravais lattice vectors $\vec{i} = \vec{a}/n_i$, $\vec{j} = \vec{b}/n_j$ and $\vec{k} = \vec{c}/n_k$. Default: “1, 1, 1” (the box is the unit cell).

11.4.15 <cluster>

Used with <mech_type> = CLUSTER This keyword sets the parameters of the external force applied to prevent molecular evaporation under free boundary conditions. In the first line, two real numbers should be provided, corresponding to the radius r_c (bohr) and the force constant k_c (hartree/bohr^2) of the trapping barrier. The default values are not available.

11.4.16 <cp2k_lib_calcforce>

Used with <ipotential> = CP2KLIB. This keyword sets how often the CP2K forces are evaluated. In most cases, this should keep the default value of 1, as the forces are evaluated at every step. However for Hybrid Monte Carlo (HMC) methods the force evaluation is not needed, and can be skipped for steps not providing training data where forces are necessary. If this is set to -1, then the force is never evaluated.

11.4.17 <cp2k_lib_chargeout>

Used with <ipotential> = CP2KLIB. This keyword collects the Mulliken and Hirshfeld charges from CP2K. To output the charges set this keyword to 1 and set the iprint_dcd keyword as well. This will cause the file charges.dcd to be output, which contains both sets of charges.

The canonical 2.8.0 fallback from input_default.dat is

```
<cp2k_lib_chargeout>
-1
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.18 <cp2k_lib_output>

Used with <ipotential> = CP2KLIB. This keyword sets the amount of calculations which will provide an output.cp2k file. If the serial code (pimd.x) is executed, setting this to 0 or 1 will result in a single file being created in the 001 directory of the calculation, this file keeps track of all the CP2K output. In the case of the parallel code (pimd.mpi.x), setting this to 0 will result in only the results of the first bead being printed to the directory 001. On the other hand, setting this to 1 will result in the DFTB+ output for each bead to be stored in a directory with a number corresponding to the bead index. When a calculator handles several beads, they share the directory numbered after its first bead. Multiple MPI workers on the same bead share a single log. In both cases, setting this to any other integer results in all contents of output.dftb from CP2K being written to /dev/null, i.e., will never be written. The default is “0”.

11.4.19 <cut_rec_3d>

Used with <method> = MTD. This keyword sets the cutoff parameter, x (real number), for the hill reconstruction in three-dimensional metadynamics. The data should be provided in the first line.

The Gaussian hills are added only to the mesh points where the contributions are larger than $\exp(-x)$. A value larger than “10.0” is recommended to suppress errors. For a value more than “37.0”, the cutoff is simply neglected. Default: “100.0” (no cutoff).

11.4.20 <delay_aenet>

Used with Self-Learning Hybrid Monte Carlo and related methods using AENET. This sets a delay in seconds, that should be input as a real number between double quotation marks. This delay is applied before all disk read and writes related to the learning process. The reason for this delay is to ensure that distributed file systems are properly synced before reading files for creating training sets, etc. If you experience problems with reading files from disk that do not seem to be missing on further inspection, it might help to extend this delay beyond the default value of 0.25.

11.4.21 <dftb_exe_command>

Used with <ipotential> = DFTB. This keyword sets the execution of DFTB. The DFTB execution command should be provided in the first line. The default is “dftb+”.

11.4.22 <dftb_lib_output>

Used with <ipotential> = DFTBLIB. This keyword sets the amount of calculations which will provide an output.dftb file. If the serial code (pimd.x) is executed, setting this to 0 or 1 will result in a single file being created in the 001 directory of the calculation, this file keeps track of all the DFTB+ output. In the case of the parallel code (pimd.mpi.x), setting this to 0 will result in only the results of the first bead being printed to the directory 001. On the other hand, setting this to 1 will result in the DFTB+ output for each bead to be stored in a directory with a number corresponding to the bead index. In both cases, setting this to any other integer results in all contents of output.dftb from DFTB+ being written to /dev/null, i.e., will never be written. The default is “0”.

11.4.23 <dftb_version>

Used with <ipotential> = DFTB. This keyword sets the version of DFTB. The DFTB version should be provided in the first line, either “18.2” or “19.1”. The default is “19.1”.

11.4.24 <dir_save_aenet>

Used with <ipotential> = AENET. This keyword sets the save directory for the trained networks which are saved periodically through the setting the <istep_save_aenet> keyword. Default: “saved_networks”

11.4.25 <dir_save_n2p2>

Used with <ipotential> = N2P2. This keyword sets the save directory for the trained networks which are saved periodically through the setting the <istep_save_n2p2> keyword. Default: “saved_networks”

11.4.26 <dir_save_mace>

Used with <ipotential> = MACE. This keyword sets the save directory for trained snapshot models written periodically through the <istep_save_mace> keyword. Default: “saved_models”.

11.4.27 <dir_train_aenet>

Used with <ipotential> = AENET. This keyword sets the scratch directory of trained networks in AENET potential. Input: one character string on the first line. Default: “trained_networks”.

11.4.28 <dir_train_mace>

Used with <ipotential> = MACE. This keyword sets the working directory used by the MACE interface for generated runtime and training files. Typical files created below this directory are *structures*, mace_training.xyz, mace.ini, mace_count_1.out, mace_count_2.out, *checkpoints*, and *mace_results*. The directory is created automatically by PIMD when needed. Input: one character string on the first line. Default: “mace_work”.

11.4.29 <dosrange_phonon>

Used with <method> = PHONON. This keyword sets the frequency range of the phonon density of states to be printed in the output file phonon_dos.out. Three real values should be provided in the first line designating the lower bound, the upper bound, and the increment frequencies in cm^{-1}. The default values are “0.0” cm^{-1}, “5000.0” cm^{-1} and “5.0” cm^{-1}, respectively.

11.4.30 <dt_conv_gad>

Used in the case of <method> = GAD. This keyword sets the GAD converged step size in $hartree^{0.5}$ femtoseconds. The data should be provided in the first line (real number) in mass femtoseconds.

The canonical 2.8.0 fallback from input_default.dat is

```
<dt_conv_gad>
1.d-5
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.31 <dt_om>

Used in the case of <method> = OMOPT, TESTOM. This keyword sets the time increment of Onsager-Machlup action. The data should be provided in the first line (real number) in femtoseconds.

The canonical 2.8.0 fallback from input_default.dat is

```
<dt_om>
0.625d0
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.32 <dt_poly>

Used in the POLYMER code. This keyword sets the mass-weighted evolution step for the centroid images. Its dimension is $\sqrt{\text{mass}}\,\text{length}$, equivalently $\sqrt{\text{energy}}\,\text{time}$; it is not an ordinary MD time step. Input: one real number on the first line. Default: “0.04d0”.

11.4.33 <dtemp_meta>

Used with <method> = MTD. This keyword sets the temperature parameter of well tempered metadynamics. A real number should be provided in the first line in kelvin. If the value is equal to or less than zero, well tempered metadynamics is switched off, and conventional metadynamics is chosen. Default: “0.0” (conventional metadynamics).

11.4.34 <dual_dat_dir>

Used with <method> = PIHMC with <ipotential> = DUAL. This keyword sets the potential of the DUAL method. Two character strings, corresponding to the high-level potential and the low-level potential, should be provided. In the second line, two character strings, corresponding to the input data directory of high-level potential and that of the low-level potential, should be provided. The default values are “dat_hi” and “dat_lo”.

11.4.35 <dual_potential>

Used with <method> = PIHMC with <ipotential> = DUAL. This keyword sets the potential of the DUAL method. Two character strings, corresponding to the high-level and low-level potentials, should be provided. The default values are “CFF” and “CFF”.

11.4.36 <dual_scr_dir>

Used with <method> = PIHMC with <ipotential> = DUAL. This keyword sets the potential of the DUAL method. Two character strings, corresponding to the high-level potential and the low-level potential, should be provided. In the second line, two character strings, corresponding to the temporary execution directories of high-level potential and that of the low-level potential, should be provided. The default values are “scr_hi” and “scr_lo”.

11.4.37 <efei>

Used with <mech_type> = EFEI. This keyword sets the parameters of the EFEI method. In the first line, two integers and one real number should be provided, corresponding to the two atoms in which the external force is applied, and the strength of the force between those atoms in hartree/bohr. The positive (negative) value of the force means repulsive (attractive). The default values are not available.

11.4.38 <ends_poly>

Used in the POLYMER code. This keyword sets the option for end centroid images, either “FIXED” or “FREE”. Input: one character string on the first line. Default: “FIXED”.

11.4.39 <ends_string>

Used with <method> = STRING. This keyword sets the boundary condition of the string. Input: one character string on the first line.

- For <method> = STRING:
  - <ends_string> = FIXED: Both ends of the string are fixed.
  - <ends_string> = FREE: Both ends of the string are optimized.
  - <ends_string> = FREEFIXED: The reactant end is optimized while the product end is fixed.
  - <ends_string> = FIXEDFREE: The reactant end is fixed while the product end is optimized.

The default is “FIXED”.

11.4.40 <eps_best>

Used in the case of <ipotential> = QMMM and <ioption_best> = 1. This keyword sets the cut off energy of the BEST method in hartree. Input: one real number on the first line. The default is “1.0e-16” hartree.

11.4.41 <equation_om>

Used in the case of <method> = OMOPT, TESTOM. This keyword sets the equation of motion of Onsager-Machlup action. Input: one character string on the first line.

- OVERDAMPED+: Overdamped Langevin equation.
- UNDERDAMPED+: Underdamped Langevin equation.

The default is “OVERDAMPED”.

11.4.42 <fc_best>

Used in the case of <ipotential> = QMMM and <ioption_best> = 1. This keyword sets the force constant of the BEST method in hartree/bohr^2. Input: one real number on the first line. The default is “25.0” hartree/bohr^2.

11.4.43 <fdiff>

Used in the case of <method> = NMA, PHONON, TESTFORCE, TESTVIRIAL, OMOPT, TESTOM. This keyword sets the finite shifts (in bohr) to compute those numerical derivatives. Input: one real number on the first line.

- For <method> = NMA, PHONON: the Hessian matrix is evaluated as the numerical derivative of forces with the finite geometric shifts, <fdiff>.
- For <method> = TESTFORCE: the force is evaluated as the numerical derivative of potential with the finite geometric shifts, <fdiff>.
- For <method> = TESTVIRIAL: the virial is evaluated as the numerical derivative of potential with the finite box shifts, <fdiff>.
- For <method> = OMOPT and TESTOM: the gradient of OM action is evaluated as the numerical derivative of forces with the maximum geometric shifts, <fdiff>.

Default: “0.0001” bohr.

11.4.44 <g03_command>

Used with <ipotential> = G03. This keyword sets the execution of G03. The first line gives the G03 execution command, and the second line gives the *formchk* execution command. The default is “g03” and “formchk”.

11.4.45 <g03_dip>

Used with <ipotential> = G03. This keyword sets the option for G03 calculation of dipole moment. Input: one integer on the first line. The dipole calculation is switched on if “1”, switched off if “0”. Default: “1” (on).

11.4.46 <g03_grad>

Used with <ipotential> = G03. This keyword sets the option for G03 calculation of potential gradient. Input: one integer on the first line. The gradient calculation is switched on if “1”, switched off if “0”. Default: “1” (on).

11.4.47 <g09_command>

Used with <ipotential> = G09. This keyword sets the execution of G09. The first line gives the G09 execution command, and the second line gives the *formchk* execution command. The default is “g09” and “formchk”.

11.4.48 <g09_dip>

Used with <ipotential> = G09. This keyword sets the option for G09 calculation of dipole moment. Input: one integer on the first line. The dipole calculation is switched on if “1”, switched off if “0”. Default: “1” (on).

11.4.49 <g09_grad>

Used with <ipotential> = G09. This keyword sets the option for G09 calculation of potential gradient. Input: one integer on the first line. The gradient calculation is switched on if “1”, switched off if “0”. Default: “1” (on).

11.4.50 <g09_oniom>

Used with <ipotential> = G09. This keyword sets the option for the use of ONIOM implemented in G09. Input: one integer on the first line. The ONIOM is switched on if “1”, switched off if “0”. Default: “0” (off).

11.4.51 <g16_command>

Used with <ipotential> = G16. This keyword sets the execution of G16. The first line gives the G16 execution command, and the second line gives the *formchk* execution command. The default is “g16” and “formchk”.

11.4.52 <g16_dip>

Used with <ipotential> = G16. This keyword sets the option for G16 calculation of dipole moment. Input: one integer on the first line. The dipole calculation is switched on if “1”, switched off if “0”. Default: “1” (on).

11.4.53 <g16_grad>

Used with <ipotential> = G16. This keyword sets the option for G16 calculation of potential gradient. Input: one integer on the first line. The gradient calculation is switched on if “1”, switched off if “0”. Default: “1” (on).

11.4.54 <g16_oniom>

Used with <ipotential> = G16. This keyword sets the option for the use of ONIOM implemented in G16. Input: one integer on the first line. The ONIOM is switched on if “1”, switched off if “0”. Default: “0” (off).

11.4.55 <g98_command>

Used with <ipotential> = G98. This keyword sets the execution of G98. The first line gives the G98 execution command, and the second line gives the *formchk* execution command. The default is “g98” and “formchk”.

11.4.56 <g98_dip>

Used with <ipotential> = G98. This keyword sets the option for G98 calculation of dipole moment. Input: one integer on the first line. The dipole calculation is switched on if “1”, switched off if “0”. Default: “1” (on).

11.4.57 <g98_grad>

Used with <ipotential> = G98. This keyword sets the option for G98 calculation of potential gradient. Input: one integer on the first line. The gradient calculation is switched on if “1”, switched off if “0”. Default: “1” (on).

11.4.58 <gamess_command>

Used with <ipotential> = GAMESS. This keyword sets the execution of GAMESS. The first line gives the GAMESS execution command. The default is "rungms gamess 00 1".

11.4.59 <gamma_gad>

Used with <method> = GAD. This keyword sets the gamma value of GAD, in atomic unit. The data should be provided in the first line. The default is "0.1".

11.4.60 <gamma_om>

Used in the case of <method> = OMOPT, TESTOM. This keyword sets the friction constant of Onsager-Machlup action. The data should be provided in the first line (real number) in femtosecond^{-1}. The default is "0.00625" femtosecond^{-1}.

11.4.61 <generate_aenet>

Used with <ipotential> = AENET. This keyword sets the generate inputs in AENET potential. Following this keyword, all lines of the AENET training input should be copied, except the AENET keyword "FILES", the number of xsf files, and xsf file names. The default values are not available.

11.4.62 <gh_meta>

Used with <method> = MTD. This keyword sets the height of Gaussian hills of metadynamics simulation, in kelvin (energy divided by Boltzmann constant). Input: one real number on the first line. In principle, this value should not be higher than the value specified by the keyword <temperature>. Default: "150" kelvin.

11.4.63 <guess_poly>

Used in the POLYMER code. This keyword sets the initial guess of centroid images from the file structure.dat, either "LINE" (line interpolation of two ends) or "SPLINE" (spline interpolation of all images). Input: one character string on the first line. Default: "LINE".

11.4.64 <gw_meta>

Used with <method> = MTD. This keyword sets the Gaussian width for each type of collective variable (CV). Each line contains the CV type followed by one real width. The canonical values in input_default.dat are

```
DIST    0.100d0
ANGL    5.000d0
DIH    10.000d0
DIFF    0.100d0
CN      0.100d0
DCN     0.100d0
XYZ     0.100d0
DXYZ    0.100d0
```

The values are indexed by CV type rather than by CV number and are used as fallbacks when <gw_meta> is absent from input.dat. For DIST, DIFF, XYZ, and DXYZ the width is in bohr; for ANGL and DIH it is in degrees; CN and DCN are dimensionless. XYZ and DXYZ use the unweighted geometric center of the selected atomic kind/group, not a mass-weighted center of mass.

11.4.65 <iatom_qtst>

This option is used when <method> = PIMD and <ensemble> = QTST. The data should be provided on the first line as an integer. Default: "0". When the value is nonzero (i), the centroid of atom i is fixed at its initial position

along the Z direction.

11.4.66 <iformat_trj>

This keyword sets the print option of trj.out. It could be chosen from below.

- iformat_trj = 1: trajectory is printed with 16 digits.
- iformat_trj = 2: trajectory is printed with 8 digits.

Default: "1".

11.4.67 <iformat_xyz>

This keyword sets the print option of trj.xyz. It could be chosen from below:

- iformat_xyz = 1: prints each bead configuration separately.
- iformat_xyz = 2: prints all beads into one configuration.
- iformat_xyz = 3: prints only the centroid configuration.

Input: one integer on the first line. Default: "2".

11.4.68 <igamma>

Used with <method> = CMD. This keyword controls the adiabaticity in centroid molecular dynamics. Input: one integer on the first line. The accuracy of CMD will increase as the value is larger. Note that the adiabaticity parameter, γ, is given by $\gamma = 1/\text{<igamma>}^2$. Default: "10".

11.4.69 <ikind_best>

Used in the case of <ipotential> = QMMM and <ioption_best> = 1. This keyword designates the atomic kind in the QM/MM BEST method. Default: "1".

11.4.70 <ikind_xyz>

Used with <iprint_xyz> > 0. This keyword controls the print range of atomic kinds in the output trajectory file, trj.xyz. The minimum and maximum values of atomic kinds should be provided in the first line (two integers). For the maximum value of atomic kind, the value "-1" could be used to specify the number of atomic kinds present in the system. The default values are "1, -1" (all atomic kinds).

11.4.71 <integrator_bcmd>

Time integrator for Brownian chain molecular dynamics. The options are "EULER", "GILLESPIE", "PLATEN", and "RESPA". The default is "RESPA".

11.4.72 <integrator_trpmd>

Time integrator for thermostatted ring polymer molecular dynamics. The options are "LEIMKUHLER" and "VANDEN-EIJNDEN". The default is "VANDEN-EIJNDEN".

11.4.73 <iobest>

Used in the case of <ipotential> = QMMM and <ioption_best> = 1. This keyword designates the atomic kind in the QM/MM BEST method. Input: one integer on the first line. Default: "1".

11.4.74 <ioption_best>

Used with <method> = QMMM. This keyword designates the usage of the QM/MM BEST method. The data should be provided in the first line (integer), either 0 (off: BEST method not used) or 1 (on: BEST method used). Default: "0" (off).

11.4.75 <ioption_meta>

Used with <method> = MTD. This historical keyword selects the multiple-walker hill-filtering option.

- ioption_meta = 0: all walkers are selected.
- ioption_meta = 1: a well-separated subset of walkers is selected.

In the current 2.8.0 source, the selection flag is computed in addhills_meta, but the final test that would suppress hills for unselected walkers is commented out. Therefore this switch does not currently change which hills are added. It is retained here for legacy-input compatibility. Input: one integer on the first line. Default: "1".

11.4.76 <ioption_xsf_aenet>

Used with <ipotential> = AENET with <method> = PIHMC or <method> = REHMC. This keyword sets the option of printing xsf files for AENET potential. It could be chosen from below:

- ioption_xsf_aenet = 0: the xsf files are printed only for the accepted structures; the last accepted structure is copied when a trial structure is rejected.
- ioption_xsf_aenet = 1: the xsf files are printed for all the trial structures, both for the accepted and rejected structures.

Input: one integer on the first line. Default: "1".

11.4.77 <ioption_xsf_n2p2>

Used with <ipotential> = N2P2 with <method> = PIHMC or <method> = REHMC. This keyword sets the option of printing xsf files for N2P2 potential. It could be chosen from below:

- ioption_xsf_n2p2 = 0: the xsf files are printed only for the accepted structures; the last accepted structure is copied when a trial structure is rejected.
- ioption_xsf_n2p2 = 1: the xsf files are printed for all the trial structures, both for the accepted and rejected structures.

Input: one integer on the first line. Default: "1".

11.4.78 <ioption_xsf_mace>

Used with <ipotential> = MACE with <method> = PIHMC or <method> = REHMC. This keyword sets the option of printing xsf files for the MACE potential. It can be selected from the following options:

- ioption_xsf_mace = 0: the xsf files are printed only for accepted structures; the last accepted structure is copied when a trial structure is rejected.
- ioption_xsf_mace = 1: the xsf files are printed for all trial structures, both accepted and rejected.

Input: one integer on the first line. Default: "1".

11.4.79 <iorder_hmc>

Used with <method> = PIHMC. This keyword sets the order of Suzuki-Trotter expansion, "2" or "4".

- iorder_hmc = 2: the second order Suzuki-Trotter expansion.

- iorder_hmc = 4: the fourth order Suzuki-Trotter expansion.

The data should be provided in the first line. Default: "2" (the second order).

11.4.80 <iprint_akin>

This keyword sets the print interval of kinetic energy to the output file akin.out. Input: one integer on the first line. Default: "-1" (do not print).

11.4.81 <iprint_alc>

Used with <method> = PIHMC with <irem_type> = HX. This keyword sets the print interval of the alchemical analysis to the output file alc.out. Input: one integer on the first line. Default: "-1" (do not print). Set to a positive integer to enable printing at that interval.

11.4.82 <iprint_best>

Used in the case of <ipotential> = QMMM and <ioption_best> = 1. This keyword sets the print interval to the output file dipole.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.83 <iprint_bond>

Used in ALL cases except <method> = STATIC, NMA, PHONON, MTD, STRING, MFE, TFS, REHMC. This keyword sets the print interval of the bond lengths (interatomic distances) to the output file dipole.out. Input: one integer on the first line. Default: "-1" (do not print).

NOTE: This analysis is not recommended when the number of atomic species is large, since the amount of computation grows quadratically.

11.4.84 <iprint_box>

Used with <method> = PIMD, PIHMC with <ensemble> = NPT, NTT, NPH, NTH. This keyword sets the print interval of the following data to the output file box.out.

- For <ensemble> = NPT, NPH, the print interval of the box matrix and the pressure matrix.
- For <ensemble> = NPT, NPH, the print interval of the box matrix, the pressure matrix, the stress matrix and the strain matrix.

Input: one integer on the first line. Default: "-1" (do not print).

11.4.85 <iprint_cavg>

This keyword sets the print interval of heat capacity to the output file cavg.out. Input: one integer on the first line. Default: "-1" (do not print).

11.4.86 <iprint_cons>

Used with <method> = MD. This keyword sets the print interval of the results of constrained molecular dynamics to the output file cons.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.87 <iprint_cv_meta>

Used with <method> = MTD. This keyword sets the print interval of the trajectory of the collective variables to the output file cv.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.88 <iprint_cv_tass>

Used with <method> = TASS. The same as iprint_cv_meta used in <method> = MTD.

The canonical 2.8.0 fallback from input_default.dat is

```
<iprint_cv_tass>
1
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.89 <iprint_dip>

Used in ALL cases except <method> = STATIC, NMA, PHONON, MTD, STRING, MFE, TFS, REHMC. This keyword sets the print interval of the dipole moments to the output file dipole.out. Input: one integer on the first line. Default: "-1" (do not print).

11.4.90 <iprint_dual>

Used with <method> = PIHMC with <ipotential> = DUAL. This keyword sets the print interval of high and low-level potentials to the output file dual.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.91 <iprint_eavg>

Used in ALL cases except <method> = STATIC, NMA, PHONON, MTD, STRING, MFE, TFS, REHMC. This keyword print interval of energy averages to the output file eavg.out. Input: one integer on the first line. Default: "-1" (do not print).

11.4.92 <iprint_hfx_aenet>

Used with <ipotential> = AENET. This keyword sets the print interval of heat flux to the output file hfx.out. Input: one integer on the first line. Default: "-1" (do not print).

11.4.93 <iprint_hfx_n2p2>

Used with <ipotential> = N2P2. This keyword sets the print interval of heat flux to the output file hfx.out. Input: one integer on the first line. Default: "-1" (no print).

11.4.94 <iprint_mech>

Used in ALL cases except <method> = STATIC, NMA, PHONON, MTD, STRING, MFE, TFS, REHMC. This keyword sets the print interval of energy components in the file mech.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.95 <iprint_meta>

Used with <method> = MTD. This keyword sets the print interval of energy components to the output file meta.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.96 <iprint_mfe>

Used with <method> = MFE. This keyword sets the print interval of the data of nonadiabatic mean field dynamics to the output file mfe.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.97 <iprint_minfo>

Used in <method> = NMA. This prints the normal mode coordinates in a format that is readable by the Sindo program. Default: "-1" (do not print). Set to "1" to print the results of normal mode analysis in the .minfo format.

11.4.98 <iprint_mom>

Used in ALL cases except <method> = STATIC, NMA, PHONON, MTD, STRING, MFE, TFS, REHMC. The keyword sets the print interval of linear momentum and angular momentum to the output file momentum.out. Input: one integer on the first line. The momentum conservation can be checked for MD and RPMD methods without thermostats. For the free boundary condition, both linear and angular momenta are conserved, while for the periodic boundary condition only the linear momentum is conserved. Default: "-1" (do not print).

11.4.99 <iprint_nac>

Used with <method> = MFE and TFS. This keyword sets the print interval of the data of nonadiabatic dynamics to the output file nac.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.100 <iprint_oniom>

Used with <ipotential> = ONIOM. The keyword sets the print interval of the potential energy compositions to the output file oniom.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.101 <iprint_poly>

Used in the POLYMER code. This keyword sets the print interval of centroid mean force during the PIMD runs. Input: one integer on the first line. Default: "10".

11.4.102 <iprint_qmmm>

Used with <ipotential> = QMMM. The keyword sets the print interval of the potential energy compositions to the output file qmmm.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.103 <iprint_rdf>

Used in ALL cases except <method> = STATIC, NMA, PHONON, MTD, STRING, MFE, TFS. This keyword sets the print interval of the distribution of atomic pairs to the output file rdf.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

NOTE: The meshes is given by the keyword <params_rdf>.

NOTE: This analysis is not recommended when the number of atomic species is large, since the amount of computation grows quadratically.

11.4.104 <iprint_rdfbead>

This keyword sets the print interval of the bead-wise distribution of atomic pairs to the output file rdfcent.out. Input: one integer on the first line. Default: "-1" (not printed).

NOTE: The meshes is given by the keyword <params_rdf>.

NOTE: This analysis is not recommended when the number of atomic species is large, since the amount of computation grows quadratically.

11.4.105 <iprint_rdfcent>

This keyword sets the print interval of the centroid-wise distribution of atomic pairs to the output file rdfbead.out. Input: one integer on the first line. Default: "-1" (not printed).

NOTE: The meshes is given by the keyword <params_rdf>.

NOTE: This analysis is not recommended when the number of atomic species is large, since the amount of computation grows quadratically.

11.4.106 <iprint_rec_meta>

Used with <method> = MTD. This keyword sets the print interval of free energy in metadynamics to the output file rec.out. Input: one integer on the first line. Default: "100" (every 100 steps). Set to "-1" to switch off.

NOTE: The accuracy of reconstructed hills in three dimensional metadynamics is controlled by the keyword <cut_rec_3d>.

11.4.107 <iprint_rgy>

Used in ALL cases except <method> = STATIC, NMA, PHONON, MTD, STRING, MFE, TFS, REHMC. This keyword sets the print interval of the radius of gyration to the output file rgy.out. Input: one integer on the first line. Default: "-1" (do not print).

11.4.108 <iprint_str>

Used with <method> = STRING, OMOPT. This keyword sets the print intervals of the output files string.xyz and string_pot.out. Input: one integer on the first line. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.109 <iprint_tfs>

Used with <method> = TFS. This keyword sets the print interval of the data of nonadiabatic surface hopping dynamics to the output file tfs.out. Default: "1" (every step). Set to "-1" to switch off printing.

11.4.110 <iprint_xsf_aenet>

Used with <ipotential> = AENET. This keyword sets the print interval of xsf files which are used to train and upgrade the AENET potential. Input: one integer on the first line. Default: "-1" (do not print).

11.4.111 <iprint_xsf_n2p2>

Used with <ipotential> = N2P2. This keyword sets the print interval of xsf files which are used to train and upgrade the N2P2 potential. Input: one integer on the first line. Default: "-1" (do not print).

11.4.112 <iprint_xsf_mace>

Used with <ipotential> = MACE. This keyword sets the print interval of xsf files used to train and upgrade the MACE potential. Input: one integer on the first line. Default: "-1" (do not print).

11.4.113 <irandom>

Used in the ALL the cases. Sometimes it is convenient to use time-dependent random numbers. This is useful for the cases in which the initial geometry, the initial velocity, etc., are randomly generated for each run. The data in the first line (integer) should be chosen from below:

- <irandom> = 0: random numbers are time-independent, giving the same series of random numbers.

- <irandom> = 1: random numbers are time-dependent by generating the seed from date command.

Default: “0”.

11.4.114 <irem_type>

Used with <method> = REHMC. This keyword specifies the type of replica exchange hybrid Monte Carlo method. It could be chosen from the following options.

- <irem_type> = T: parallel temperatures without the replica exchange.
- <irem_type> = TX: temperature replica exchange.
- <irem_type> = HX: Hamiltonian replica exchange.

The data (character) should be specified in the first line. Default: “TX”.

11.4.115 <istate_init>

Used with <method> = MFE and <method> = TFS. This keyword sets the initial electronic state for the nonadiabatic dynamics. Default: “1” (the ground state).

11.4.116 <istep_adjust_hmc>

Used with <method> = PIHMC and <method> = REHMC. This keyword sets the step interval of changing the MD steps per Metropolis step. When it is “-1” the MD steps per Metropolis step is set constant. Default: “-1”.

11.4.117 <istep_ax_hmc>

Used with <method> = REHMC. This keyword sets the interval of atom exchange. Input: one integer on the first line. Default: “-1” (no exchange).

11.4.118 <istep_hmc>

Used with <method> = PIHMC and <method> = REHMC. This keyword sets the number of MD steps per Metropolis step in hybrid Monte Carlo. When it is set to be varied by `istep_adjust_hmc > 0`, this keyword sets the smallest value. Default: “2” (every two steps).

11.4.119 <istep_max_hmc>

Used with <method> = PIHMC and <method> = REHMC with `istep_adjust_hmc > 0`. This keyword sets the largest value of the interval of changing the MD steps per Metropolis step. Default: “128”.

11.4.120 <istep_mul_hmc>

Used with <method> = PIHMC and <method> = REHMC with `istep_adjust_hmc > 0`. This keyword sets the common ratio of geometric progression, respectively, for the MD steps per Metropolis step. Default: “2”.

11.4.121 <iprint_rec_tass>

Used with <method> = TASS. The same as `iprint_rec_meta` used in <method> = MTD.

The canonical 2.8.0 fallback from input_default.dat is

```
<iprint_rec_tass>
100
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.122 <istep_save_aenet>

Used with <ipotential> = AENET. This keyword sets the save frequency for the trained networks. If set to a positive integer value the trained networks will be saved in the directory set by <dir_save_aenet> keyword. The trained networks will have the suffix "_iter_XXXXXX", where XXXXXX will be a number indicating which iteration the saved network stems from. Default: "-1" (do not save intermediary networks).

11.4.123 <istep_save_n2p2>

Used with <ipotential> = N2P2. This keyword sets the save frequency for the trained networks. If set to a positive integer value the trained networks will be saved in the directory set by <dir_save_n2p2> keyword. The trained networks will have the suffix "_iter_XXXXXX", where XXXXXX will be a number indicating which iteration the saved network stems from. Default: "-1" (do not save intermediary networks).

11.4.124 <istep_save_mace>

Used with <ipotential> = MACE. This keyword sets the save frequency for trained MACE models. If set to a positive integer value, snapshot models are saved in the directory specified by <dir_save_mace>. Default: "-1" (do not save intermediary models).

11.4.125 <iprint_tass>

Used with <method> = TASS. The same as iprint_meta used in <method> = MTD.

The canonical 2.8.0 fallback from input_default.dat is

```
<iprint_tass>
1
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.126 <istep_train_aenet>

Used with <ipotential> = AENET. This keyword sets the integer i to set the step interval to train and upgrade the AENET potential. Input: one integer on the first line. Default: "-1" (do not train).

11.4.127 <istep_train_n2p2>

Used with <ipotential> = N2P2. This keyword sets the integer i to set the step interval to train and upgrade the N2P2 potential. Input: one integer on the first line. Default: "-1" (do not train).

11.4.128 <istep_train_mace>

Used with <ipotential> = MACE. This keyword sets the integer i to specify the step interval for training and upgrading the MACE potential. Input: one integer on the first line. Default: "0" (do not train).

11.4.129 <ivar_qmmm>

Used with <ipotential> = QMMM in the periodic boundary condition. However, this is currently not supported.

The canonical 2.8.0 fallback from input_default.dat is

```
<ivar_qmmm>
0
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.130 <joption_meta>

Used with <method> = MTD. This keyword sets the option for metadynamics. It could be chosen from below:

- joption_meta = 0: a new hill is added only when both conditions are satisfied: the normalized CV displacement is at least 1.5 Gaussian widths and the elapsed time is at least <time_limit_meta>.
- joption_meta = 1: the displacement condition is skipped, so a new hill is added when the elapsed time reaches <time_limit_meta>.

Input: one integer on the first line. Default: “0”.

11.4.131 <kdisp_phonon>

Used in <method> = PHONON. The first line gives the number of wave vectors and each following line gives g_x, g_y, and g_z in bohr^{-1}. These are dimensional wave-vector components used directly in $\mathbf{g} \cdot \mathbf{d}$, not fractional reciprocal-lattice coordinates. The canonical fallback is one point at the Γ point:

```
1
0.0 0.0 0.0
```

Thus omitting the keyword does not disable the calculation; it selects a single Γ-point displacement.

11.4.132 <kdos_phonon>

Used in <method> = PHONON. This keyword sets the k-point sampling for the calculation of the vibrational density of states. Three integers should be provided. The default values are “1, 1, 1” (Γ point only).

11.4.133 <kickstep_bcmd>

Used in <method> = BCMD. This keyword sets the step interval of the random kick used in the improved integrator for BCMD simulations. One integer should be provided. Default: “1” (every step).

The kick is applied every kickstep_bcmd steps in a similar manner to Andersen thermostats. However, the magnitude of the kick is controlled so as to obey the form of the overdamped Langevin equation as long as the kickstep_bcmd value is small enough. As kickstep_bcmd becomes larger, the kick becomes softer, and energy conservation is generally improved. To find a reasonable range of values for kickstep_bcmd, simply use the code “test_bcmd.F” provided in the tools directory.

11.4.134 <lstep_train_aenet>

Used with <ipotential> = AENET. This keyword sets the integer i to set the last step to train and upgrade the AENET potential. Input: one integer on the first line. Default: “99999999”.

11.4.135 <lstep_train_n2p2>

Used with <ipotential> = N2P2. This keyword sets the integer i to set the last step to train and upgrade the N2P2 potential. Input: one integer on the first line. Default: “99999999”.

11.4.136 <lstep_train_mace>

Used with <ipotential> = MACE. This keyword sets the last step for training and upgrading the MACE potential. Input: one integer on the first line. Default: “99999999”.

11.4.137 <mace_use_cuda>

Used with <ipotential> = MACE only as a compatibility fallback when the libtorch runtime configuration cannot be read. A nonzero integer enables CUDA and zero selects CPU. It is read from input.dat and then input_

default.dat; the canonical fallback is 0.

11.4.138 <mace_device_id>

Used with <ipotential>=MACE only in the same compatibility fallback path. It selects the CUDA device identifier when CUDA is enabled. It is read from input.dat and then input_default.dat; the canonical fallback is 0.

11.4.139 <mace_rmax>

Used with <ipotential>=MACE only in the same compatibility fallback path. It supplies an optional cutoff override to the libtorch model. It is read from input.dat and then input_default.dat; the canonical fallback is -1.0, meaning that no positive cutoff override is supplied.

11.4.140 <mace_generate_command>

This tag is obsolete in the current PIMD 2.8.0 source. It is not read by the active MACE interface and has therefore been removed from the canonical input_default.dat. Do not use it in new inputs; use the current MACE inference/training YAML workflow described in Section 6.4.5.

11.4.141 <ase_infer_options_file>

Used with <ipotential>=ASE. This keyword sets the YAML file used for ASE calculator inference. The file is resolved relative to the current run directory unless an absolute path is given. Relative paths written inside that YAML file are resolved relative to the directory containing the YAML file itself. Default: "ase_infer_options.yaml".

11.4.142 <mace_infer_options_file>

Used with <ipotential>=MACE. This keyword sets the YAML file used for MACE inference. The file is resolved relative to the current run directory unless an absolute path is given. The model path specified by the model entry inside the YAML file is resolved relative to the directory containing that YAML file. Default: "mace_infer_options.yaml".

11.4.143 <mace_train_options_file>

Used with <ipotential>=MACE. This keyword sets the YAML file used for MACE training. The file is resolved relative to the current run directory unless an absolute path is given. The path given by foundation_model inside the YAML file is resolved relative to the directory containing that YAML file. Default: "mace_train_options.yaml".

11.4.144 <mech_type>

This keyword sets the use of external forces. The data in the first line (character) should be chosen from below:

- NONE: no external force.
- EFEI: external force explicitly included.
- CLUSTER: external force explicitly included.
- ZONE: confined to a zone with half harmonic potentials.

If "EFEI" is chosen, the EFEI term

$$V_{\text{mech}} = -Fr_{ij}, \tag{4}$$

is added to the potential. In this case, three parameters, atom i, atom j, and the force amplitude F in au (hartree/bohr), should be provided by the keyword <efei>.

If “CLUSTER” is chosen, external force is applied to prevent from molecular evaporation under free boundary condition. This option could be used for molecular clusters. A trapping barrier

$$V_{\text{mech}} = \frac{1}{2}\sum_{i=1}^{n} k_c \left\{\max\left(0, r_{i,g} - r_c\right)\right\}^2 \tag{5}$$

is added to the potential, where

$$r_{i,g} = \left|\mathbf{r}_i - \mathbf{r}_g\right| \tag{6}$$

is the distance of the i-th atom from the center of mass of the molecular system,

$$\frac{\sum_{j=1}^{n} m_i \mathbf{r}_i}{\sum_{j=1}^{n} m_i}. \tag{7}$$

In this case, two parameters, the radius r_c (bohr) and the force constant k_c (hartree/bohr^2) of the trapping barrier should be provided by the keyword **<cluster>**.

If “ZONE” is chosen, external force is applied so as to confine the set of collective variables, $\{q\}$, within a given zone. We use the sum of half-harmonic potentials with respect to the outward shift from the zone of each collective variable, Δq_α, namely,

$$V_{\text{mech}} = \frac{1}{2}\sum_{\alpha} k_\alpha \Delta q_\alpha^2 \tag{8}$$

where k_α is the force constant and

$$\Delta q_\alpha = \min\left(\max\left(0, q_\alpha - q_\alpha^{\max}\right), q_\alpha - q_\alpha^{\min})\right), \tag{9}$$

and $q_\alpha^{\min}$ and $q_\alpha^{\max}$ are the minimum and maximum values, respectively, of the zone without external forces. In this case, the parameters are provided by the keyword **<zone>**. In the first line, number of collective variables, n, is specified. The following n lines contain one zone record each. The current source implements the zone types DIST, CN, XYZ, CXYZ, and CM. The parsers also recognize ANGL, DIH, DIFF, DCN, and DXYZ, but these branches are explicitly marked unfinished and terminate with an input error; they should not be used. Two representative implemented forms are:

- ```
  DIST $1 $2 $3 $4 $5
       ... distance of atoms $1 and $2
       is confined in a zone with the lower bound $3
       and the upper bound $4 (in bohrs). A half-harmonic
       potential with force constant $5 is applied outside
       the zone.
  ```

```
CN $1 $2 $3 $4 $5 $6 $7 $8
     ... coordination number of species $1 and $2 given
     as a rational function (nu, mu, req = $3, $4, $5)
     is confined in a zone with the lower bound $6
     and the upper bound $7. A half-harmonic potential
     with force constant $8 is applied outside the zone.
```

Default: “NONE’.

11.4.145 <zone>

Used only when the external mechanical term is selected as **<mech>** = ZONE. The parser first searches input.dat and, if the tag is absent, tries input_default.dat. The canonical 2.8.0 input_default.dat intentionally contains no **<zone>** block because the atom/CV definitions are system specific. Thus the block is required in input.dat whenever **<mech>** = ZONE is selected; there is no generic default. The first value is the number of zone records, followed by the records described above. The source currently implements DIST, CN, XYZ, CXYZ, and CM; ANGL, DIH, DIFF, DCN, and DXYZ are recognized but explicitly unfinished and cause input termination.
```

11.4.146 <mg_meta>

Used with <method> = MTD. This keyword sets maximum number of Gaussian hills in metadynamics. Input: one integer on the first line. This is necessary in the code in order to control the memory size. Default: "50000".

11.4.147 <minxsf_train_aenet>

Used with <ipotential> = AENET. This keyword sets the minimum number of xsf structure files required to train the artificial neural network potential. Input: one integer on the first line. Default: 300.

11.4.148 <minxsf_train_n2p2>

Used with <ipotential> = N2P2. This keyword sets the minimum number of xsf structure files required to train the artificial neural network potential. Input: one integer on the first line. Default: 300.

11.4.149 <minxsf_train_mace>

Used with <ipotential> = MACE. This keyword sets the minimum number of xsf structure files required to start MACE training. Input: one integer on the first line. Default: 0.

11.4.150 <model_water>

Used with <ipotential> = WATER. This keyword selects one of the built-in flexible water potentials. Input: one integer on the first line.

- <model_water> = 0: q-TIP4P/F [43] (default).
- <model_water> = 1: flexible SPC/F2 [44].
- <model_water> = 2: Reimers–Watts–Klein (RWK) [45].

Default: "0" (q-TIP4P/F).

All three built-in WATER models support XMPI force evaluation for the XMPI-enabled methods. The variable-cell PIMD NPT/NPH/NTT integrators themselves are not XMPI-enabled; run those ensembles with ordinary MPI. XMPI TESTVIRIAL nevertheless perturbs the current box and validates the variable-cell force/virial refresh without claiming an XMPI NPT integrator.

NOTE: All three WATER models use the same automatic H_2O connectivity setup. The atom order is arbitrary. On the first force call, each H atom must have exactly one O atom within 1.4 Å under the minimum-image convention and each O must be assigned exactly two H atoms. Ambiguous, incomplete, or non-H_2O structures terminate with an input error. The recognized connectivity from bead 1 is then used unchanged for every bead and held fixed for the entire trajectory. No bead performs a new nearest-neighbor assignment, including for SPC/F.

NOTE: FREE and periodic boundaries are supported. For periodic systems, Coulomb interactions are evaluated through the CFF Ewald machinery. For q-TIP4P/F and RWK, the negative charge is carried by the model's virtual site and its force is projected back to the O/H atoms. Their CFF dipole moment is evaluated at the virtual-site coordinates before the O coordinates are restored. SPC/F uses its O/H point charges directly. Intramolecular 1–2 and 1–3 nonbonded exclusions are constructed automatically from the recognized O–H bond graph.

NOTE: WATER does not define the nuclear masses. The physical and fictitious masses used for MD/PIMD are the PIMD-side masses (for example, physmass/normal-mode masses and any mass-changing input). The data.lammps file written by WATER contains nominal H/O masses only for export compatibility; those values do not override the masses used by the PIMD dynamics. Isotopic substitution is therefore a mass-setting issue, not a change of <model_water>.

NOTE: On first initialization WATER writes input.lammps, data.lammps, and parm.lammps describing the recognized topology and the CFF part of the selected model. For q-TIP4P/F these files describe the complete built-in CFF representation. SPC/F and RWK contain model-specific residual terms in the WATER kernel;

their generated LAMMPS-style files therefore document the CFF electrostatic/topological part and are not, by themselves, complete stock LAMMPS representations of those two models.

11.4.151 <molpro_command>

Used with <ipotential> = MOLPRO. This keyword sets the execution command of MOLPRO. The data (character) should be provided in the first line. The default is “molpro”.

11.4.152 <mopac_command>

Used with <ipotential> = MOPAC. This keyword sets the execution of MOPAC. The first line gives the MOPAC execution command. The default is “nohup /opt/mopac/mopac/MOPAC2007.out”

11.4.153 <n2p2_en_unit>

Used for <ipotential> = N2P2. This keyword sets the energy unit received from the N2P2 MLP. The options are: HARTREE (default), EV (electronic volts) or a floating point number indicating the conversion factor from N2P2 units to HARTREE.

11.4.154 <n2p2_len_unit>

Used for <ipotential> = N2P2. This keyword sets the length unit for the coordinates sent to the N2P2 MLP. The options are: BOHR (default), AA (Aangstroms) or a floating point number indicating the conversion factor from N2P2 units to BOHR.

11.4.155 <ncons>

Used with <method> = MD, AFED. This keyword sets the number of constraints followed by the parameters of each constraint. In line 1, the number constraints, n (integer), is given. In lines 2-$(n+1)$, the format depends on the type of constraints. The format is as follows:

- type 1: bond distance of atoms #1–#2.
  “1, atom #1, atom #2, minimum distance [bohr], maximum distance [bohr]”.
- type 2: bond angle of atoms #1–#2–#3.
  “2, atom #1, atom #2, atom #3, minimum angle [degrees], maximum angle [degrees]”.
- type 3: dihedral angle of atoms #1–#2–#3–#4.
  “3, atom #1, atom #2, atom #3, atom #4, minimum angle [degrees], maximum angle [degrees]”.
- type 4: bond difference of atoms #1–#2 and atoms #2–#3.
  “4, atom #1, atom #2, atom #3, minimum value [bohr], maximum value [bohr]”
- type 5: coordination number of species #1 surrounded by species #2.
  “5, spec #1, spec #2, ν, μ, equilibrium distance [bohr], minimum coordination number, maximum coordination number”.
- type 6: difference in coordination numbers for species #1 surrounded by species #2 and #3 surrounded by #4.
  “6, spec #1, spec #2, ν, μ, equilibrium distance [bohr], spec #3, spec #4, ν, μ, equilibrium distance [bohr], minimum difference in coordination numbers, maximum difference in coordination numbers”.
- type 7: Cartesian coordinate of the geometric center of an atomic kind/group.
  “7, xyz component (1-3), spec #1, minimum value [bohr], maximum value [bohr]”.
- type 8: difference between geometric-center Cartesian coordinates of two atomic kinds/groups.
  “8, xyz component (1-3), spec #1, spec #2, minimum value [bohr], maximum value [bohr]”.

The constraint types, 1–8, can be replaced by “DIST”, “ANGL”, “DIH”, “DIFF’, “CN”, “DCN”, “XYZ” and “DXYZ”, respectively. For the definition of coordination number, see Section 13.14.1.

The canonical 2.8.0 fallback from input_default.dat is

```
<ncons>
0
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.156 <ncycle_poly>

Used in the POLYMER code. This keyword sets the number of update cycles of centroid images by the string method. Input: one integer on the first line. Default: "1".

11.4.157 <neighbor_list_skin>

Used for <ipotential> = N2P2. This sets the neighbor list skin distances for N2P2 calculations. The units used for this is BOHR, regardless of the length units used by the N2P2 MLP. The neighbor list radius is automatically deduced from the N2P2 input file, and this skin is added to avoid recalculating the neighbor list in each step.

The canonical 2.8.0 fallback from input_default.dat is

```
<neighbor_list_skin>
1.0
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.158 <ngrid_poly>

Used in the POLYMER code. This keyword sets finer grids to compute free energy between the images by thermodynamic integration. Input: one character string on the first line. Default: 10.

11.4.159 <ngrid_string>

Used with <method> = STRING. This keyword sets the number of fine grids to interpolate between the images of the string. Default: "11".

11.4.160 <nmeta>

Used with <method> = MTD. This keyword sets the dimension of collective variables in metadynamics, and the parameters necessary for each collective variable.

- **line 1**: the dimension of collective variables n.
- **line 2**: the parameter set of 1st collective variable.
- **line 3**: the parameter set of 2nd collective variable.
- **line** $(n+1)$: the parameter set of n-th collective variable.

In lines 2-$(n+1)$, the format depends on the type of collective variables. The format is as follows:

- type 1: bond distance of atoms #1–#2.
  "1, atom #1, atom #2".
- type 2: bond angle of atoms #1–#2–#3.
  "2, atom #1, atom #2, atom #3".
- type 3: dihedral angle of atoms #1–#2–#3–#4.
  "3, atom #1, atom #2, atom #3, atom #4".
- type 4: bond difference of atoms #1–#2 and atoms #2–#3.
  "4, atom #1, atom #2, atom #3"

- type 5: coordination number of species #1 surrounded by species #2.
  "5, spec #1, spec #2, ν, μ, distance [bohr]"
- type 6: difference coordination numbers of species #1 surrounded by species #2 and #3 surrounded by species #4.
  "6, spec #1, spec #2, ν, μ, distance [bohr], spec #3, spec #4, ν, μ, distance [bohr]"
- type 7: Cartesian coordinate of the geometric center of an atomic kind/group.
  "7, xyz component (1-3), spec #1".
- type 8: difference between geometric-center Cartesian coordinates of two atomic kinds/groups.
  "8, xyz component (1-3), spec #1, spec #2".

The collective variable types, 1–8, can be replaced by "DIST", "ANGL", "DIH", "DIFF', "CN", "DCN", "XYZ" and 'DXYZ", respectively. For the definition of coordination number, see Section 13.14.1.

From version 2.1.1, it is possible to set the boundary to each collective variables. Add two numbers, i.e., the minimum value and the maximum value, at the end of each line. If these values are not set, the free boundary is assumed.

The canonical 2.8.0 fallback from input_default.dat is

```
<nmeta>
not_available
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.161 <nmulti>

Used with <method> = MTS. This keyword controls the time increment in the multiple time scale molecular dynamics and path integral molecular dynamics using the ONIOM or QM/MM potentials. In the MTS method, the time step for updating the forces with respect to the layer A is given by <dt>. Meanwhile, the time step for updating the forces with respect to the layer B, and the interaction between the layers A and B, is set to be <dt>/<nmulti>. Thus, the parameter <nmulti> is the number of updates with respect to the changes in the layer B. In addition, the atomic masses in the layer B are set smaller that the reality, by the scaled factor of $1/\text{<nmulti>}^2$. Default: "100".

11.4.162 <np_poly>

Used in the POLYMER code. This keyword sets the parallel number with respect to centroid images. Input: one integer on the first line. Default: "1".

11.4.163 <nph_type>

Used with <ensemble> = NPH. This keyword sets the box shape of the NPH ensemble.

- For <ensemble> = NPH:
  - <nph_type> = CUBIC1: Andersen's isobaric molecular dynamics for a cubic box.
  - <nph_type> = CUBIC2: Martyna's isobaric molecular dynamics for a cubic box.
  - <nph_type> = PPHEX: Parrinello-Rahman's isobaric molecular dynamics for a parallel-piped hexahedron box.

Default: "CUBIC2".

11.4.164 <npoly>

Used in the POLYMER code. This keyword sets the the number of centroid images. Input: one integer on the first line. Default: "16".

11.4.165 <npt_type>

Used with <ensemble> = NPT. This keyword sets the box shape of the NPT ensemble.

- For <ensemble> = NPT:
  - <npt_type> = CUBIC1: Andersen's isobaric molecular dynamics for a cubic box.
  - <npt_type> = CUBIC2: Martyna's isobaric molecular dynamics for a cubic box.
  - <npt_type> = PPHEX: Parrinello-Rahman's isobaric molecular dynamics for a parallel-piped hexahedron box.

Default: "CUBIC2".

11.4.166 <nref_meta>

Used with <method> = MTD. This keyword controls the reference step size of harmonic forces between collective variable and fictitious particle in metadynamics. Input: one integer on the first line. Default: "10".

NOTE: The harmonic force is controlled by the keyword <time_fc_meta>.

11.4.167 <nref_tass>

Used with <method> = TASS. The same as nref_meta used in <method> = MTD.

The canonical 2.8.0 fallback from input_default.dat is

```
<nref_tass>
100
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.168 <nstep_modes>

Used with <method> = NMA. This keyword sets the number of steps for the normal mode vibration printed in "modes/mode.*.xyz". Default: 0 (do not print).

11.4.169 <nstate>

Used with <method> = TFS and <method> = MFE. This keyword sets the number of electronic states in nonadiabatic dynamics. Default: "1".

NOTE: If <nstate> = 1, both TFS and MFE methods are equal to the standard (Born-Oppenheimer) MD of the NVE ensemble.

11.4.170 <ntass_rec>

Used with <method> = TASS.

- line 1: CV dimension for free energy reconstruction (n)
- line 2: The n CVs for free energy reconstruction

The canonical 2.8.0 fallback from input_default.dat is

```
<ntass_rec>
-1
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.171 <nth_type>

Used with <ensemble> = NTH. This keyword sets the box type of NtH ensemble.

- For <ensemble> = NTH:
  - <nth_type> = CUBIC1: Andersen's isobaric molecular dynamics for a cubic box.
  - <nth_type> = CUBIC2: Martyna's isobaric molecular dynamics for a cubic box.
  - <nth_type> = PPHEX: Parrinello-Rahman's isobaric molecular dynamics for a parallel-piped hexahedron box.

Default: "PPHEX".

11.4.172 <ntt_type>

Used with <ensemble> = NTT. This keyword sets the box type of NtT ensemble.

- For <ensemble> = NTT:
  - <ntt_type> = CUBIC1: Andersen's isobaric molecular dynamics for a cubic box.
  - <ntt_type> = CUBIC2: Martyna's isobaric molecular dynamics for a cubic box.
  - <ntt_type> = PPHEX: Parrinello-Rahman's isobaric molecular dynamics for a parallel-piped hexahedron box.

Default: "PPHEX".

11.4.173 <ntype_aenet>

Used with <ipotential> = AENET. This keyword sets the atomic types used in the artificial neural network potential. The data in the first line (integer) should be the number of atomic types, n. The next n lines should be the atomic type, followed by the artificial neural network potential file of the atomic type. Default: not available.

11.4.174 <oniom_dat_dir>

Used with <ipotential> = ONIOM. This keyword sets the input directories of ONIOM method. Three data (character strings) should be provided in the first line, corresponding to the directory names which include the input files for the high-level calculations of subsystem A, for the low-level calculations of subsystem A, and for the low-level calculations of the system A+B. Default: "dat_1, dat_2, dat_3".

11.4.175 <oniom_hi_potential>

Used with <ipotential> = ONIOM. This keyword sets the high-level potential of ONIOM method. A data (character) should be provided in the first line, corresponding to the the high-level potential. Default: "SMASH".

11.4.176 <oniom_lo_potential>

Used with <ipotential> = ONIOM. This keyword sets the low-level potential of ONIOM method. A data (character) should be provided in the first line, corresponding to the the low-level potential. Default: "CFF".

11.4.177 <oniom_scr_dir>

Used with <ipotential> = ONIOM. This keyword sets the temporary scratch directories of ONIOM method. Three data (character strings) should be provided in the first line, corresponding to the directory names which include the scratch files for the high-level calculations of subsystem A, for the low-level calculations of subsystem A, and for the low-level calculations of the system A+B. Default: "scr_1, scr_2, scr_3".

11.4.178 <orca_command>

Used with <ipotential> = ORCA. This keyword sets the execution command of ORCA. The data (character) should be provided in the first line. The default is “orca”.

11.4.179 <params_cons>

Used when constraint parameters are required, including constrained MD and AFED-family calculations. For each line, the constraint type is followed by one real force constant. The canonical fallback values in input_default.dat are

```
DIST    5.0d+0
ANGL    5.0d-2
DIH     5.0d-3
DIFF    2.5d+0
CN      1.0d+0
DCN     1.0d+0
XYZ     5.0d+0
DXYZ    5.0d+0
```

The values are indexed by constraint type and are shared by constraints of the same type. The current reader consumes all eight rows in order; a partial override can therefore fail and fall back to the default block. New TOML should use force_constant_TYPE, which is expanded by **pimd-prep** to a complete eight-row block.

11.4.180 <params_dual>

This keyword sets the mixing weight parameter for self-learning hybrid Monte Carlo (SLHMC-MIX). Two real numbers should be provided in the first line. The first one is the weight of high-level potential. If it is 1.0 (0.0), trial moves are accepted by purely high-level (low-level) potential. This can be tuned between 0.0 and 1.0 for higher efficiency. The second number is a parameter for safety. If it is larger than 0.0, the move is rejected when the correction is too large. The default is “1.0, 0.0” (high-level potential, with no correction).

11.4.181 <minweight_train_dual>

This keyword sets an optional training-set cutoff for DUAL HMC calculations. For a trial structure evaluated with the high-level potential, define

$$w_{\mathrm{train}} = \exp[-\beta \Delta H_{\mathrm{high}}],$$

where ΔH_{high} is the change in the pure high-level HMC Hamiltonian. In NPT and NTT calculations this is the corresponding extended Hamiltonian used by the HMC move, including the active cell/barostat terms. For REHMC, the replica-specific inverse temperature is used.

When rejected-inclusive training output is requested with <ioption_xsf_aenet>, <ioption_xsf_n2p2>, or <ioption_xsf_mace> equal to 1, a trial structure is written only when w_{train} is not smaller than this cutoff. The comparison is performed in logarithmic form, so very small Boltzmann factors do not need to be evaluated explicitly. Accepted-only output (option 0) is unchanged.

The default is -1.0, which disables this filter and preserves the previous behavior. When the filter is enabled, the allowed range is $0 < w_{\mathrm{min}} \leq 1$. Thus -1.0 is the only disabled sentinel; zero and other negative values are invalid.

11.4.182 <params_lbfgs>

Used with <method> = GEOOPT, BOXOPT, FULLOPT. This keyword sets the convergence criteria in the limited memory BFGS method. The data (four real numbers) should be provided in the first line. The four numbers correspond to the maximum displacement in au (bohr), the root-mean-square displacement in au (bohr), the maximum force in au (hartree/bohr), and the root-mean-square force in au (hartree/bohr), respectively.

The canonical 2.8.0 fallback from input_default.dat is

```
<params_lbfgs>
1.d-4,  1.d-5,  1.d-5,  1.d-6
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.183 <params_rdf>

Used in the cases when <iprint_rdf> is positive. This keyword sets the meshes of the atomic pair distribution. Three real numbers should be provided in the first line, i.e., the minimum value, the maximum value, and the mesh size in au (bohr). The default values are "1.0, 20.0, 0.01".

11.4.184 <params_rec_meta>

Used with <method> = MTD. This keyword sets the mesh used to reconstruct metadynamics hills. Each line contains the CV type followed by minimum, maximum, and mesh spacing. The canonical fallback values in input_default.dat are

```
DIST   2.0    5.0   0.1
ANGL   0.0  180.0   6.0
DIH    0.0  360.0  12.0
DIFF  -3.0    3.0   0.2
CN     0.0    3.0   0.1
DCN   -2.0    2.0   0.1
XYZ    2.0    5.0   0.1
DXYZ   2.0    5.0   0.1
```

The values are indexed by CV type rather than by CV number. DIST, DIFF, XYZ, and DXYZ use bohr; ANGL and DIH use degrees; CN and DCN are dimensionless.

11.4.185 <phase0_proc>

Used when <ipotential> = PHASE/0. This keyword sets the PHASE/0 settings for parallel computation. The three integers, ne nk ng, correspond to the number of band parallels, k point parallels, G point parallels, respectively. 1 should be specified as the number corresponding to ng when a two-dimensional version is used. Each value should be determined so that the MPI parallel number allocated to each replica by PIMD (np_force) is equal to the product of ne nk ng. The default is -1, 1, 1, in this case ne is adjusted to np_force, and +nk=1+ and +ng=1+.

11.4.186 <projcmf_poly>

Used in the POLYMER code. This keyword sets the projection of mean forces applied to centroid images, either "ON" (projection applied) or "OFF" (not applied). Input: one character string on the first line. Default: "OFF".

11.4.187 <qe_input_file_name>

Used with <ipotential> = QE. This keyword sets the input file name of the QE code. The data should be provided in the first line.

Default: "qe.dat".

11.4.188 <qe_output>

Used with <ipotential> = QE. This keyword sets the log file of the QE code. The following two data should be provided in the first line.

- The option, “0” or “1”. If “0”, the log file of only the first bead is printed. If “1”, the log files of all the beads are printed.
- The print interval n (integer). The log file is printed every n steps.

The default values are “0, 1”, which means that the first bead is printed every step.

11.4.189 <qmmm_embedding>

Used with <ipotential> = QMMM. This keyword sets the embedding type of electrostatic interaction in QM/MM method. A data (character) should be provided in the first line, either “ME” (mechanical embedding) or “EE” (electronic embedding). Default: “EE”.

11.4.190 <qmmm_dat_dir>

Used with <ipotential> = QMMM and <qmmm_embedding> = ME. This keyword sets the input directory of QMMM method. A data (character) should be provided in the first line, corresponding to the directory name which includes the input files for the QM calculations of subsystem A. Default: “dat”.

11.4.191 <qmmm_potential>

Used with <ipotential> = QMMM. This keyword sets the QM potential in QM/MM method. A data (character) should be provided in the first line. Default: “SMASH”.

11.4.192 <qmmm_scr_dir>

Used with <ipotential> = QMMM and <qmmm_embedding> = ME. This keyword sets the temporary scratch directory of QMMM method. A data (character) should be provided in the first line, corresponding to the directory name which includes the scratch files for the QM calculations of subsystem A. Default: “scr”.

11.4.193 <ratio_max_hmc>

Used with <method> = PIHMC and <method> = REHMC with `istep_adjust_hmc > 0`. This keyword sets the lower bound of the target acceptance ratio. The MD steps per Metropolis step is increased if the acceptance ratio is higher than this value. Default: “0.20”.

geometric progression, respectively, for the MD steps per Metropolis step.

11.4.194 <ratio_min_hmc>

Used with <method> = PIHMC and <method> = REHMC with `istep_adjust_hmc > 0`. This keyword sets the lower bound of the target acceptance ratio. The MD steps per Metropolis step is decreased if the acceptance ratio is lower than this value. Default: “0.05”.

11.4.195 <scale_bcmd>

A scaling factor for the friction constant in BCMD. The default is 1.0 (not scaled).

11.4.196 <scale_trpmd>

Used in <method> = TRPMD. This keyword is the dimensionless TRPMD damping parameter λ. For each non-centroid ring-polymer normal mode k, the current source uses

$$\gamma_k = 2\,\lambda\,\widetilde{\omega}_k,$$

i.e. `gamma = scale_trpmd * (2 * omega_mode)` in the TRPMD cycle routines. The default is `0.5`, corresponding to $\lambda = 1/2$ in Rossi, Ceriotti, and Manolopoulos [34]. In their convention $\lambda = 1$ is critical damping, $\lambda = 0$ recovers RPMD, and $\lambda = 1/2$ was found to be close to optimal for the systems studied.

11.4.197 <scan_exe_shell>

Used with <method> = SCAN. This keyword sets the shell command executed for each structure in the SCAN method. A data (character) should be provided in the first line, corresponding to the part of the file name of shell command (*name*). The file name is *name.integer*.sh. The *integer* is the bead number with three digits. Default: "./scan". By default (<nbead> = 1) the file "./scan.001.sh" is executed for each structure. Otherwise, for instance, <nbead> = 4, the files "./scan.001.sh" "./scan.002.sh" "./scan.003.sh" and "./scan.004.sh" are executed for bead 1, 2, 3 and 4, respectively.

11.4.198 <skin_aenet>

Used with <ipotential> = AENET. This keyword sets the skin value of neighbor list in AENET. A real value should be provided in the first line in angstroms. Default: 2.0 angstroms.

11.4.199 <smash_dip>

For embedded <ipotential> = SMASH (3.0.2), "1" returns the HF/DFT SCF dipole in atomic units and "0" disables its contribution to PIMD. Default: "1". For dynamics, <iprint_dip> controls dipole.out output. MP2 dipoles are unsupported, set this flag to "0" for MP2 calculations. QM/MM handles dipoles independently of this flag.

11.4.200 <smash_grad>

Used with the CMake embedded <ipotential> = SMASH implementation (3.0.2). This keyword controls whether analytic gradients are requested: "1" (on) or "0" (energy only). Default: "1". QM/MM calculations always request gradients and do not use this flag.

11.4.201 <smash_guess>

For embedded SMASH 3.0.2, including QM/MM, PREVIOUS (default) reuses the previous converged checkpoint for each bead. The first call uses SMASH's native initial guess. With DEFAULT, every call uses the native guess configured in <smash_options> (extended Hückel by default). An explicit native guess=check still requests a checkpoint. Other values are rejected by this interface.

The following extrapolation options apply only to the legacy internally linked <ipotential> = SMASH2 implementation (-Dsmash). This keyword selects extrapolation of molecular-orbital coefficients between PIMD force calls. The current source recognizes <smash_guess> = PREVIOUS (no extrapolation), <smash_guess> = PULAY or PULAY-4 (four-point Pulay), KOLAFA-2, KOLAFA-4, <smash_guess> = KOLAFA or KOLAFA-6, and <smash_guess> = NIKLASSON (six-point Niklasson extrapolation). An unrecognized value falls back internally to no coefficient extrapolation. Default: PREVIOUS.

11.4.202 <smash_options>

Used by internally linked <ipotential> = SMASH2 (compiled with -Dsmash) and CMake embedded <ipotential> = SMASH (3.0.2), including QM/MM EE and ME. This block is required for both internally linked versions. The PIMD patch to SMASH fileio.F90 reads this block directly from input.dat; it does not use input_default.dat. It contains the ordinary SMASH input records beginning with 'job', 'control', etc.; basis-set and ECP records may also be included. The 'geom' block and coordinates are omitted because PIMD supplies the coordinates internally.

11.4.203 <temp_tamd_tass>

Scaled temperature in TASS. If this is negative, it is set to the physical temperature set by the keyword <temperature>. The default is -1 (physical temperature).

11.4.204 <temprange_phonon>

Used with <method> = PHONON. This keyword sets the temperature range of phonon energy calculation. Three real numbers should be provided in the first line, i.e., the minimum value, the maximum value, and the increment in kelvin. The default values are “0.0, 3000.0, 10.0”.

11.4.205 <temprange_rem>

Used for temperature-replica exchange in <method> = REMC and <method> = REHMC, including the NPT REHMC path when the selected replica-exchange mode requires a temperature range. Two real numbers, the first- and last-replica temperatures in kelvin, are read directly from input.dat. The intermediate replica temperatures are placed geometrically between these endpoints using <nbead> as the number of replicas. There is no canonical fallback in input_default.dat; when the selected replica-exchange mode needs this range the tag is required explicitly.

11.4.206 <tension>

Used with <ensemble> = NTH, NTT. This keyword sets the tension matrix.

t_{xx}, t_{xy}, t_{xz}
t_{xy}, t_{yy}, t_{yz}
t_{xz}, t_{yz}, t_{zz}
The default values are nine zeros (zero tension).

NOTE: the tension matrix must be real and symmetric.

11.4.207 <testforce_tolerance>

Absolute acceptance tolerance for <method> = TESTFORCE in hartree/bohr. The default is 10^{-7}. Input: one finite, positive real number. A force component passes when the full-precision absolute difference is strictly smaller than the selected tolerance.

11.4.208 <testvirial_tolerance>

Absolute acceptance tolerance for <method> = TESTVIRIAL in hartree/bohr. The default is 10^{-5}. Input: one finite, positive real number. A cell-derivative component passes when the full-precision absolute difference is strictly smaller than the selected tolerance.

11.4.209 <time_baro>

Used with <ensemble> = NPT, NTT, NPH, NTH. This keyword controls the mass of barostat. The mass of barostat is related to the characteristic time scale of the system. This keyword sets this time scale in femtoseconds. Input: one real number on the first line. Default: “1000.0” femtoseconds.

11.4.210 <time_cv_bath>

Used with <method> = MTD. This keyword sets the characteristic time scale of thermostat attached to fictitious particle in femtoseconds. Input: one real number on the first line. Default: “1000.0” femtoseconds.

11.4.211 <time_cv_meta>

Used with <method> = MTD. This keyword sets characteristic time scale, in femtoseconds, for the fictitious particle to travel 1.5 times of the Gaussian hill width, It is therefore the characteristic time interval of adding Gaussian hills. By controlling the mass of the fictitious particle, the adiabatic separation between the collective variables (slow) and fast all other degrees of freedom (fast) is controlled. A better adiabaticity and thus higher accuracy should be accomplished as the <time_cv_meta> value is larger. Input: one real number on the first line. Default: “75.0” femtoseconds.

11.4.212 <time_fc_meta>

Used with <method> = MTD. This keyword sets time scale of oscillation, in femtoseconds, that controls the the strength of harmonic force between collective variable and fictitious particle. The value should be large enough so that harmonic interaction is strong, keeping the collective variable and the position of fictitious particle close to each other. Input: one real number on the first line. Default: "30.0" femtoseconds.

NOTE: The time increment of updating of the harmonic force is controlled by the keyword <nref_meta>.

11.4.213 <time_limit_meta>

Used with <method> = MTD. This keyword sets the time limit of adding a new hill in femtoseconds. It is recommended to set this value at least more than three times as large as the value of <time_cv_meta>. Input: one real number on the first line. Default: "225.0" femtoseconds, three times the default value of <time_cv_meta>.

11.4.214 <time_mode>

The time scale of fictitious non-centroid mass in fs. If this is negative, it is set to the standard value. The default is -1 (standard value).

11.4.215 <train_aenet>

Used with <ipotential> = AENET when the AENET training path is invoked. This keyword sets the AENET training input. The current source opens input.dat, locates <train_aenet>, and copies the following non-blank lines verbatim into train.in. There is no input_default.dat fallback; the block must therefore be provided explicitly in input.dat whenever this training path is used.

11.4.216 <turbo_command>

Used with <ipotential> = TURBOMOLE. This keyword sets the execution of TURBOMOLE. Two execution commands should be given to compute energy, forces, and dipole moment. The first line gives the first TURBOMOLE execution command followed by the name of the first control file. The second line gives the second TURBOMOLE execution command followed by the name of the second control file.

The canonical 2.8.0 fallback from input_default.dat is

```
<turbo_command>
dscf    control.1
grad    control.2
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.4.217 <turbo_guess>

Used with <ipotential> = TURBOMOLE. This keyword gives the option for the initial guess of molecular orbital coefficients.

- <turbo_guess> = PREVIOUS: Molecular orbital coefficients of the last step is used as the initial guess.
- <turbo_guess> = PULAY: Extrapolation of the molecular orbital coefficients of the last four steps are used as the initial guess using Pulay's parameter set [18].
- <turbo_guess> = KOLAFA, Extrapolation of the molecular orbital coefficients of the last six steps are used as the initial guess using Kolafa's parameter set [19].

Default: PREVIOUS.

NOTE: For safe execution of TURBOMOLE, it is recommended to set this value to PREVIOUS.

11.4.218 <user_command>

Used with <ipotential> = USER. This keyword gives the user execution command to calculate potential, forces, and optionally, virial, for a given atomic positions. Default: not available.

11.4.219 <user_input_file>

Used with <ipotential> = USER. This keyword gives the name of the input file of user execution command. The input file should be in the following format. For a single bead case,

```
x1   y1   z1   (Cartesian coordinates of atom 1 in bohr)
x2   y2   z2   (Cartesian coordinates of atom 2 in bohr)
..   ..   ..
..   ..   ..
xn   yn   yz   (Cartesian coordinates of atom n in bohr)
```

For a multiple bead case,

```
x1   y1   z1   (Cartesian coordinates of atom 1 of bead 1 in bohr)
x2   y2   z2   (Cartesian coordinates of atom 2 of bead 1 in bohr)
..   ..   ..
xn   yn   yn   (Cartesian coordinates of atom n of bead 1 in bohr)
x1   y1   z1   (Cartesian coordinates of atom 1 of bead 2 in bohr)
x2   y2   z2   (Cartesian coordinates of atom 2 of bead 2 in bohr)
..   ..   ..
xn   yn   yn   (Cartesian coordinates of atom n of bead 2 in bohr)
..   ..   ..
..   ..   ..
x1   y1   z1   (Cartesian coordinates of atom 1 of bead m in bohr)
x2   y2   z2   (Cartesian coordinates of atom 2 of bead m in bohr)
..   ..   ..
xn   yn   yn   (Cartesian coordinates of atom n of bead m in bohr)
```

Default: not available.

11.4.220 <user_output_file>

Used with <ipotential> = USER. This keyword gives the name of the output file of user execution command. The output file should be in the following format. For a single bead case,

```
pot             (potential)
fx1  fy1  fz1   (force of atom 1 in bohr)
fx2  fy2  fz2   (force of atom 2 in bohr)
...  ...  ...
fxn  fyn  fyn   (forces of atom n in hartree/bohr)
v11  v12  v13   (virial matrix in au, optional)
v21  v22  v23   (virial matrix in au, optional)
v31  v32  v33   (virial matrix in au, optional)
```

For a multiple bead case,

```
pot             (potential of bead 1)
fx1  fy1  fz1   (force of atom 1 of bead 1 in bohr)
fx2  fy2  fz2   (force of atom 2 of bead 1 in bohr)
...  ...  ...
fxn  fyn  fyn   (force of atom n of bead 1 in hartree/bohr)
fx1  fy1  fz1   (force of atom 1 of bead 2 in hartree/bohr)
fx2  fy2  fz2   (force of atom 2 of bead 2 in hartree/bohr)
...  ...  ...
fxn  fyn  fyn   (force of atom n of bead 2 in hartree/bohr)
```

```
...  ...  ...
...  ...  ...
fx1  fy1  fz1   (force of atom 1 of bead m in hartree/bohr)
fx2  fy2  fz2   (force of atom 2 of bead m in hartree/bohr)
...  ...  ...
fxn  fyn  fyn   (force of atom n of bead m in hartree/bohr)
v11  v12  v13   (virial matrix in au, optional)
v21  v22  v23   (virial matrix in au, optional)
v31  v32  v33   (virial matrix in au, optional)
```

Default: not available.

11.4.221 <vasp_command>

Used with <ipotential> = VASP5. This keyword sets the execution command of the VASP code. The data should be provided in the first line. The default is "mpirun -np 4 vasp > output.vasp".

11.4.222 <vasp_energy>

Used with <ipotential> = VASP6. This keyword sets which energy is picked up from the VASP calculation. Default: FREE_ENERGY, which picks up the energy corresponding to the line containing "free energy TOTEN" in the outcar file. It is also possible to set this to EXTRAPOLATED, in which case the energy on the line containing "energy(sigma->0)" is selected. All other strings are not accepted as a valid input to this keyword. In both cases the energy is transferred directly from the VASP code and not actually read from the OUTCAR file.

11.4.223 <vasp_keyword_energy>

Used with <ipotential> = VASP5. The current VASP5 interface reads three values from a single line: (1) the output-search string, (2) the line offset from the matched line, and (3) the column from which the energy is extracted. The canonical fallback in input_default.dat is

```
'FREE ENERGIE OF THE ION-ELECTRON SYSTEM (eV)',  2,  5
```

The VASP5 run-directory input.dat is searched first and its input_default.dat is used as fallback.

11.4.224 <vasp_keyword_gradient>

Used with <ipotential> = VASP5. The current VASP5 interface reads three values from a single line: (1) the output-search string, (2) the line offset from the matched line, and (3) the starting column for the force/gradient records. The canonical fallback in input_default.dat is

```
'POSITION                                       TOTAL-FORCE (eV/Angst)',  2,  4
```

The VASP5 run-directory input.dat is searched first and its input_default.dat is used as fallback.

11.4.225 <vasp_keyword_stress>

Used with <ipotential> = VASP5. The current VASP5 interface reads three values from a single line: (1) the output-search string, (2) the line offset from the matched line, and (3) the starting column for the stress records. The canonical fallback in input_default.dat is

```
'FORCE on cell =-STRESS in cart. coord.  units (eV):',  14,  3
```

The VASP5 run-directory input.dat is searched first and its input_default.dat is used as fallback.

11.4.226 <vasp_output>

Used with <ipotential> = VASP. This keyword sets the log file of the VASP code. The following two data should be provided in the first line.

- The option, "0" or "1". If "0", the log file of only the first bead is printed. If "1", the log files of all the beads are printed.
- The print interval n (integer). The log file is printed every n steps.

The default values are "0, 1", which means that the first bead is printed every step.

11.4.227 <vasp_reuse_wavefunction>

Used with <ipotential> = VASP. This keyword controls the reuse the guess of Kohn-Sham (KS) orbitals of VASP. An integer, "0" or "1", should be provided in the first line. If "0", the guess of the KS orbitals is built from scratch at each step. If "1", the KS molecular orbitals of the last step are reused as the guesses for the next step. Default: "1" (reuse).

11.4.228 <xmpi>

This keyword selects the extensively parallel MPI (XMPI) decomposition in pimd.mpi.x. The user values are "OFF" or "ON"; "DEBUG" is reserved for the internal RPMD/NVE diagnostic path. "OFF" uses the ordinary MPI algorithms. "ON" decomposes work into <np_beads> bead groups, each containing <np_force> ranks. The total MPI size must therefore equal <np_beads>×<np_force>. If either value is supplied explicitly and MPI initialization would have to rewrite it, XMPI terminates with an error rather than silently using a different decomposition. The historical automatic adjustment remains available to ordinary MPI. Current XMPI method support is:

Method	Ensemble / thermostat	XMPI
TESTFORCE	–	yes
TESTVIRIAL	–	yes
GEOOPT	–	yes
PIMD	NVT / MNHC	yes
CMD	NVE	yes
RPMD	NVE	yes
BCMD	NVE or NVT	yes

Other methods and ensembles use ordinary MPI and must be run with <xmpi>="OFF". In particular, variable-cell PIMD (NPH/NPT/NTT), RPMD/NVT, TRPMD, MD, and PIHMC are not XMPI integrators. The program checks this at startup and terminates with an error instead of silently ignoring an unsupported XMPI request. The force backend must also have an XMPI implementation. Current backends are N2P2, WATER, CFF, QM/MM mechanical embedding, ONIOM, ASE, MACE, AENET, ACCELNET, MTP, ADP, TERSOFF, METAL-WATER, EAM, PAIR, HELIUM, and HYDROGEN. Optional backends such as N2P2, ASE, MACE, AENET, and ACCELNET must be enabled in the executable at compile time; otherwise the startup check rejects them. QM/MM electronic embedding is not XMPI-enabled. N2P2, AENET, and MACE are inference-only XMPI paths; on-the-fly training (and N2P2 XSF-generation analysis) requires ordinary MPI. See Section 10 for additional CFF-specific restrictions. ACCELNET supports fixed-model inference only (Section 6.4.2). For TOML input, use `[parallel].xmpi = true` (or "ON"). Default: "OFF".

11.4.229 <xtb_exe_command>

Used with <ipotential> = XTB. This keyword sets the execution command of the XTB code. The data should be provided in the first line. The default is "xtb xtb.xyz –chrg 0 –uhf 0 –grad –vparam xtb.dat > xtb.out".

11.5 Conditional and auxiliary keywords in input.dat

These entries are conditional rather than universally optional. Some are required once a feature is selected—for example, <components> and <molecules> are required for <method> = ROTOR.

11.5.1 <atom_change>

This keyword resets the atomic masses of specified atoms.

- Line 1: Number of changes (n).
- Lines 2–($n + 1$): The type of change, the initial atom, and the final atom.

The type of change can be chosen from the following options.

- FREEZE or HEAVY: Reset the atomic mass (the fictitious centroid mass for path integrals) to a large value.
- HYDROGEN: Reset the atomic mass (the fictitious centroid mass) to the hydrogen mass.

The canonical 2.8.0 fallback from input_default.dat is

```
<atom_change>
0
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.5.2 <charge_libnnp>

Used with <ipotential> = LIBNNP. This real value sets the reference charge used by the LIBNNP dipole-surface interface. It is read from input.dat, with fallback to input_default.dat; the canonical default is 0.d0.

11.5.3 <components>

Used when <method> = ROTOR. Also used by the prep_liquid.x code. This keyword defines the molecular components. The data should be provided in the following format.

- The first line: the number of molecular components, n (integer).
- The next n blocks: data for each molecular component.

Each of the latter blocks has the following format. The block is read directly from input.dat; there is no input_default.dat fallback. Therefore <components> is required when <method> = ROTOR is used.

- The first line contains three entries: the name of the molecule (character), the number of atoms per molecule m (integer), and the molecular type (character). The molecular type must be chosen from the following options.
  - GENERAL: the molecule has a nonlinear structure with six degrees of freedom.
  - LINEAR: the molecule has a linear structure with five degrees of freedom.
  - MONOATOM: the molecule is monoatomic and has three degrees of freedom.
- Each of the next m lines contains three values, corresponding to the coordinates x, y, and z of each atom in the molecule, given in bohr.

11.5.4 <file_dms_libnnp>

Used with <ipotential> = LIBNNP when the dipole-moment surface is enabled. This character value names the LIBNNP input file for the dipole-moment neural network. It is read from input.dat, with fallback to input_default.dat; the canonical default is NN-DMS/input.nn.

11.5.5 <file_ip_libnnp>

Used with <ipotential> = LIBNNP when the separate interaction-potential network is enabled. This character value names the corresponding LIBNNP input file. It is read from input.dat, with fallback to input_default.dat; the canonical default is NN-IP/input.nn.

11.5.6 <file_pes_libnnp>

Used with <ipotential> = LIBNNP. This character value names the LIBNNP input file for the potential-energy/force neural network. It is read from input.dat, with fallback to input_default.dat; the canonical default is NN-PES/input.nn.

11.5.7 <ioption_dms_libnnp>

Used with <ipotential> = LIBNNP. Set to 1 to initialize and evaluate the dipole-moment neural network and to 0 to disable it. It is read from input.dat, with fallback to input_default.dat; the canonical default is 1.

11.5.8 <molecules>

Used when <method> = ROTOR. Also used by the **prep_liquid.x** code. This keyword specifies the atoms belonging to each molecule.

- The first line: the number of molecules, n (integer).
- The next n lines: the name of the molecule and the atoms that belong to that molecule. The molecule name must be chosen from those defined by the keyword <components>.

The block is read directly from input.dat; there is no input_default.dat fallback. Therefore <molecules> is required when <method> = ROTOR is used.

11.5.9 <natom_ip_libnnp>

Used with <ipotential> = LIBNNP. This integer is the number of atoms handled by the separate interaction-potential neural network. A value of zero disables that third network; values greater than or equal to one enable it, while the main PES network is applied to the remaining atoms. It is read from input.dat, with fallback to input_default.dat; the canonical default is 0.

11.6 Keywords in calc.dat

11.6.1 <ncalc>

This keyword sets the statistical analysis.

- **line 1**: number of variables, n.
- **lines 2-(n+1)**: information for each variable (analysis keywords, followed by integers specifying atoms or species, and the names of output files).

The default value is “0” (no variables). The analysis keywords should be chosen from the following:

- `lin.atom.list, #1, #2, file name`
  Generates a list of bond lengths between atoms #1 and #2.
- `angl.atom.list, #1, #2, #3, file name`
  Generates a list of bond angles for atoms #1-#2-#3.
- `dih.atom.list, #1, #2, #3, #4, file name`
  Generates a list of dihedral angles for atoms #1-#2-#3-#4.
- `diff.atom.list, #1, #2, #3, file name`
  Generates a list of bond length differences between atoms #1-#2 and #2-#3.

- lin.spec.list, #1, #2, file name
  Generates a list of bond lengths between atomic species #1 and #2.
- angl.spec.list, #1, #2, #3, file name
  Generates a list of bond angles for atomic species #1-#2-#3.
- dih.spec.list, #1, #2, #3, #4, file name
  Generates a list of dihedral angles for atomic species #1-#2-#3-#4.
- diff.spec.list, #1, #2, #3, file name
  Generates a list of bond length differences between atomic species #1-#2 and #2-#3.
- sdiff.spec.list, #1, #2, #3, file name
  Generates a list of absolute bond length differences between atomic species #1-#2 and #2-#3.
- lin.atom.dens, #1, #2, file name
  Generates a density profile of bond lengths between atoms #1 and #2. The mesh size of the density profile is set by the keyword <params_lin_dens>.
- angl.atom.dens, #1, #2, #3, file name
  Generates a density profile of bond angles for atoms #1-#2-#3. The mesh size of the density profile is set by the keyword <params_angl_dens>.
- dih.atom.dens, #1, #2, #3, #4, file name
  Generates a density profile of dihedral angles for atoms #1-#2-#3-#4. The mesh size of the density profile is set by the keyword <params_dih_dens>.
- diff.atom.dens, #1, #2, #3, file name
  Generates a density profile of bond length differences between atoms #1-#2 and #2-#3. The mesh size of the density profile is set by the keyword <params_diff_dens>.
- sdiff.atom.dens, #1, #2, #3, file name
  Generates a density profile of absolute bond length differences between atoms #1-#2 and #2-#3. The mesh size of the density profile is set by the keyword <params_sdiff_dens>.
- lin.spec.dens, #1, #2, file name
  Generates a density profile of bond lengths of atomic species #1 and #2. The mesh size of the density profile is set by the keyword <params_lin_dens>.
- angl.spec.dens, #1, #2, #3, file name
  Generates a density profile of bond angles of atomic species #1-#2-#3. The mesh size of the density profile is set by the keyword <params_angl_dens>.
- dih.spec.dens, #1, #2, #3, #4, file name
  Generates a density profile of dihedral angles of atomic species #1-#2-#3-#4. The mesh size of the density profile is set by the keyword <params_dih_dens>.
- diff.spec.dens, #1, #2, #3, file name
  Generates a density profile of bond length differences between atomic species #1-#2 and #2-#3. The mesh size of the density profile is set by the keyword <params_diff_dens>.
- sdiff.spec.dens, #1, #2, #3, file name
  Generates a density profile of absolute bond length differences between atomic species #1-#2 and #2-#3. The mesh size of the density profile is set by the keyword <params_sdiff_dens>.
- lin.atom.avg, #1, #2, file name
  Calculates the average and dispersion of bond lengths between atoms #1 and #2.
- angl.atom.avg, #1, #2, #3, file name
  Calculates the average and dispersion of bond angles for atoms #1-#2-#3.
- dih.atom.avg, #1, #2, #3, #4, file name
  Calculates the average and dispersion of dihedral angles for atoms #1-#2-#3-#4.
- diff.atom.avg, #1, #2, #3, #4, file name
  Calculates the average and dispersion of bond length differences between atoms #1-#2 and #2-#3.
- lin.spec.avg, #1, #2, file name
  Calculates the average and dispersion of bond lengths between atomic species #1 and #2.
- angl.spec.avg, #1, #2, #3, file name
  Calculates the average and dispersion of bond angles of atomic species #1-#2-#3.
- dih.spec.avg, #1, #2, #3, #4, file name
  Calculates the average and dispersion of dihedral angles of atomic species #1-#2-#3-#4.

- diff.spec.avg, #1, #2, #3, #4, file name
  Calculates the average and dispersion of bond length differences between atomic species #1-#2 and #2-#3.
- xla.spec.dens, #1, #2, #3, file name
  Generates a two-dimensional density profile between bond lengths of atomic species #1-#2 and bond angles of atomic species #1-#2-#3. The mesh sizes of the density profile are set by the keywords <params_lin_dens> and <params_angl_dens>.
- 3d.spec.cube, #1, #2, #3, #4, file name
  Generates a three-dimensional density profile of atomic species #1 using the frame defined by #2, #3, and #4. The mesh sizes of the density profile are set by the keyword <3d_range_cube>.

The species numbers correspond to those in the file structure.dat. **NOTE:** In the calculation of bond angles and dihedral angles, the Jacobian is not included.

11.6.2 <iprint_calc>

This keyword sets the print interval of statistical analysis. It is read from calc.dat by the CALC post-processing program; if absent or malformed there, the parser reads the same tag from calc_default.dat. The data should be provided in the first line (integer). The default value in calc_default.dat is "1" (every step).

11.6.3 <iprint_std_calc>

This keyword sets the print interval of standard output. The data should be provided in the first line (integer). The default value is "100" (every 100 steps).

11.6.4 <iconf_fin_calc>

This keyword sets the final step number read from trj.out. The data should be provided in the first line (integer). The default value is "-1" (the last step in trj.out).

11.6.5 <iconf_ini_calc>

This keyword sets the initial step number read from trj.out. The data should be provided in the first line (integer). The default value is 1 (the first step in trj.out).

11.6.6 <jformat_xyz>

This keyword sets the print option of the xyz trajectory. It is read from calc.dat; if absent or malformed there, the parser reads calc_default.dat. One of the following options can be chosen:

- jformat_xyz = 1: print each bead configuration separately.
- jformat_xyz = 2: print all beads into a single configuration.
- jformat_xyz = 3: print only the centroid configuration.
- jformat_xyz = 4: print only the selected bead defined by the keyword <jxyz_bead>.

The data should be provided in the first line (integer). The default value in calc_default.dat is "2".

11.6.7 <jorigin_xyz>

This keyword sets the origin option for the xyz trajectory. It is read from calc.dat; if absent or malformed there, the parser reads calc_default.dat. One of the following options can be chosen:

- jorigin_xyz = 1: origin is $(0.0, 0.0, 0.0)$.
- jorigin_xyz = 2: Cartesian coordinates of the origin are defined by the keyword <jxyz_origin>.

- jorigin_xyz = 3: origin is fixed to an atom defined by the keyword <jxyz_atom>.

The data should be provided in the first line (integer). The default value in calc_default.dat is “1”.

11.6.8 <jprint_xyz>

This keyword sets the print interval of the xyz trajectory. It is read from calc.dat; if absent or malformed there, the parser reads calc_default.dat. The data should be provided in the first line (integer). The default value in calc_default.dat is -1 (do not print).

11.6.9 <jspec_xyz>

Used when <jprint_xyz> is positive. This keyword specifies the species to be printed in the output file calc.xyz. Two integers should be provided in the first line, indicating the first species to be printed and the last species to be printed. Exceptionally, “-1” refers to the last species in the system. The default values are “1” (first species in the system) and “-1” (last species in the system), meaning that all species are printed.

11.6.10 <jxyz_atom>

This keyword sets the central atom of the xyz trajectory. It is activated only when jorigin_xyz = 3. In that branch the source searches calc.dat directly and, if the tag or value is unavailable, searches calc_default.dat. The data should be provided in the first line (integer). The default value in calc_default.dat is 1 (the first atom).

11.6.11 <jxyz_bead>

This keyword sets the bead number of the xyz trajectory. It is activated only when jformat_xyz = 4. It is read from calc.dat; if absent or malformed there, the parser reads calc_default.dat. The data should be provided in the first line (integer). The default value in calc_default.dat is “1” (the first bead).

11.6.12 <jxyz_origin>

This keyword sets the origin of the xyz trajectory. It is activated only when jorigin_xyz = 2. In that branch the source searches calc.dat directly and, if the tag or value is unavailable, searches calc_default.dat. The data should be provided in the first line (three real numbers). The default values in calc_default.dat are “0.0, 0.0, 0.0”.

11.6.13 <params_lin_dens>

This keyword sets the mesh for bond length distributions. Three real numbers should be provided in the first line: the minimum value, the maximum value, and the mesh size in au (bohr).

11.6.14 <params_adiff_dens>

This keyword sets the mesh for absolute bond length difference distributions. Three real numbers should be provided in the first line: the minimum value, the maximum value, and the mesh size in bohrs. It is read first from calc.dat. If absent there, the current source reads calc_default.dat. The distributed 2.8.0 CALC examples use the fallback “0.0 3.0 0.01” (bohr, bohr, bohr).

11.6.15 <params_angl_dens>

This keyword sets the mesh for bond angle distributions. Three real numbers should be provided in the first line: the minimum value, the maximum value, and the mesh size in degrees.

11.6.16 <params_diff_dens>

This keyword sets the mesh for bond length difference distributions. Three real numbers should be provided in the first line: the minimum value, the maximum value, and the mesh size in bohrs.

11.6.17 <params_dih_dens>

This keyword sets the mesh for dihedral angle distributions. Three real numbers should be provided in the first line: the minimum value, the maximum value, and the mesh size in degrees.

11.6.18 <3d_range_cube>

This keyword sets the range of three-dimensional atomic distributions as follows:

```
minimum of x, maximum of x, mesh of x in bohr units
minimum of y, maximum of y, mesh of y in bohr units
minimum of z, maximum of z, mesh of z in bohr units
```

11.7 EAM, ADP, and tabulated-pair LAMMPS input

The EAM potential has the form [46, 47]

$$V_{\text{eam}} = \sum_i F_{s_i}(\rho_i) + \sum_i \sum_{j>i} \phi_{s_i,s_j}(r_{ij}), \tag{10}$$

$$\rho_i = \sum_{j\neq i} \varrho_{s_j}(r_{ij}). \tag{11}$$

For <ipotential> = EAM, these functions are read from a LAMMPS DYNAMO setfl file selected by `pair_style eam/alloy` and `pair_coeff * * FILE ELEMENT...`. PIMD follows the setfl convention that the pair table stores $r\phi(r)$. The setfl density grid `Nrho,drho` and radial grid `Nr,dr` are independent and are stored without forcing the two grid lengths to be equal.

The ADP potential [51] augments the EAM energy by dipole- and quadrupole-like terms constructed from the tabulated functions $u(r)$ and $w(r)$. With <ipotential> = ADP, PIMD reads the standard LAMMPS extended-setfl ordering selected by `pair_style adp`.

For <ipotential> = PAIR, PIMD accepts `pair_style table spline N` and explicit numeric `pair_coeff i j FILE KEY [cutoff]` records. The table section follows the ordinary LAMMPS table layout with an `N` header and rows containing index, distance, energy, and force. PIMD uses the distance and energy columns for its own spline evaluation.

All three interfaces require `units metal`. The neighbor-list skin is read from the first numerical argument of the LAMMPS `neighbor` command; if that command is omitted, PIMD uses a 2.0 Å skin. PIMD trajectory controls (ensemble, time step, bead count, thermostats, and XMPI layout) remain PIMD controls and are not taken from LAMMPS `fix` or `run` commands.

11.8 LAMMPS-compatible Tersoff input

For <ipotential> = TERSOFF, the source of truth is `input.lammps`, not a PIMD-specific Tersoff parameter block. The minimum supported form is

```
units metal
pair_style tersoff
pair_coeff * * potential.tersoff Element1 Element2 ...
```

The element list after the potential-file name maps PIMD atom types to element labels in the same order as the PIMD kinds. Exactly one mapping is required for each atom type, and `NULL` mappings are rejected

for standalone TERSOFF calculations. `pair_style tersoff` must not carry style options; in particular, the LAMMPS `shift` option is not supported. `units metal` must be stated explicitly.

The referenced parameter file follows the standard LAMMPS Tersoff format. Each ordered triplet has 17 fields,

```
element1 element2 element3 m gamma lambda3 c d costheta0
n beta lambda2 B R D lambda1 A
```

and an entry may span more than one physical line. PIMD reads the file as a token stream, accepts $m = 1$ and $m = 3$, and requires every ordered triplet needed by the mapped element set. Missing or duplicate required triplets, unsupported m values, and malformed parameter streams terminate explicitly.

The Tersoff cutoff represented by the LAMMPS parameters is $R \pm D$. Neighbor-list padding is independent of that physical cutoff. It can be supplied with

```
neighbor 2.0 bin
```

in `input.lammps`; when omitted, PIMD uses a 2.0 Å skin. `include` directives are supported. A `read_data` command is optional for interoperability with stock LAMMPS and is not required by the native PIMD force path.

The distributed example `examples/03_optimization_properties/diamond/boxopt_tersoff` uses `C.tersoff` through this interface.

11.9 Common keywords shared with constrained MD and metadynamics

11.9.1 <ncons>

See Section 11.4.155.

11.9.2 <nref>

See Section 11.3.12.

11.9.3 <params_cons>

See Section 11.4.179.

11.10 Common keywords shared in all types of AFED

11.10.1 <afed_type>

Used in the cases of <method> = AFED. This keyword sets the type of adiabatic free energy dynamics. The data should be provided in the first line (character). The options are as follows.

- GRAD: Calculation of the free energy gradient with a constant constraint.
- DESCENT: Descent search for free energy minimum points using the steepest descent method.
- ASCENT: Ascent search for free energy saddle points.
- AUTO: Automated search for free energy stationary points by combining DESCENT and ASCENT.
- TAMD: Temperature accelerated molecular dynamics.
- LOGMFD: Logarithmic mean force dynamics.

The default value is “GRAD”.

11.10.2 <fenergy_max_afed>

Used in the cases of <method> = AFED, with <afed_type> = DESCENT, ASCENT, AUTO, TAMD, LOGMFD. This keyword sets the upper bound of the free energy to be explored in adiabatic free energy dynamics. The data should be provided in the first line (real number). The default value is "0.16" hartree (about 100 kcal/mol).

11.10.3 <mdcycle_pro_afed>

Used in all cases of <method> = AFED. This keyword sets the ensemble of productive MD runs used to compute the average free energy gradients in adiabatic free energy dynamics. The data should be provided in the first line (character).

- NVT: NVT ensemble.
- NVE: NVE ensemble.

The default value is "NVT".

11.10.4 <niter_afed>

Used in all cases of <method> = AFED and <afed_type> = DESCENT, ASCENT, AUTO, TAMD, LOGMFD.

- For <afed_type> = DESCENT, ASCENT, AUTO: This keyword sets the maximum number of AFED steps (iterations) in adiabatic free energy dynamics.
- For <afed_type> = TAMD, LOGMFD: This keyword sets the number of AFED steps (iterations) in temperature accelerated molecular dynamics and logarithmic mean force dynamics.

The data should be provided in the first line (real number). The default value is "200" steps.

11.10.5 <nstep_pre_afed>

Used in all cases of <method> = AFED. This keyword sets the number of preliminary MD steps in adiabatic free energy dynamics to equilibrate the system under constraint. The data should be provided in the first line (real number). The default value is "1000" steps.

11.10.6 <nstep_pro_afed>

Used in all cases of <method> = AFED. This keyword sets the number of productive MD steps used to compute the average free energy gradients in adiabatic free energy dynamics. The data should be provided in the first line (real number). The default value is "2000" steps.

11.10.7 <params_afed>

Used in all cases of <method> = AFED. This keyword sets the CV-dependent parameters for adiabatic free-energy dynamics. Each line contains the CV type followed by exactly three real numbers: the reference increment $\Delta\xi_d^{\mathrm{ref}}$, the lower bound ξ_d^{min}, and the upper bound ξ_d^{max}. The canonical fallback values in input_default.dat are

```
DIST  0.08d0    2.0d0   10.0d0
ANGL  3.00d0    0.0d0  180.0d0
DIH   5.00d0 -180.0d0  180.0d0
DIFF  0.08d0   -4.0d0    4.0d0
CN    0.04d0    0.0d0    4.0d0
DCN   0.04d0   -2.0d0    2.0d0
XYZ   0.08d0    2.0d0   10.0d0
DXYZ  0.08d0    2.0d0   10.0d0
```

The values are indexed by CV type. In TAMD and LogMFD the fictitious CV mass is derived from $\Delta\xi_d^{\mathrm{ref}}$ as described in the AFED theory section.

11.11 Additional keywords in DESCENT, ASCENT and AUTO

11.11.1 <algo_ascent_afed>

Used in the cases of <method> = AFED, with <afed_type> = ASCENT. This keyword sets the numerical algorithm for ascent search in adiabatic free energy dynamics. The data should be provided in the first line (character).

- GAD: Gentlest ascent dynamics.
- EVF: Eigenvector following.

The default value is “GAD”.

11.11.2 <ascent_sampling_afed>

Used in the cases of <method> = AFED, with <afed_type> = ASCENT. This keyword sets the sampling method of the Hessian term for the ascent search in adiabatic free energy dynamics. The data should be provided in the first line (character).

- ANALYTICAL: Analytical evaluation of the Hessian term.
- NUMERICAL: Numerical evaluation of the Hessian term using finite difference method.

The default value is “NUMERICAL”. For numerical evaluation, the finite difference value can be controlled by the keyword <fdiff_sampling_afed>.

11.11.3 <dt_ascent_afed>

Used in the cases of <method> = AFED with <afed_type> = ASCENT, AUTO. This keyword sets the initial step size for ascent search in adiabatic free energy dynamics. The data should be provided in the first line (real number). The default value is “1.0” femtoseconds.

11.11.4 <dt_conv_afed>

Used in the cases of <method> = AFED, with <afed_type> = DESCENT, ASCENT, AUTO. This keyword sets the AFED step size at convergence in adiabatic free energy dynamics. The data should be provided in the first line (real number). The default value is “0.05” (dimensionless).

11.11.5 <dt_damp_afed>

Used in the cases of <method> = AFED, with <afed_type> = DESCENT, ASCENT, AUTO. This keyword sets the damping factor, λ^{afed}, for the AFED step size when approaching a stationary point in adiabatic free energy dynamics. The data should be provided in the first line (real number). The default value is “0.7” (dimensionless).

11.11.6 <dt_descent_afed>

Used in the cases of <method> = AFED with <afed_type> = DESCENT, AUTO. This keyword sets the initial step size for descent search in adiabatic free energy dynamics. The data should be provided in the first line (real number). The default value is “0.5” femtoseconds.

11.11.7 <fdiff_sampling_afed>

Used in the cases of <method> = AFED, with <afed_type> = ASCENT and <ascent_sampling_afed> = NUMERICAL. This keyword sets the finite difference value used to numerically sample the Hessian term for the ascent search in adiabatic free energy dynamics. The data should be provided in the first line (real number). The default value is “5.0”.

11.11.8 <gamma_ascent_afed>

Used in the cases of <method> = AFED, with <afed_type> = ASCENT, AUTO. This keyword sets the parameter γ^{gad} in the ascent search of adiabatic free energy dynamics, in hartree. The data should be provided in the first line (real number). The default value is “1.0” hartree.

11.11.9 <niter_gad2evf_afed>

Used in the cases of <method> = AFED with <afed_type> = ASCENT, AUTO and <algo_ascent_afed> = GAD. This keyword sets the iteration step to switch the algorithm from GAD to EVF when approaching the saddle point. The data should be provided in the first line (integer). The default value is “80”.

11.11.10 <niter_refresh_afed>

Used in the cases of <method> = AFED with <afed_type> = ASCENT, AUTO and <algo_ascent_afed> = GAD, EVF. This keyword sets the step interval for numerical Hessian calculations, refreshing the guiding vector to the lowest eigenvector of the Hessian. The data should be provided in the first line (integer). The default value is “20”.

11.11.11 <nmiss_auto_afed>

Used in the cases of <method> = AFED, with <afed_type> = AUTO. This keyword sets the number of consecutive missed shots from each minimum, N_{miss}. Exceeding this number, a new shooting starts from another minimum, which is listed in the file auto.ini with the label “EQ”. The data should be provided in the first line (integer). The default value is “5”.

11.11.12 <ioption_eigen_afed>

Used in the cases of <method> = AFED with <afed_type> = DESCENT, ASCENT, AUTO. This keyword sets the option for eigenvalue calculations at the free energy stationary points found in adiabatic free energy dynamics. The data should be provided in the first line (integer).

- 1: Eigenvalues are calculated.
- 0: Eigenvalues are not calculated.

The default value is “1”.

11.11.13 <nshot_auto_afed>

Used in the cases of <method> = AFED, with <afed_type> = AUTO. This keyword sets the maximum number of shots from each minimum, N_{shot}. Exceeding this number, the new shooting starts from another minimum, which is listed in the file auto.ini with the label “EQ”. The data should be provided in the first line (integer). The default value is “10”.

11.11.14 <radius_auto_afed>

Used in the cases of <method> = AFED, with <afed_type> = AUTO. This keyword sets the reference radius in the automated search of adiabatic free energy dynamics. The data should be provided in the first line (real number). The default value is “5.0”.

11.11.15 <iprint_xyz_afed>

Used in the cases of <method> = AFED, with <afed_type> = DESCENT, ASCENT, AUTO, TAMD, LOGMFD. This keyword sets the print interval of molecular snapshots written to the file afed.xyz. A snapshot is taken at the final step of the MD run per AFED step in adiabatic free energy dynamics, temperature accelerated molecular dynamics, and logarithmic mean force dynamics. The data should be provided in the first line (integer). The default value is “1” step (every AFED step). Set to “-1” to disable printing.

11.11.16 <iprint_test_afed>

Used in the cases of <method> = AFED, with <afed_type> = DESCENT, ASCENT, AUTO, TAMD, LOGMFD. This keyword sets the print interval of the mean force for the convergence test. The data should be provided in the first line (integer). The default value is “100” steps.

11.12 Additional keywords in TAMD and LogMFD

11.12.1 <alpha_logmfd>

Used in the case of <method> = AFED and <afed_type> = LOGMFD. This keyword sets the parameter $\alpha_{\log}(=\alpha^{'}/k_BT > 0)$ for logarithmic mean force dynamics ($\alpha^{'}$ is specified by this keyword). The data should be provided in the first line (dimensionless real number). Note that T is the physical temperature given by the keyword <temperature>.

The canonical 2.8.0 fallback from input_default.dat is

```
<alpha_logmfd>
4.d0
```

When the keyword is omitted from input.dat, this fallback is used by the current parser.

11.12.2 <dt_logmfd>

Used in the case of <method> = AFED with <afed_type> = LOGMFD. This keyword sets the step size in logarithmic mean force dynamics, Δt_{cv}. The data should be provided in the first line (real number). The default value is “1.0”.

11.12.3 <dt_tamd>

Used in the case of <method> = AFED with <afed_type> = TAMD. This keyword sets the step size in temperature accelerated molecular dynamics, Δt_{cv}. The data should be provided in the first line (real number). The default value is “1.0”.

11.12.4 <fenergy_ini_logmfd>

Used in the case of <method> = AFED, with <afed_type> = LOGMFD. This keyword sets the initial value of the free energy estimated on-the-fly during the LogMFD run, which defines the origin of the energy scale. The default value is “0.005” hartree.

11.12.5 <fenergy_ini_tamd>

Used in the case of <method> = AFED, with <afed_type> = TAMD. This keyword sets the initial value of the free energy estimated on-the-fly during the TAMD run, which defines the origin of the energy scale. The default value is “0.0” hartree.

11.12.6 <gamma_logmfd>

Used in the case of <method> = AFED and <afed_type> = LOGMFD. This keyword sets the parameter $\gamma_{\log}(=\gamma^{'}k_BT)$ in logarithmic mean force dynamics ($\gamma^{'}$ is specified by this keyword). The data should be provided in the first line (dimensionless real number). The default setting is “$\gamma^{'} = 1/\alpha^{'}$”, i.e., “$\gamma_{\log} = 1/\alpha_{\log}$”. Note that T is the physical temperature given by the keyword <temperature>.

11.12.7 <ioption_afed>

Used in the cases of <method> = AFED, with <afed_type> = TAMD, LOGMFD. This keyword sets the weight W_l in parallel-dynamics discussed in Sec.13.16. With "1", $W_l = 1$ for all the beads (replicas), while W_l is set by Eq.(204) (see Sec.13.16) with "2". The default value is "1" ($W_l = 1$ for all the beads).

11.12.8 <logmfd_type>

Used in the case of <method> = AFED and <afed_type> = LOGMFD. This keyword sets the type of equation of motion for logarithmic mean force dynamics. The data should be provided in the first line. Choose one of the following:

- <logmfd_type> = NVE: no thermostat is used for fictitious particles.
- <logmfd_type> = NVT: single Nosé-Hoover thermostat is attached to fictitious particles.
- <logmfd_type> = VS: velocity scaling (Gaussian thermostat) applied to fictitious particles.

The default value is "NVE".

11.12.9 <mass_afed>

Used in the case of <method> = AFED with <afed_type> = TAMD, LOGMFD. This keyword sets the parameters that re-adjust the mass of the fictitious CV particles. If "AUTOMATIC" is set, the mass is given by default using the keyword <params_afed> (see Sec.11.10.7). If "MANUAL" is set, the mass given by <params_afed> is multiplied by scaling factors. The scale factor of the mass of the ith particle should be provided in the ith line below "MANUAL". The mass values used in the LogMFD/TAMD run can be checked in the output file logmfd.out.

11.12.10 <niter_bath_afed>

Used in the case of <method> = AFED with <afed_type> = TAMD, LOGMFD and <tamd_type> = NVT or <logmfd_type> = NVT. This keyword sets the number of AFED steps n_b, which determines the time scale, $\tau_{\rm cv} = n_b \Delta t_{\rm cv}$, of the Nosé-Hoover thermostat attached to fictitious particles in TAMD and LogMFD in the NVT ensemble. This sets the mass of the Nosé-Hoover thermostat, Q_η, from the relation $Q_\eta = D k_B T \tau_{\rm cv}^2$, where D is the number of fictitious particles. The data should be provided in the first line (real number). The default value is "100" (dimensionless). Note that n_b should be appropriately adjusted according to the value of $\Delta\xi_d^{\rm ref}$; n_b should be increased when $\Delta\xi_d^{\rm ref}$ is decreased, and vice versa.

11.12.11 <tamd_type>

Used in the case of <method> = AFED and <afed_type> = TAMD. This keyword sets the type of equation of motion for temperature accelerated molecular dynamics. The data should be provided in the first line. Choose one of the following:

- <tamd_type> = NVE: no thermostat is used for fictitious particles.
- <tamd_type> = NVT: single Nosé-Hoover thermostat attached to fictitious particles.
- <tamd_type> = VS: velocity scaling (Gaussian thermostat) applied to fictitious particles.

The default value is "NVT".

11.12.12 <temperature_tamd>

Used in the case of <method> = AFED with <afed_type> = TAMD. This keyword sets the temperature $T_{\rm tamd}$ for temperature accelerated molecular dynamics. The data should be provided in the first line (real number) in kelvin. The default value is "3000.0" kelvin.

11.12.13 <start_afed>

Used in the case of <method> = AFED with <afed_type> = TAMD, LOGMFD. This keyword sets the initial position and velocity of each fictitious CV particle (variable). If "MANUAL" is set in the first line, the initial position and velocity of each CV particle should be provided for each line (see Sec.5.4.6). If "AUTOMATIC" is set instead (default), the initial positions are automatically estimated from the initial configuration of the MD system, and the initial velocities are set randomly corresponding to the physical temperature given by the keyword <temperature>.

11.12.14 <start_therm_afed>

Used in the case of <method> = AFED with <afed_type> = TAMD, LOGMFD. This keyword sets the initial position and velocity of the Nosé-Hoover thermostat variable η. If "MANUAL" is specified, its initial position and velocity should be provided (default setting is "MANUAL" with $\eta(0) = 0$ and $v_\eta(0) = 0$). If "AUTOMATIC" is specified instead, η is set to "0" and v_η is automatically set according to the preset physical temperature.

11.13 Source-audited keywords added in version 2.8.0

The entries in this subsection document active tags that are read by the PIMD 2.8.0 source code but were not described explicitly in the previous manual. The file from which each tag is read is stated because not every tag belongs to `input.dat`. Defaults quoted below are the defaults implemented in the source code; where no fallback exists, the keyword is described as required for the corresponding feature.

11.13.1 <afir>

Read from `input.dat` for the artificial-force induced reaction (AFIR) external term. The first line is the number of atom pairs. Each following line contains the two atom indices and the force parameter for that pair.

11.13.2 <ax>

Read from `input.dat` in atom-exchange/replica-exchange setup. The line contains the first and last atom indices eligible for atom exchange. If the tag is unavailable or either bound is non-positive, the corresponding bound is replaced by 1 or `natom`, respectively.

11.13.3 <nsubstep_ax>

Read from `input.dat` when atom exchange is combined with a `DUAL` potential. It gives the number of inner low-level steps used by the dual-level atom-exchange procedure. It is required for this setup.

11.13.4 <bond>

Read from `input.dat` for harmonic bond restraints in the external-force module. The first line is the number of restrained bonds. Each subsequent line contains atom 1, atom 2, the equilibrium distance, and the force constant.

11.13.5 <group>

Read from `input.dat` for grouped mechanical external forces. The first line gives the number of groups. Each group line contains a label, the number of atoms in the group, three Cartesian force coefficients, and a potential flag. The label is the name of a separate file containing the atom indices, one index per line; the indices are not written directly after the <group> block.

11.13.6 <esf>

Read from `input.dat` for a uniform external electrostatic field. The first line gives the three Cartesian field components. It is followed by `natom` lines containing an atom index and the charge assigned to that atom. The field contribution is then evaluated from these charges.

11.13.7 <dvr>

Read from `input.dat` by the one-dimensional DVR method. The line gives the lower and upper limits of the DVR coordinate. The number of DVR grid points is `nbead` and the code constructs the interior mesh from these limits. The present DVR setup requires two atoms.

11.13.8 <quartic>

Read from `input.dat` by the one-dimensional quartic model potential. Five coefficients are supplied, defining

$$V(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4.$$

11.13.9 <mbead_matsubara>

Read from `input.dat` for `MATSUBARA-1D`. It specifies the number of Matsubara modes retained in the dynamics. No canonical fallback is provided, so it must be set for this method.

11.13.10 <iprint_matsubara>

Read from `input.dat` for `MATSUBARA-1D`. It gives the output interval for Matsubara-specific data. A non-positive value disables this output. No canonical default is defined in `input_default.dat`.

11.13.11 <ntass>

Read from `input.dat` by temperature-accelerated sliced sampling (TASS). It gives the number of collective variables treated by TASS and is required by the TASS setup. Each following CV line begins with the sampling assignment (`AFED`, `CONS`, or `META`) followed by the CV definition. In new TOML these two concepts are separated as `[tass.cv.N].sampling` and `[tass.cv.N].type`.

11.13.12 <ntrial_tps>

Read from `input.dat` in transition-path sampling. It is the number of trial trajectories generated in the TPS cycle and is required for that method.

11.13.13 <temp_kick_tps>

Read from `input.dat` in transition-path sampling. It sets the temperature scale used for the random velocity perturbation (the kick) when a trial path is generated. The velocity-kick amplitude is scaled by the square root of the ratio of this temperature to the simulation temperature.

11.13.14 <qe7_command>

Read from `input.dat` by the system-call Quantum ESPRESSO 7 interface. It is the command used to launch the QE7 calculation. The code also searches `input_default.dat` if it is absent from `input.dat`, but the canonical 2.8.0 default file contains no value; therefore it must be supplied when this interface is used.

11.13.15 <cutoff_helium>

Read from `input.dat` by the helium pair-potential implementation. It sets the interaction cutoff. A negative value requests automatic cutoff selection: the periodic case uses the half-cell criterion, whereas the isolated

case is effectively unbounded by the simulation cell. The canonical 2.8.0 fallback is “-1.0”, which is the default when the keyword is omitted and therefore selects this automatic-cutoff behavior.

11.13.16 <ncart>

Read from `cart.dat`. It gives the number of Cartesian restraint or polynomial terms. Each following line contains the atom index, the powers in x, y, and z, the three reference coordinates, and the force coefficient. If <ncart> is absent, the number of terms is zero.

11.13.17 <energy>

Read from `hob.dat` for the HOB potential. It gives the reference energy associated with the HOB reference geometry. This is an auxiliary-file record, not an `input.dat` keyword. The literal marker `<energy>` also occurs in MOLPRO table output; there it is an output marker and likewise is not a PIMD input keyword.

11.13.18 <lengthunit>

Read from `hob.dat` for the harmonic-plus-one-dimensional-barrier (HOB) potential. It specifies the length unit of the reference geometry. When the value is `ANGSTROM`, the stored coordinates are converted to bohr.

11.13.19 <coordinates>

Read from `hob.dat`. It introduces the reference geometry about which the harmonic expansion is defined. The block contains one species label and three Cartesian coordinates for each of the `natom` atoms.

11.13.20 <hessian>

Read from `hob.dat`. It supplies the Hessian associated with the HOB reference geometry. The source reads the matrix entries, symmetrizes the matrix, and uses the masses when constructing the normal modes.

11.13.21 <imode>

Read from `hob.dat`. It selects the normal mode whose harmonic potential is replaced by a tabulated one-dimensional potential. The <imode> block also contains the number of tabulated grid points and the coordinate–potential pairs used to construct the spline interpolation.

11.13.22 <dipoles>

Read from the CFF/LAMMPS input for molecular dipole analysis with the CFF potential. The first value is the number of molecular groups. The following information defines the number of atoms in each group and their atom indices. If this tag is unavailable, the periodic CFF dipole path falls back to the atomic form.

11.13.23 <ljrule>

Read from the CFF/LAMMPS input by the atom-based Lennard–Jones implementation. It specifies the combination rules used to construct unlike-pair epsilon and sigma parameters. The source default is `DEFAULT DEFAULT`.

11.13.24 <ljcutoff>

Read from the CFF/LAMMPS input. It gives the inner and outer switching/cutoff distances for the atom-based Lennard–Jones interaction. The source fallback values are 20 and 25, respectively, in the internal convention used by this routine.

11.13.25 <ljpair>

Read from the CFF/LAMMPS input. The first value is the number of atom-specific Lennard–Jones parameter records. Each following line contains an atom index, epsilon, and sigma. This tag does *not* select the cross-species combination rule; that is the role of <ljrule>.

11.13.26 <ljbond>

Read from the CFF/LAMMPS input for bonded-pair Lennard–Jones records used by the atom-based CFF implementation. If the tag is absent, the number of such records is zero.

11.13.27 <skin_ljpair>

Read from the CFF/LAMMPS input. It sets the neighbor-list skin added to the Lennard–Jones cutoff. The source fallback is 4.0. The current implementation does not apply the commented-out Angstrom-to-bohr conversion at this point, so the value should be kept consistent with the existing CFF internal-input convention.

11.13.28 <skin_ewald>

Read from the CFF/LAMMPS input. It sets the neighbor-list skin used for the real- space Ewald list. The source fallback is 4.0. As for <skin_ljpair>, the current source does not execute the commented-out Angstrom-to-bohr conversion at this point.

11.13.29 <pme_mesh_ewald>

Read from the CFF/LAMMPS input for particle-mesh Ewald. It specifies the target mesh density in grid points per bohr. The source fallback is 1.09; the mesh size in each cell direction is then rounded to a power of two.

11.13.30 EAM/ADP/PAIR neighbor skin

The many-body and tabulated-pair interfaces take their neighbor-list skin from the LAMMPS `neighbor SKIN ...` command. If omitted, the source fallback is 2.0 Å, converted internally to bohr.

11.13.31 Tersoff neighbor skin

For TERSOFF, the neighbor-list skin is read from the LAMMPS `neighbor SKIN ...` command in `input.lammps`. If omitted, the source uses 2.0 Å, converted internally to bohr.

11.13.32 <oh_cutoff_oss>

Read from `oss.dat`. It gives the inner and outer O–H cutoff distances used by the OSS water potential. If unavailable, the source uses 50 and 53.

11.13.33 <oo_cutoff_oss>

Read from `oss.dat`. It gives the inner and outer O–O cutoff distances used by the OSS water potential. If unavailable, the source uses 50 and 53.

11.13.34 <3b_cutoff_oss>

Read from `oss.dat`. It sets the cutoff for the three-body OSS term. The source fallback is 5.4.

11.13.35 <ioption_oss>

Read from `oss.dat`. It controls the augmented short-range OSS branch. The source fallback is 1. Setting it to 0 disables the additional short-range O–O contribution selected by this option.

11.13.36 <ewald_oss>

Read from `oss.dat`. Two numbers set the Ewald accuracy and the target balance between real- and reciprocal-space work for the OSS electrostatics. The source fallback values are 10^{-8} and 4.

11.13.37 <ewpol_oss>

Read from `oss.dat`. Two numbers set the Ewald accuracy and real/ reciprocal-space balance for polarization terms. The source fallback values are 10^{-10} and 0.1.

11.13.38 <nacme>

The literal marker `<nacme>` is read from the intermediate MOLPRO table file `table.molpro` by the nonadiabatic interface. It labels a nonadiabatic-coupling table (including the state pair in the title); it is not a PIMD `input.dat` keyword and has no default. The corresponding MOLPRO input template is responsible for writing this marker.

12 Files

The files can be categorized as follows.

- Input files: named `*.dat`.
- Output files: most are named `*.out`. The exceptions are the trajectory files in xyz format, `trj.xyz` and `calc.xyz`.
- Restart files: named `*.ini`. These files are created at the end of each run and include the data required to restart the next run. In principle, these files do not need to be modified by the user.

12.1 Input files

12.1.1 input.dat

Used in ALL calculations. This is the engine-level input file read by the Fortran code. Keywords are represented by brackets, `<...>`; the line or lines following each keyword contain its data, and all remaining text is treated as comments. Keyword blocks may appear in arbitrary order.

In the curated PIMD 2.8.0 workflow, users normally edit `pimd.toml` and **`pimd-prep`** generates `input.dat`; see Section 2. Therefore a generated `input.dat` should normally not be edited independently. Existing hand-written `input.dat` files remain fully supported.

The file does not need to contain every keyword. Keywords absent from `input.dat` are complemented by values in `input_default.dat`. The complete engine-level keyword semantics are listed in Section 11.

Defaults apply only when a keyword is absent. If a keyword is present but its data cannot be read with the requested type, the reader reports an error. Scalar and vector real inputs must be finite. A required default that is absent or invalid is also an error. These errors return a nonzero process status.

12.1.2 input_default.dat

Used in ALL calculations. This file contains the default values of the keywords. Keywords that are not defined in `input_default.dat` must be explicitly specified in `input.dat`.

The distributed curated examples use the single canonical `examples/input_default.dat`; launchers link it for the duration of the run. It should normally be kept unchanged. It also serves as the engine-level reference for defaults and for validation of legacy tags by **pimd-prep**.

12.1.3 centroid.dat

NOTE: This file is used only for **<input_style>** = OLD. For **<input_style>** = NEW, `structure.dat` is used instead.

Used in ALL calculations for an initial start, except for **<method>** = STRING. This file contains a list of Cartesian coordinates of atoms in atomic units (bohr), given as:

```
x  y  z  (of atom 1)
x  y  z  (of atom 2)
...
```

The number of lines must be equal to the number of atoms.

12.1.4 LAMMPS many-body/table files

For **<ipotential>** = EAM, **<ipotential>** = ADP, and **<ipotential>** = TERSOFF, the active potential selection is made from `input.lammps`. The same input mechanism is used for **<ipotential>** = PAIR and for the metal part of **<ipotential>** = METALWATER. The referenced potential is a LAMMPS setfl/extended-setfl file, a native LAMMPS Tersoff parameter file, or a LAMMPS pair-table file as appropriate. A `data.lammps` file may be present so the same input can be validated with LAMMPS; PIMD coordinates, masses, and cell remain controlled by the PIMD structure/input path. See Sections 11.7 and 11.8.

12.1.5 gamess.dat

Used for the GAMESS potential, when **<ipotential>** = GAMESS. This file must be a GAMESS input file. See Section 6.3.6.

12.1.6 g03.dat

Used for the GAUSSIAN 03 potential, when **<ipotential>** = G03. This file must be a G03 input file. See Section 6.3.7.

12.1.7 g98.dat

Used for the GAUSSIAN 98 potential, when **<ipotential>** = G98. This file must be a G98 input file. See Section 6.3.7.

12.1.8 CFF files

For **<ipotential>** = CFF, the classical force field is read from `input.lammps`, the data/topology file selected by its `read_data` command (default `data.lammps`), and usually one or more explicitly included coefficient files such as `parm.lammps`. The command file selects styles and dependencies; the selected data file contains atom, topology, mass, charge, and simulation-cell data. A separate `cmap.dat` is used when a CMAP grid is required. The legacy `mm.dat` file is not read. See Section 10.

12.1.9 molpro.dat

Used for the MOLPRO potential, when **<ipotential>** = MOLPRO. This file must be a MOLPRO input file. See Section 6.3.8. See also Section 5.9.1 for performing nonadiabatic dynamics calculations, i.e., surface hopping dynamics and mean-field dynamics, using MOLPRO.

12.1.10 mopac.dat

Used for the MOPAC potential, when <ipotential> = MOPAC. This file must be a MOPAC input file. See Section 6.3.9.

12.1.11 orca.dat

Used for the ORCA potential, when <ipotential> = ORCA. This file must be an ORCA input file. See Section 6.3.11.

12.1.12 structure.dat

Used in ALL calculations for an initial start. This file contains a list of Cartesian coordinates of atoms in XYZ format, given as:

```
N
unit
symbol  x  y  z
symbol  x  y  z
......  .......
```

In the first line, N is the number of atoms (integer). In the second line, unit is an option (character), either BOHR or ANGSTROM. In lines 3–$(N+2)$, symbol denotes the atomic species, and x, y, and z are the Cartesian coordinates of the corresponding atoms.

In the case of <method> = STRING, OMOPT, the list of Cartesian coordinates is duplicated, corresponding to the reactant and the product. An initial guess is generated as a linear path connecting the reactant and the product.

In general, this file must be prepared to start a new calculation. However, when the restart file geometry.ini is present, this file is no longer required. The PIMD code always searches for the restart file geometry.ini before looking for centroid.dat.

NOTE: This file is used only for <input_style> = NEW. For <input_style> = OLD, centroid.dat is used instead.

12.1.13 turbo.dat

Used for TURBOMOLE, when <ipotential> = TURBOMOLE. This file contains the set of TURBOMOLE execution commands used to perform energy and force calculations, along with the corresponding TURBOMOLE control files. If this file is not found in the execution directory, the code alternatively searches for the required information in input.dat, or otherwise in input_default.dat, following the keyword <turbo_command>. The following lines should be specified:

- Hartree–Fock calculation:

```
     dscf   control.1
     grad   control.2
```

- MP2 calculation:

```
     dscf   control.1
   mpgrad   control.2
```

- DFT calculation:

```
     dscf   control.1
     grad   control.2
```

- RI-MP2 calculation:

```
     dscf    control.1
    rimp2    control.2
```

- RI-DFT calculation:

```
    ridft    control.1
   rdgrad    control.2
```

- RI-CC2 calculation:

```
     dscf    control.1
    ricc2    control.2
```

12.1.14 TAMD

(For the initial run) The following files are required:
`input.dat`, `input_default.dat`, `centroid.dat`
(If **<input_style>** = `NEW`, `structure.dat` should be prepared instead of `centroid.dat`.)

(For restart) The following files are required:
`input.dat`, `input_default.dat`, `centroid.dat`, `geometry.ini`, `bath.ini`, `step.ini`, `afed.ini`
(If **<input_style>** = `NEW`, `structure.dat` should be prepared instead of `centroid.dat`.)

Optional files: `average.ini`

Note that all `*.ini` files are overwritten at the end of the run.

12.1.15 LogMFD

(For the initial run) The following files are required:
`input.dat`, `input_default.dat`, `centroid.dat`
(If **<input_style>** = `NEW`, `structure.dat` should be prepared instead of `centroid.dat`.)

(For restart) The following files are required:
`input.dat`, `input_default.dat`, `centroid.dat`, `geometry.ini`, `bath.ini`, `step.ini`, `afed.ini`
(If **<input_style>** = `NEW`, `structure.dat` should be prepared instead of `centroid.dat`.)

Optional files: `average.ini`

Note that all `*.ini` files are overwritten at the end of the run.

12.2 Output files

12.2.1 afed.out

Used when **<method>** = `AFED`. This file contains the trajectory of the collective variables. In n-dimensional dynamics, each of the n printed lines corresponds to one collective variable. Each line contains the following columns:

- **<afed_type>** = `GRAD, TEST`:
  - **column 1** (`step`): AFED step number.
  - **column 2** (`st`): AFED status.
    - `GR`: mean-force calculation.

    - `TE`: convergence-test calculation.
  - **column 3** (`cv`): collective-variable index, d.
  - **column 4** (`r-ideal`): reference position of the fictitious particle, ξ_d^{ref}.
  - **column 5** (`r-mean`): mean position of the collective variable, ξ_d.
  - **column 6** (`f-energy`): free energy in hartree.
  - **column 7** (`-df/dr`): mean force on the fictitious particle, $-\partial A_{\mathrm{cv}}/\partial \xi_d^{\mathrm{ref}}$.
  - **column 8** (`dr/dt`): velocity of the fictitious particle, $d\xi_d^{\mathrm{ref}}/dt$.
  - **column 9** (`dt`): AFED step size, Δt_{cv}.
- <afed_type> = ASCENT, DESCENT, AUTO:
  - **column 1** (`step`): AFED step number.
  - **column 2** (`st`): AFED status.
    - `DE`: descent trajectory to a free-energy minimum.
    - `D1`: descent trajectory started in the $+\mathbf{n}$ direction from a free-energy saddle point to a free-energy minimum.
    - `D2`: descent trajectory started in the $-\mathbf{n}$ direction from a free-energy saddle point to a free-energy minimum.
    - `AS`: ascent trajectory to search for a free-energy saddle point.
    - `EQ`: free-energy minimum point.
    - `TS`: free-energy saddle point.
    - `HI`: terminated because the free energy exceeds the limit defined by <fenergy_max_afed>.
    - `OB`: terminated because the CV is out of the bounds defined by <params_afed>.
  - **column 3** (`cv`): collective-variable index, d.
  - **column 4** (`r-ideal`): reference position of the fictitious particle, ξ_d^{ref}.
  - **column 5** (`r-mean`): mean position of the collective variable, ξ_d.
  - **column 6** (`f-energy`): free energy in hartree.
  - **column 7** (`-df/dr`): mean force on the fictitious particle, $-\partial A_{\mathrm{cv}}/\partial \xi_d^{\mathrm{ref}}$.
  - **column 8** (`dr/dt`): velocity of the fictitious particle, $d\xi_d^{\mathrm{ref}}/dt$.
  - **column 9** (`dt`): AFED step size, Δt_{cv}.
  - **column 10** (`n`): unit vector, $\mathbf{n}$.
    - In an ascent trajectory, $\mathbf{n}$ is the guiding vector.
    - In a descent trajectory, $\mathbf{n}$ is parallel to the mean force $-\partial A_{\mathrm{cv}}/\partial q_d^{\mathrm{ref}}$.
  - **column 11** (`xi`): angle ξ.
- <afed_type> = TAMD, LOGMFD:
  - **column 1** (`step`): AFED step number.
  - **column 2** (`st`): AFED status.
    - `TA`: TAMD trajectory.
    - `LO`: LogMFD trajectory.
    - `HI`: terminated because the free energy exceeds the limit defined by <fenergy_max_afed>.
    - `OB`: terminated because the CV is out of the bounds defined by <params_afed>.
  - **column 3** (`cv`): collective-variable index, d.
  - **column 4** (`r-ideal`): reference position of the fictitious particle, ξ_d^{ref}.
  - **column 5** (`r-mean`): mean position of the collective variable, ξ_d.
  - **column 6** (`f-energy`): free energy in hartree.
  - **column 7** (`-df/dr`): mean force on the fictitious particle, $-\partial A_{\mathrm{cv}}/\partial \xi_d^{\mathrm{ref}}$.
  - **column 8** (`dr/dt`): velocity of the fictitious particle, $d\xi_d^{\mathrm{ref}}/dt$.
  - **column 9** (`h-energy`): total Hamiltonian, H^{tamd} or H^{log}.
  - **column 10** (`temp`): instantaneous temperature of the fictitious particle.

The file is appended when a run is restarted.

12.2.2 afed.xyz

This file contains molecular snapshots at the end of each AFED iteration in xyz format. For multiple beads, the data are printed <nbead> times. The file is appended when a run is restarted.

12.2.3 afed_weight.out

Used when <method> = AFED with <afed_type> = LOGMFD. The weight W_l in parallel dynamics (discussed in Sec. 13.16) is written at every AFED step to afed_weight.out. The file is appended when a run is restarted.

12.2.4 alc.out

Used when <method> = PIHMC with <irem_type> = HX. This file contains the alchemical analysis. The print interval can be controlled by the keyword <iprint_alc>.

- For <method> = PIHMC with <irem_type> = HX:
  - **column 1**: step number.
  - **column 2**: free-energy difference.
  - **column 3**: potential-energy difference.
  - **column 4**: entropy difference.
  - **column 5**: cumulative average of the free-energy difference.
  - **column 6**: cumulative average of the potential-energy difference.
  - **column 7**: cumulative average of the entropy difference.

12.2.5 best.out

Used when <ipotential> = QMMM and <ioption_best> = 1. This file contains data from the BEST method. The print interval can be controlled by the keyword <iprint_best>. Each line contains the following five columns:

- **column 1**: step number.
- **column 2**: number of boundary atoms.
- **column 3**: number of boundary atoms in the QM region.
- **column 4**: number of boundary atoms in the MM region.
- **column 5**: bias potential in BEST.

For <nbead> = n, the data are repeated n times.

12.2.6 bond.out

Used when <method> = MD, PIMD, PIHMC, REHMC. This file contains bond-length data. The print interval can be controlled by the keyword <iprint_bond>.

- For <method> = MD, PIMD and <method> = PIHMC with `iorder_hmc = 2`:
  - **column 1**: step number.
  - **columns 2–3**: atomic-species pair defined in structure.dat.
  - **columns 4–6**: bond lengths associated with the Kubo, centroid, and bead correlation functions, respectively.
  - **columns 7–9**: mean-square bond lengths associated with the Kubo, centroid, and bead correlation functions, respectively.
  - **columns 10–12**: cumulative averages of the bond lengths associated with the Kubo, centroid, and bead correlation functions, respectively.
  - **columns 13–15**: cumulative averages of the mean-square bond lengths associated with the Kubo, centroid, and bead correlation functions, respectively.

- For <method> = PIHMC with iorder_hmc = 4:
  - **column 1**: step number.
  - **columns 2–3**: atomic-species pair defined in structure.dat.
  - **column 4**: bond length.
  - **column 5**: second-order contribution to the bond length.
  - **column 6**: fourth-order contribution to the bond length.
  - **column 7**: cumulative average of the bond length.
  - **column 8**: cumulative average of the second-order contribution.
  - **column 9**: cumulative average of the fourth-order contribution.
- For <method> = REHMC:
  - **column 1**: step number.
  - **column 2**: bead number.
  - **columns 3–4**: atomic-species pair defined in structure.dat.
  - **column 5**: bond length.
  - **column 6**: mean-square bond length.
  - **column 7**: cumulative average of the bond length.
  - **column 8**: cumulative average of the mean-square bond length.

NOTE: This analysis is not recommended when the number of atomic species is large, since the computational cost grows quadratically.

12.2.7 box.dcd

Used when periodic boundary conditions are applied.

This binary file contains the periodic boundary box and the virial of the system. The print interval can be controlled by the keyword <iprint_dcd>.

The file contains a 276-byte header similar to that of trj.dcd. The important parts are as follows:

- **byte 1–4**: (int) 84, indicating the endianness of the file.
- **byte 17–20**: (int) output interval of the box information.
- **byte 269–272**: (int) number of beads.

After the header, the box data are organized as follows:

- **byte 1–4**: (int) size of the next read (144).
- **byte 5–76**: (doubles [8 bytes]) box information ordered as A(1), B(1), C(1), A(2), ...
- **byte 77–148**: (doubles [8 bytes]) virial ordered in the same way.

These data blocks are repeated for each bead when multiple beads are used.

12.2.8 box.out

Used when <ensemble> = NPH, NPT, NTH, NTT. This file contains the trajectory of the box matrix. The print interval can be controlled by the keyword <iprint_box>.

- For <method> = NPH, NPT:
  - **line 1, column 1**: step number.
  - **line 1, column 2**: volume.
  - **line 1, column 3**: volume average.
  - **lines 2–4, columns 1–3**: box matrix.
  - **lines 2–4, columns 4–6**: average box matrix.
  - **line 5, column 1**: step number.
  - **line 5, column 2**: isotropic pressure.

  - **line 5, column 3**: average isotropic pressure.
  - **lines 6–8, columns 1–3**: pressure matrix.
  - **lines 6–8, columns 4–6**: average pressure matrix.
- For <method> = NTH, NTT:
  - **line 1, column 1**: step number.
  - **line 1, column 2**: volume.
  - **line 1, column 3**: volume average.
  - **lines 2–4, columns 1–3**: box matrix.
  - **lines 2–4, columns 4–6**: average box matrix.
  - **line 5, column 1**: step number.
  - **line 5, column 2**: isotropic pressure.
  - **line 5, column 3**: average isotropic pressure.
  - **lines 6–8, columns 1–3**: pressure matrix.
  - **lines 6–8, columns 4–6**: average pressure matrix.
  - **line 9, column 1**: step number.
  - **line 9, column 2**: isotropic stress.
  - **line 9, column 3**: average isotropic stress.
  - **lines 10–12, columns 1–3**: stress matrix.
  - **lines 10–12, columns 4–6**: average stress matrix.
  - **line 13, column 1**: step number.
  - **line 13, column 2**: isotropic strain.
  - **line 13, column 3**: average isotropic strain.
  - **lines 14–16, columns 1–3**: strain matrix.
  - **lines 14–16, columns 4–6**: average strain matrix.

12.2.9 calc.xyz

Used by the analysis code calc.x. This file contains the trajectory in xyz format. The print interval can be controlled by the keyword <jprint_xyz> in calc.dat. For <nbead> = n, the data are repeated n times. The output is controlled by the keywords <jformat_xyz> and <jorigin_xyz> in calc.dat.

12.2.10 charges.dcd

This binary file is currently only output for CP2K calculations. It contains Mulliken and Hirshfeld charges of each atom for the frames printed by PIMD, according to the <iprint_dcd> keyword.

The file contains a 276-byte header similar to that of trj.dcd. The important parts are as follows:

- **byte 1–4**: (int) 84, indicating the endianness of the file.
- **byte 17–20**: (int) output interval of the box information.
- **byte 269–272**: (int) number of beads.

After the header, the data are organized as follows, where **N** is four times the number of atoms:

- **byte 1–4**: (int) size of the next read (8 × numatoms × numbeads).
- **next N bytes**: (floats [4 bytes]) Mulliken charges.
- **next N bytes**: (floats [4 bytes]) Hirshfeld charges.

The two data blocks are repeated for each bead when multiple beads are used.

12.2.11 cons.out

Used when <method> = MD with <cons> = 1. This file contains constraint information. The print interval can be controlled by the keyword <iprint_cons>. When n constraints are applied, each of the n printed lines

corresponds to one constraint. Each line contains the following ten columns:

- For <method> = MD:
  - **column 1**: step number.
  - **column 2**: equilibrium constraint value.
  - **column 3**: instantaneous constraint value.
  - **column 4**: constraint force.
  - **column 5**: average of column 3.
  - **column 6**: average of column 4.
  - **column 7**: potential of mean force with respect to the constraints.
  - **column 8**: potential of mean force with respect to the constraints, including the correction.
  - **column 9**: shortest distance between two groups restrained by the coordination number (zero if no coordination restraint is set).
  - **column 10**: average of column 9.

For <nbead> = n, the data are repeated n times.

12.2.12 cv.out

Used when <method> = MTD. This file contains the trajectory of the collective variables. The print interval can be controlled by the keyword <iprint_cv>. In n-dimensional metadynamics, each of the n printed lines corresponds to one collective variable. Each line contains the following three columns:

- For <method> = MTD:
  - **column 1**: step number.
  - **column 2**: collective-variable coordinate.
  - **column 3**: fictitious-particle coordinate.

For metadynamics with n walkers, the data are repeated n times.

12.2.13 dip.dcd

Used for all methods. This binary file contains the dipole moment. The print interval can be controlled by the keyword <iprint_dcd>. If the dipole moment is not computed in a given method, zeros are written using the same format described below.

The file contains a 276-byte header similar to that of trj.dcd. The important parts are as follows:

- **byte 1–4**: (int) 84, indicating the endianness of the file.
- **byte 17–20**: (int) output interval of the dipole information.
- **byte 269–272**: (int) number of beads.

After the header, the dipole data are organized as follows:

- **byte 1–4**: (int) size of the next read (24).
- **byte 5–28**: (doubles [8 bytes]) X, Y, and Z components of the molecular dipole moment.

This block is repeated for each bead.

12.2.14 dipole.out

Used for all methods. This file contains the dipole moment. The print interval can be controlled by the keyword <iprint_dip>.

- For all methods:
  - **column 1**: step number.

  - **columns 2–4**: x, y, and z components of the dipole moment, (μ_x, μ_y, μ_z), in atomic units.

For <nbead> = n, the data are repeated n times.

12.2.15 dipole_scan.out

Used when <method> = SCAN. This file contains the dipole moment.

- The following data are repeated for each static calculation:
  - **column 1**: step number.
  - **columns 2–4**: x, y, and z components of the dipole moment, (μ_x, μ_y, μ_z), in atomic units.

For multiple beads with <nbead> = P, the data are repeated P times.

12.2.16 dual.out

Used when <method> = PIHMC with <ipotential> = DUAL. This file contains energies from the high- and low-level methods. The print interval can be controlled by the keyword <iprint_dual>.

- For <method> = PIHMC with <ipotential> = DUAL:
  - **column 1**: step number.
  - **column 2**: high-level potential energy.
  - **column 3**: low-level potential energy.

For <nbead> = n, the data are repeated n times.

12.2.17 eavg.out

Used when <method> = MD, PIMD, PIHMC, REHMC. This file contains energy components. The print interval can be controlled by the keyword <iprint_eavg>.

- For <method> = MD:
  - **column 1**: step number.
  - **column 2**: potential energy.
  - **column 3**: kinetic energy.
  - **column 4**: total energy (columns 2 + 3).
  - **columns 5–7**: cumulative averages of columns 2–4.
  - **column 8**: specific heat (NVT) in k_B units.
- For <method> = PIMD, PIHMC:
  - **column 1**: step number.
  - **column 2**: potential energy.
  - **column 3**: kinetic energy (primitive estimator).
  - **column 4**: kinetic energy (virial estimator).
  - **column 5**: total energy (columns 2 + 4).
  - **columns 6–9**: cumulative averages of columns 2–5.
  - **column 10**: specific heat (NVT) in k_B units.
- For <method> = REHMC:
  - **column 1**: step number.
  - **column 2**: bead number.
  - **column 3**: potential energy.
  - **column 4**: cumulative average of the potential energy.

12.2.18 final.xyz

Written when a calculation calls the final-geometry writer. Each bead is stored as one XYZ frame and follows the same coordinate convention as structure.dat. For a periodic system (<iboundary> corresponding to a periodic boundary), the second line is an EXTXYZ comment line containing the current Lattice="..." field, the property definition, pbc="T T T", and the bead energy in hartree. For a free-boundary system the second line begins with ANGSTROM and the energy is appended as a comment. The final step and bead index are also written on the comment line. Atomic records contain species, Cartesian coordinates in angstrom, and the optional PIMD kind index. Consequently, a periodic frame of final.xyz can be reused as an EXTXYZ-style structure.dat frame without reconstructing the cell.

12.2.19 forces.out

Used when <method> = STATIC. This file contains the potential energy and forces.

- For all methods:
  - **line 1**: potential energy in atomic units (hartree).
  - **line 2**: x, y, and z components of the force (negative of the potential gradient) on atom 1 in atomic units (hartree/bohr).
  - **line 3**: x, y, and z components of the force (negative of the potential gradient) on atom 2 in atomic units (hartree/bohr).
  - **line** n: x, y, and z components of the force (negative of the potential gradient) on atom $(n-1)$ in atomic units (hartree/bohr).

Forces are listed in the same order as in the initial geometry file structure.dat. For <nbead> = n, the data are repeated n times.

12.2.20 forces_scan.out

Used when <method> = SCAN. This file contains forces (negative of the potential gradient).

- The following data are repeated for each static calculation:
  - **column 1**: step number.
  - **columns 2–4**: x, y, and z force components in atomic units (hartree/bohr). The data consist of a block of n lines, where n is the number of atoms. Forces are listed in the order specified in structure.dat.

For multiple beads with <nbead> = P, the data are repeated P times.

12.2.21 force.dcd

This binary file contains the calculated forces for the system (negative of the potential gradient). The print interval can be controlled by the keyword <iprint_dcd>.

The file contains a 276-byte header similar to that of trj.dcd. The important parts are as follows:

- **byte 1–4**: (int) 84, indicating the endianness of the file.
- **byte 17–20**: (int) output interval of the force information.
- **byte 269–272**: (int) number of beads times number of atoms.

After the header, the force data are organized as follows. If periodic boundary conditions are present, the box information is written first, as described for trj.dcd. The force components are then written as follows, where **N** is four times the number of atoms times the number of beads:

- **next 4 bytes**: (int) size of the next read (number of beads times number of atoms).
- **next N bytes**: (floats [4 bytes]) x components of the forces.
- **next 4 bytes**: size of the previous read.

This block is repeated for the y and z force components. The order of atoms is the same as in template.xyz. All forces are given in atomic units.

12.2.22 force_high.dcd

This binary file contains the calculated forces for the system computed by the high-level potential. Used when <ipotential> = DUAL. (negative of the potential gradient). The print interval can be controlled by the keyword <iprint_dcd>. This file follows the same layout as force.dcd described above.

12.2.23 force_low.dcd

This binary file contains the calculated forces for the system computed by the low-level potential. Used when <ipotential> = DUAL. (negative of the potential gradient). The print interval can be controlled by the keyword <iprint_dcd>. This file follows the same layout as force.dcd described above.

12.2.24 hessian.out

Used when <method> = NMA, PHONON. This file contains the Hessian matrix, i.e., the second derivative of the potential energy, $H_{ij} = \frac{\partial^2 V}{\partial r_i \partial r_j}$.

- For <method> = NMA, PHONON:
  - **column 1**: index i.
  - **column 2**: index j.
  - **column 3**: H_{ij}.

The data are printed for $1 \leq i, j \leq 3N$, where N is the number of atoms.

12.2.25 logmfd.out

Used when <method> = AFED with <afed_type> = LOGMFD or TAMD. This file contains the time series of the collective variables, fictitious CV particles, mean forces, thermostat variables, and free energy.

- For <afed_type> = LOGMFD, TAMD:
  - **column 1**: AFED step number.
  - **column 2**: free energy in hartree.
  - **column 3**: temperature of the CV particles in kelvin.
  - **column 4**: Nosé–Hoover variable η.
  - **column 5**: Nosé–Hoover variable v_η.
  - **column (4d+2)**: $R_d(\mathbf{r})$ for the d-th CV, averaged over N_m MD steps (<nstep_pro_afed> = N_m).
  - **column (4d+3)**: ξ_d for the d-th fictitious CV particle.
  - **column (4d+4)**: mean force acting on ξ_d for the d-th CV ($1 \leq d \leq n$).

Note that each line corresponds to one AFED step, in contrast to afed.out. Mass values for the CV particles and thermostats are printed at the beginning of the file. The file is appended when a run is restarted.

12.2.26 mech.out

Used when <method> = MTD. This file contains energy components when an external force is applied. The print interval can be controlled by the keyword <iprint_mech>.

- For <method> = MTD:
  - **column 1**: step number.
  - **column 2**: potential energy of the system.
  - **column 3**: potential energy due to the external force.
  - **column 4**: total potential energy (columns 2 + 3).

12.2.27 meta.out

Used when <method> = MTD. This file contains energy components in metadynamics. The print interval can be controlled by the keyword <iprint_meta>.

- For <method> = MTD:
  - **row 1**: step number, $\sum_l V_l^{\mathrm{sys}}$, V^{hills}, $\sum_l V_l^{\mathrm{sys}} + V^{\mathrm{hills}}$.
  - **row 2**: step number, $\sum_l V_l^{\mathrm{cv}}$, V^{cor}, $\sum_l K_l^{\mathrm{sys}}$.
  - **row 3**: step number, $\sum_l K_l^{\mathrm{cv}}$, $H_{\mathrm{bath}}^{\mathrm{sys}}$, $H_{\mathrm{bath}}^{\mathrm{cv}}$.
  - **row 4**: step number, H, T^{sys}, T^{cv}.

See Section 13.13 for definitions.

12.2.28 mfe.out

Used when <method> = MFE. This file contains data from mean-field dynamics. The print interval can be controlled by the keyword <iprint_mfe>.

- For <method> = MFE:
  - **column 1**: step number.
  - **column 2**: bead number.
  - **column 3**: state number.
  - **column 4**: 0 (reserved).
  - **column 5**: population (state coefficient) of each state.
  - **column 6**: occupation ratio of each state.
  - **column 7**: potential energy of each state.
  - **column 8**: average potential energy of each bead.

12.2.29 nac.out

Used when <method> = TFS, MFE. This file contains nonadiabatic-dynamics data, including:

- For <method> = TFS, MFE:
  - **block 1** (2 columns): real and imaginary parts of the state coefficients (m lines).
  - **block 2** (1 column): adiabatic potentials (m^2 lines).
  - **block 3** (3 columns): x, y, and z components of the forces ($n \times m^2$ lines).
  - **block 4** (3 columns): x, y, and z components of the dipole moments (m lines).
  - **block 5** (3 columns): x, y, and z components of the nonadiabatic coupling matrix ($n \times m^2$ lines).

Here, n is the number of atoms and m is the number of adiabatic states. The print interval can be controlled by the keyword <iprint_nac>. See the source files standard_tfs.F90 or standard_mfe.F90 for the detailed data format.

12.2.30 oniom.out

Used when <ipotential> = ONIOM or <ipotential> = HYBRID with <hybrid> = ONIOM. This file contains energy components from QM/MM calculations. The print interval can be controlled by the keyword <iprint_oniom>.

- For <ipotential> = ONIOM:
  - **column 1**: step number.
  - **column 2**: high-level potential energy of subsystem A (or A+L), in hartree.
  - **column 3**: low-level potential energy of subsystem A (or A+L), in hartree.
  - **column 4**: low-level potential energy of the whole system (A+B), in hartree.

12.2.31 phonon_dos.out

Used when <method> = PHONON. This file contains the vibrational density of states (VDOS).

- For <method> = PHONON:
  - **column 1**: phonon frequency in cm^{-1}.
  - **column 2**: density of states.

The frequency range can be controlled by the keyword <dosrange_phonon>.

12.2.32 phonon_energy.out

Used when <method> = PHONON. This file contains vibrational thermodynamic quantities.

- For <method> = PHONON:
  - **column 1**: temperature in kelvin.
  - **column 2**: internal energy (classical Boltzmann statistics).
  - **column 3**: Helmholtz free energy (classical Boltzmann statistics).
  - **column 4**: entropy (classical Boltzmann statistics).
  - **column 5**: internal energy (quantum Boltzmann statistics).
  - **column 6**: Helmholtz free energy (quantum Boltzmann statistics).
  - **column 7**: entropy (quantum Boltzmann statistics).

The temperature range can be controlled by the keyword <temprange_phonon>.

12.2.33 phonon_kdisp.out

Used when <method> = PHONON. This file contains phonon-dispersion data.

- For <method> = PHONON:
  - **columns 1–3**: components of the k vector in bohr^{-1}.
  - **column 4**: vibrational-branch index.
  - **column 5**: vibrational frequency in cm^{-1}.

The k-path (or k-points) is specified by the keyword <kdisp_phonon>.

12.2.34 phonon_kdos.out

Used when <method> = PHONON. This file contains phonon frequencies obtained for the VDOS calculation.

- For <method> = PHONON:
  - **columns 1–3**: components of the k vector in bohr^{-1}.
  - **column 4**: vibrational-branch index.
  - **column 5**: vibrational frequency in cm^{-1}.

The k-point sampling is specified by the keyword <kdos_phonon>.

12.2.35 pot.dcd

This binary file contains the potential energy of each bead, as well as the temperature and Hamiltonian of the whole system. The print interval can be controlled by the keyword <iprint_dcd>.

The file contains a 276-byte header similar to that of trj.dcd. The important parts are as follows:

- **byte 1–4**: (int) 84, indicating the endianness of the file.
- **byte 17–20**: (int) output interval of the potential information.

- **byte 269–272**: (int) number of beads.

After the header, the data are organized as follows:

- **byte 1–4**: (int) size of the next read ($8 \times (2 +$ number of beads)).
- **byte 5–**: (doubles [8 bytes]) potential energies of each bead, followed by the temperature and the Hamiltonian of the system.

Energies and the Hamiltonian are given in atomic units, and the temperature is given in kelvin.

12.2.36 potential_scan.out

Used when <method> = SCAN. This file contains the potential energy.

- The following data are repeated for each static calculation:
  - **column 1**: step number.
  - **column 2**: potential energy in hartree.

For multiple beads with <nbead> = P, the data are repeated P times.

12.2.37 qmmm.out

Used when <ipotential> = QMMM or <ipotential> = HYBRID with <hybrid> = QMMM. This file contains energy components from QM/MM calculations. The print interval can be controlled by the keyword <iprint_qmmm>.

- For <ipotential> = QMMM:
  - **column 1**: step number.
  - **column 2**: high-level potential energy of subsystem A (or A+L), in hartree.
  - **column 3**: MM potential energy of subsystem A (or A+L), in hartree.
  - **column 4**: MM potential energy of the whole system (A+B), in hartree.

12.2.38 rdf.out

Used when <method> = MD, PIMD, PIHMC, REHMC. This file contains atomic pair-distribution data. The print interval can be controlled by the keyword <iprint_rdf>. Mesh parameters are specified by the keyword <params_rdf>.

- For <method> = MD, PIMD, PIHMC, REHMC:
  - **column 1**: step number.
  - **columns 2–3**: atomic-species pair defined in structure.dat.
  - **column 4**: mesh point of the interatomic distance.
  - **column 5**: pair distribution.
  - **column 6**: radial distribution (for <iboundary> = 1), or a copy of column 5 (for <iboundary> = 0).
  - **column 7**: coordination number (number of species 2 around species 1).
  - **column 8**: coordination number (number of species 1 around species 2).

For REHMC, the distribution of each bead i is printed in a separate file, rdf.i.out.

NOTE: This analysis is not recommended when the number of atomic species is large, since the computational cost grows quadratically.

12.2.39 rec.out

Used when <method> = MTD. This file contains the reconstructed free-energy surface from a metadynamics simulation. The print interval can be controlled by the keyword <iprint_rec>. In three-dimensional metadynamics, accuracy and efficiency can be controlled by the keyword <cut_rec_3d>. Before version 2.1.1:

- For <method> = MTD:
  - **column 1**: mesh point of the collective variable.
  - **column 2**: sum of history-dependent Gaussian hills (equal to the negative reconstructed free energy).

Version 2.1.1 and later:

- For <method> = MTD:
  - **column 1**: mesh point of the collective variable.
  - **column 2**: reconstructed free energy (positive sign).

12.2.40 rgy.out

Used when <method> = PIMD, PIHMC. This file contains the atomic radius of gyration. The print interval can be controlled by the keyword <iprint_rgy>.

- For <method> = PIMD, PIHMC:
  - **column 1**: step number.
  - **column 2**: atomic species defined in structure.dat.
  - **column 3**: radius of gyration.

12.2.41 standard.out

Used in all cases. This file contains the standard output. The print interval can be controlled by the keyword <iprint_std>.

- For <method> = MD:
  - **column 1**: step number.
  - **column 2**: total energy.
  - **column 3**: potential energy.
  - **column 4**: instantaneous temperature.
  - **column 5**: wall-clock time.
- For <method> = PIMD, CMD, RPMD, MTD:
  - **column 1**: step number.
  - **column 2**: total energy.
  - **column 3**: bead-averaged potential energy.
  - **column 4**: instantaneous temperature.
  - **column 5**: wall-clock time.
- For <method> = GEOOPT:
  - **column 1**: step number.
  - **column 2**: potential energy.
  - **column 3**: maximum displacement.
  - **column 4**: root-mean-square displacement.
  - **column 5**: maximum force.
  - **column 6**: root-mean-square force.
- For <method> = SD:
  - **column 1**: step number.
  - **column 2**: potential energy.
- For <method> = REHMC:
  - **column 1**: step number.
  - **column 2**: step size.
  - **column 3**: acceptance ratio of the HMC trial move.

  - **column 4**: acceptance ratio of the REM trial move.
  - **column 5**: bead-averaged potential energy.
  - **column 6**: wall-clock time.
- For <method> = PIHMC:
  - **column 1**: step number.
  - **column 2**: step size.
  - **column 3**: acceptance ratio of the HMC trial move.
  - **column 4**: bead-averaged potential energy.
  - **column 5**: instantaneous temperature.
  - **column 6**: wall-clock time.
- For <method> = STRING:
  - **column 1**: step number.
  - **column 2**: image with the maximum potential energy.
  - **column 3**: maximum potential energy.
  - **column 4**: image with the minimum potential energy.
  - **column 5**: minimum potential energy.
  - **column 6**: wall-clock time.
- For <method> = OMOPT:
  - **column 1**: step number.
  - **column 2**: Onsager–Machlup action.
  - **column 3**: bead with the maximum potential energy.
  - **column 4**: maximum potential energy.
  - **column 5**: wall-clock time.

12.2.42 string.out

Used when <method> = STRING. This file contains the potential energies of the images along the string at each step.

- **column 1**: step number.
- **column 2**: image number.
- **column 3**: mass-weighted reaction coordinate [a.u.].
- **column 4**: potential energy [hartree].

The print interval can be controlled by the keyword <iprint_str>.

12.2.43 string.xyz

Used when <method> = STRING. This file contains the positions of the images along the string at the final step in xyz format. The print interval can be controlled by the keyword <iprint_str>.

12.2.44 string_final.out

Used when <method> = STRING. This file contains the final potential energies of the images along the string.

- **column 1**: step number.
- **column 2**: image number.
- **column 3**: mass-weighted reaction coordinate [a.u.].
- **column 4**: potential energy [hartree].

The print interval can be controlled by the keyword <iprint_rest>.

12.2.45 template.xyz

Used in all cases except <method> = STATIC, NMA, PHONON, when the trajectory is written in DCD format via the <iprint_dcd> keyword. This file contains copies of the initial coordinates repeated n times, where n is the number of beads/replicas/images. It is provided to simplify reading trj.dcd by supplying atomic information for the coordinates stored in that file. This file does *not* contain trajectory data. When using it to assist with reading DCD data, be sure to remove or ignore these duplicated initial-coordinate records before further analysis.

12.2.46 tfs.out

Used when <method> = TFS. This file contains data from the surface-hopping method. The print interval can be controlled by the keyword <iprint_tfs>.

- For <method> = TFS:
  - **column 1**: step number.
  - **column 2**: bead number.
  - **column 3**: state number.
  - **column 4**: occupied state.
  - **column 5**: population (state coefficient) of each state.
  - **column 6**: occupation ratio of each state.
  - **column 7**: potential energy of each state.
  - **column 8**: potential energy of the occupied state.

12.2.47 trj.dcd

Used in all cases except <method> = STATIC, NMA, PHONON. This file contains the trajectory in DCD format. For simulations with multiple beads/replicas/images (<nbead> = n), the data are repeated n times. The print interval can be controlled by the keyword <iprint_dcd>. To assist in reading the DCD file, template.xyz is also written when DCD output is enabled. This file contains the coordinates of the initial structure repeated n times, where n is the number of beads/replicas/images. Because the coordinates in template.xyz are *not* part of the simulation trajectory, you should delete or ignore them in any subsequent trajectory analysis. Coordinates are written unwrapped when periodic boundary conditions are used, so diffusion can be followed without unfolding the trajectory.

The format starts with a 276-byte header, described below. This header is also used in other *.dcd outputs, with minor modifications described in the corresponding sections of this manual. The header contents are as follows:

- **byte 1–4**: (int) 84, indicating the endianness of the file.
- **byte 5–8**: (chars [1 byte]) CORD.
- **byte 9–12**: (int) 0.
- **byte 13–16**: (int) starting step (usually 0).
- **byte 17–20**: (int) trajectory output interval.
- **byte 21–44**: (ints [4 bytes]) 0, 0, 0, 0, 0, 0.
- **byte 45–48**: (float) time step in AKMA units.
- **byte 49–52**: (int) 1 if PBC are used, 0 otherwise.
- **byte 53–84**: (ints [4 bytes]) 0, 0, 0, 0, 0, 0, 0, 0.
- **byte 85–88**: (int) 24.
- **byte 89–92**: (int) 84.
- **byte 93–96**: (int) 164.
- **byte 97–100**: (int) 2.
- **byte 101–180**: (chars [1 byte]) a string, 'PIMD - TRAJECTORY\0'.
- **byte 181–260**: (chars [1 byte]) a string containing the date of the calculation.

- **byte 261–264**: (int) 164.
- **byte 265–268**: (int) 4.
- **byte 269–272**: (int) number of beads times number of atoms.
- **byte 273–276**: (int) 4.

After the header, the trajectory data are organized as follows. If periodic boundary conditions are present, the box information is written first (same format as described above for other DCD outputs):

- **byte 1–4**: (int) size of the next read (48).
- **byte 5–52**: (doubles [8 bytes]) box information ordered as A, γ, B, β, α, C. Here, A, B, and C are side lengths in Å, and α, β, and γ are angles in radians.
- **byte 53–56**: (int) size of the previous read (48).

The coordinates are then written as follows, where **N** is four times the number of atoms times the number of beads:

- **next 4 bytes**: (int) size of the next read (number of beads times number of atoms).
- **next N bytes**: (floats [4 bytes]) x coordinates.
- **next 4 bytes**: size of the previous read.

This block is repeated for the y and z coordinates. The trajectory data are wrapped so that coordinates lie inside the periodic box (if present). All atomic coordinates are given in Å.

12.2.48 trj.out

Used in all cases except **<method>** = `STATIC, NMA, PHONON`. This file contains the standard trajectory output. The print interval can be controlled by the keyword **<iprint_trj>**.

- For all methods:
  - **column 1**: step number.
  - **columns 2–4**: atomic position (x, y, z) in atomic units (bohr).
  - **columns 5–7**: atomic velocity (v_x, v_y, v_z) in atomic units (bohr per atomic time), where atomic time $= \hbar/\text{hartree} \simeq 0.0241888$ fs.
  - **columns 8–10**: atomic force (f_x, f_y, f_z) in atomic units (hartree/bohr).
  - **column 11**: potential energy V in atomic units (hartree).

For simulations with multiple beads/replicas/images (**<nbead>** $= n$), the data are repeated n times. The block with the same step number corresponds to one configuration.

12.2.49 trj.xsf

Written when **<iprint_xsf>** is positive. In path-integral calculations, *one bead is one XSF animation frame*. The energy comment, simulation step, bead number, cell (for periodic boundaries), coordinates, and forces of that bead are kept in the same frame. Thus a configuration with **<nbead>**$= P$ contributes P consecutive XSF frames instead of one frame containing all P beads. For periodic systems each bead frame contains `CRYSTAL`, `PRIMVEC`, and `PRIMCOORD` records; for free boundaries it contains an `ATOMS` frame. The frame index is sequential throughout the trajectory.

12.2.50 trj.xyz

Used in all cases except **<method>** = `STATIC, NMA, PHONON`. This file contains the trajectory in xyz format.

- For all methods:
  - **line 1**: number of atoms.
  - **line 2**: step number.
  - **lines 3–**$(n+2)$: atomic species followed by Cartesian coordinates (x, y, z) for each atom. For periodic boundary conditions, coordinates may be duplicated to display both folded and unfolded trajectories.

For simulations with multiple beads/replicas/images (<nbead> = n), the data are repeated n times. The print interval can be controlled by the keyword <iprint_xyz>. The output is controlled by the options <iformat_xyz> and <jorigin_xyz>.

12.2.51 trjext.xyz

Written when <iprint_extxyz> is positive. This file contains the trajectory in extended xyz format for use with machine-learning potential workflows. For simulations with multiple beads/replicas/images (<nbead> = n), each bead is written as a separate extended xyz frame. Some methods write beadwise trajectories to `trjext.001.xyz`, `trjext.002.xyz`, ... instead of combining all bead frames in `trjext.xyz`.

- For all methods:
  - **line 1**: number of atoms.
  - **line 2**: extended xyz header. The header contains `Lattice` in Å, `Properties`, `energy` in eV, the total frame dipole `dipole` in e Å, periodic-boundary flags `pbc`, the simulation step `Step`, and the bead number `Bead`.
  - **lines 3–**$(n+2)$: atomic species, Cartesian coordinates (x, y, z) in Å, and forces (f_x, f_y, f_z) in eV/Å.
  - When charges are available, an atom-wise `charge` property is also written in units of the elementary charge.

For <ipotential> = `DUAL`, the high-level potential, force, and dipole are written.

12.2.52 vel.dcd

This binary file contains particle velocities. The print interval can be controlled by the keyword <iprint_dcd>.

The file contains a 276-byte header similar to that of `trj.dcd`. The important parts are as follows:

- **byte 1–4**: (int) 84, indicating the endianness of the file.
- **byte 17–20**: (int) output interval of the velocity information.
- **byte 269–272**: (int) number of beads times number of atoms.

After the header, the velocity data are organized as follows. If periodic boundary conditions are present, the box information is written first:

- **byte 1–4**: (int) size of the next read (48).
- **byte 5–52**: (doubles [8 bytes]) box information ordered as A, γ, B, β, α, C. Here, A, B, and C are side lengths in Å, and α, β, and γ are angles in radians.
- **byte 53–56**: (int) size of the previous read (48).

The velocity data are then written as follows, where **N** is four times the number of atoms times the number of beads:

- **next 4 bytes**: (int) size of the next read (number of beads times number of atoms).
- **next N bytes**: (floats [4 bytes]) x velocities.
- **next 4 bytes**: size of the previous read.

This block is repeated for the y and z velocities. The atom order is the same as in `template.xyz`. All velocities are given in Å/ps.

12.3 Restart files

12.3.1 afed.ini

Used in the case of <method> = `AFED` with <afed_type> = `DESCENT, ASCENT, AUTO, TAMD, LOGMFD`. This file contains the latest values of adiabatic free energy dynamics, temperature-accelerated molecular dynamics, and logarithmic mean force dynamics. Each line contains the following columns, where n is the dimension of the collective variables and m is the length of the Nosé-Hoover chain thermostats:

- For <afed_type> = DESCENT, ASCENT, AUTO:
  - **column 1**: AFED status.
    - DE: descent trajectory to a free-energy minimum point.
    - D1: descent trajectory started in the $+\mathbf{n}$ direction from a free-energy saddle point to a free-energy minimum point.
    - D2: descent trajectory started in the $-\mathbf{n}$ direction from a free-energy saddle point to a free-energy minimum point.
    - AS: ascent trajectory to search for a free-energy saddle point.
    - EQ: free-energy minimum point.
    - TS: free-energy saddle point.
    - HI: point terminated where the free energy exceeds the limit defined by the keyword <fenergy_max_afed>.
    - OB: point terminated where the CV is out of bounds defined by the keyword <params_afed>.
  - **column 2**: AFED step number (iteration number).
  - **column 3**: free energy in hartree.
  - **column 4-(3+n)**: the position of the fictitious particle and the unit vector $\mathbf{n}$.
  - **column (4+n)**: root point ID (starting point of the trajectory that leads to this point).
  - **column (5+n)**: AFED step size.
- For <afed_type> = TAMD:
  - **column 1**: AFED status.
    - TA: TAMD trajectory.
    - HI: point terminated where the free energy exceeds the limit defined by the keyword <fenergy_max_afed>.
    - OB: point terminated where the CV is out of bounds defined by the keyword <params_afed>.
  - **column 2**: AFED step number (iteration number).
  - **column 3**: free energy in hartree.
  - **column 4-(3+n)**: the position and the mass-weighted velocity of the fictitious particle.
  - **column (4+$n+m$)**: the position and the mass-weighted velocity of the Nosé-Hoover thermostats (only if they exist).
- For <afed_type> = LOGMFD:
  - **column 1**: AFED status.
    - LO: LogMFD trajectory.
    - HI: point terminated where the free energy exceeds the limit defined by the keyword <fenergy_max_afed>.
    - OB: point terminated where the CV is out of bounds defined by the keyword <params_afed>.
  - **column 2**: AFED step number (iteration number).
  - **column 3**: free energy in hartree.
  - **column 4-(3+n)**: the position and the mass-weighted velocity of the fictitious particle.
  - **column (4+n)**: the origin of the free energy in hartree.
  - **column (5+$n+m$)**: the position and the mass-weighted velocity of the Nosé-Hoover thermostats (only if they exist).

The file is overwritten for restarted runs.

12.3.2 auto.ini

Used in the case of <method> = AFED with <afed_type> = AUTO. This file contains the list of landmarks in the AFED automated search. Each line contains the following columns, where n is the dimension of the collective variables:

- **column 1**: landmark number.

- **column 2**: AFED status.
  - EQ: free-energy minimum point.
  - TS: free-energy saddle point.
  - HI: point terminated where the free energy exceeds the limit.
  - OB: point terminated where the CV is out of bounds.
- **column 3**: landmark label.
  - NEW: new point in the list.
  - OLD: old point in the list.
- **column 4**: free energy in hartree.
- **column 5**: landmark ID.
- **column 6**: name of restart tar file.
- **column 7**: root point ID (starting point of the trajectory that leads to this point).
- **column 8-(7+n)**: position of each collective variable.
- **column (8+n)-(7+2n)**: unit vector $\mathbf{n}$ in the ascent trajectory, which should be parallel to the imaginary-frequency mode. These columns are present only when the AFED status is TS.

The file is appended for restarted runs.

12.3.3 averages.ini

Used in ALL methods. This file contains the data of statistical averages at the final step of the last run. If this file is not found at the beginning of the run, the corresponding data are initialized.

The data affect the results in the output files that include averages (bond.out, eavg.out, rdf.out, rgy.out). They also affect the acceptance ratio in PIHMC and REHMC methods (standard.out).

IMPORTANT: Do not change the keywords <iprint_...> in the file input.dat in restarted runs, or the average data may be corrupted.

12.3.4 bath.ini

Used in the cases of <method> = MD, PIMD, CMD, MTD. This file contains the thermostat positions and the thermostat velocities at the final step of the last run. If this file is not found at the beginning of the run, the data are initialized.

NOTE: The data depend on the bath type, the method, and the system. Exceptionally, the data are transferable when switching from <method> = PIMD to <method> = CMD when <bath_type> = MNHC is chosen.

12.3.5 box.ini

Used in ALL calculations with periodic boundary conditions, i.e., when <iboundary> = 1. This file contains the step number and the box matrix $h(i, j)$ at the final step of the last run.

step number, $h(1,1)$, $h(1,2)$, $h(1,3)$
step number, $h(2,1)$, $h(2,2)$, $h(2,3)$
step number, $h(3,1)$, $h(3,2)$, $h(3,3)$

If this file is not found at the beginning of the run, the data are read from the file input.dat under the keyword <iboundary>.

12.3.6 cstate.ini

Used in the cases of <method> = TFS, MFE. This file contains the state coefficients (complex numbers) at the final step of the last run. If this file is not found at the beginning of the run, the data are read from the file input.dat under the keyword <istate_init>.

12.3.7 cv.ini

Used in the case of <method> = MTD. This file contains the positions and velocities of the collective variables at the final step of the last run. If this file is not found at the beginning of the run, the data are initialized.

12.3.8 gad.ini

Used in the case of <method> = GAD. This file contains the latest values of gentlest ascent dynamics. Each line contains the data of the guiding unit vector of each trajectory, followed by the convergence flag of each trajectory.

12.3.9 geometry.ini

Used in ALL methods except <method> = STATIC, NMA, PHONON. This file contains the latest values of the positions and velocities of atoms at the final step of the last run. If this file is not found at the beginning of the run, the data are read from the file structure.dat.

The data are transferable among <method> = STATIC, GEOOPT, BOXOPT, FULLOPT, ELASTIC, SD, NMA, PHONON, MD, with the following format for n atoms.

$$
\begin{array}{llllllllll}
0, & x_{1,1}, & y_{1,1}, & z_{1,1}, & vx_{1,1}, & vy_{1,1}, & vz_{1,1}, & n_{1,1,1}, & n_{2,1,1}, & n_{3,1,1} \\
\cdots & & & & & & & & & \\
0, & x_{n,1}, & y_{n,1}, & z_{n,1}, & vx_{n,1}, & vy_{n,1}, & vz_{n,1}, & n_{1,n,1}, & n_{2,n,1}, & n_{3,n,1}
\end{array}
$$

where indices $(n_{1,i,j}, n_{2,i,j}, n_{3,i,j})$ are important only for periodic boundary conditions. The indices indicate the displacement with respect to the box matrix in the folded trajectories.

The data are also transferable among the path-integral methods, <method> = PIMD, CMD, RPMD, PIHMC, where the normal-mode positions and velocities are given in mass-weighted coordinates.

12.3.10 hills.ini

Used in the case of <method> = MTD. This file contains the list of data for the history-dependent Gaussian hill potential.

- **column 1**: metadynamics step.
- **column 2**: Gaussian height in hartree.
- **column 3**: Gaussian width in bohr, degrees, etc.
- **column 4**: Gaussian center in bohr, degrees, etc.

The unit of the Gaussian width corresponds to the unit used under the keyword <gw_meta> in input.dat. One data unit consists of m lines, where m is the dimension of the collective variables, which is given by the keyword <nmeta> in input.dat. If this file is not found at the beginning of the run, the data are initialized with no Gaussian hills.

This file is read at the beginning of the metadynamics run. The user may simply add “wall potentials” (high Gaussian hills with small width) to the list, to limit the exploration of the collective variables. In this case, the step number can be any integer.

Note that in well-tempered metadynamics, the Gaussian hills are not the negative of the free energy. Thus, the deposited hills cannot be transferred between conventional and well-tempered metadynamics.

12.3.11 step.ini

Used in ALL methods except <method> = STATIC, NMA, PHONON. This file contains the final step number of the last run. If this file is not found at the beginning of the run, the data are initialized to zero.

12.3.12 string.ini

Used in <method> = STRING, OMOPT. This file contains the latest values of the positions and velocities of atoms at the final step of the last run. If this file is not found at the beginning of the run, the data are read from the file structure.dat.

This file can be edited by the user to start from a guessed path. For a system of n atoms with m images, the data should be provided in the following format.

$$
\begin{array}{llll}
0, & x_{1,1}, & y_{1,1}, & z_{1,1} \\
\cdots & & & \\
0, & x_{n,1}, & y_{N,1}, & z_{n,1} \\
0, & x_{1,2}, & y_{1,2}, & z_{1,2} \\
\cdots & & & \\
0, & x_{n,2}, & y_{N,2}, & z_{n,2} \\
\cdots & & & \\
\cdots & & & \\
0, & x_{1,m}, & y_{1,m}, & z_{1,m} \\
\cdots & & & \\
0, & x_{n,m}, & y_{n,m}, & z_{n,m}
\end{array}
$$

where $(x_{i,j}, y_{i,j}, z_{i,j})$ are the Cartesian coordinates of the i-th atom in the j-th image in bohr. "0" in the first column is the step number.

Part VI

Theoretical Background

13 Theoretical background

In this section, the theoretical background of the methods implemented in the PIMD code is briefly summarized. Throughout this section, the following notation is used.

- Scalars are denoted by regular font (e.g., x), vectors by bold font (e.g., $\mathbf{x}$), and matrices by bold font with arrows (e.g., $\overleftrightarrow{\mathbf{x}}$).
- The number of atoms is denoted by N.
- The mass of the i-th atom is denoted by m_i.
- The set of atomic positions (nuclear coordinates) is denoted by

$$\{\mathbf{r}\} = \{\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N\} = \{r_{1x}, r_{1y}, r_{1z}, \cdots, r_{Nx}, r_{Ny}, r_{Nz}\}. \tag{12}$$

  When P beads are used, the atomic positions of the s-th bead are

$$\{\mathbf{r}^{(s)}\} = \{\mathbf{r}_1^{(s)}, \mathbf{r}_2^{(s)}, \cdots, \mathbf{r}_N^{(s)}\} = \{r_{1x}^{(s)}, r_{1y}^{(s)}, r_{1z}^{(s)}, \cdots, r_{Nx}^{(s)}, r_{Ny}^{(s)}, r_{Nz}^{(s)}\}, \tag{13}$$

  for $1 \le s \le P$.

- The potential energy is expressed as
$$V = V(\{\mathbf{r}\})\,. \tag{14}$$
- The set of atomic momenta is denoted by
$$\{\mathbf{p}\} = \left\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\right\} = \left\{p_{1x}, p_{1y}, p_{1z}, \cdots, p_{Nx}, p_{Ny}, p_{Nz}\right\}. \tag{15}$$
When P beads are used, the momenta of the s-th bead are
$$\left\{\mathbf{p}^{(s)}\right\} = \left\{\mathbf{p}_1^{(s)}, \mathbf{p}_2^{(s)}, \cdots, \mathbf{p}_N^{(s)}\right\} = \left\{p_{1x}^{(s)}, p_{1y}^{(s)}, p_{1z}^{(s)}, \cdots, p_{Nx}^{(s)}, p_{Ny}^{(s)}, p_{Nz}^{(s)}\right\}. \tag{16}$$
The phase-space coordinates of the s-th bead are then written as
$$\mathbf{\Gamma}^{(s)} = \left\{\mathbf{r}^{(s)}, \mathbf{p}^{(s)}\right\}. \tag{17}$$
- The physical atomic masses are denoted by m_i. When fictitious masses are introduced, they are denoted by μ_i.
- The simulation box is described by the box matrix
$$\overleftrightarrow{\mathbf{h}} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} = \begin{pmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{pmatrix}, \tag{18}$$
where $\mathbf{a} = (a_x, a_y, a_z)$, $\mathbf{b} = (b_x, b_y, b_z)$, and $\mathbf{c} = (c_x, c_y, c_z)$ are the box vectors. The volume of the simulation box is given by
$$V_{\text{cell}} = \det\left(\overleftrightarrow{\mathbf{h}}\right). \tag{19}$$
- The temperature is denoted by T, and
$$\beta = \frac{1}{k_{\text{B}}T}, \tag{20}$$
where k_{B} is the Boltzmann constant.
- The external pressure is denoted by P_{ext}. In the general case, the external stress (or tension) is represented by a real symmetric matrix
$$\overleftrightarrow{\mathbf{t}} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{12} & t_{22} & t_{23} \\ t_{13} & t_{23} & t_{33} \end{pmatrix}. \tag{21}$$

13.1 Mechanochemistry

To deal with mechanochemistry, the external force explicitly included (EFEI) method is employed. A constant repulsive force, F, is applied to the specified pair of atoms i and j in the direction of the interatomic vector. The potential
$$V^{\text{efei}} = -Fr_{ij} \tag{22}$$
and the force
$$F_{i\alpha}^{\text{efei}} = F\frac{r_{i\alpha} - r_{j\alpha}}{r_{ij}} \tag{23}$$
are added to the physical potential and the physical force, respectively, where
$$r_{ij} = \sqrt{\sum_{\alpha=x,y,z}\left(r_{i\alpha} - r_{j\alpha}\right)^2} \tag{24}$$
is the distance between atoms i and j.

13.2 Steepest descent

The intrinsic reaction coordinate (IRC) is the steepest descent path starting from the saddle point and ending at the minimum point on the potential energy surface in mass-weighted coordinate space. The mass-weighted coordinate is defined as

$$\mathbf{q} = \left\{\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N\right\} = \left\{\sqrt{m_1}\mathbf{r}_1, \sqrt{m_2}\mathbf{r}_2, \cdots, \sqrt{m_N}\mathbf{r}_N\right\}. \tag{25}$$

The steepest descent path is defined by the trajectory of the equation

$$\frac{d\mathbf{q}_i}{dt} = -\frac{\mathbf{g}_i}{|\mathbf{g}|}. \tag{26}$$

where the potential gradient with respect to $\mathbf{q}_i$ is defined by

$$\mathbf{g}_i = \frac{dV}{d\mathbf{q}_i} = \frac{1}{\sqrt{m_i}}\frac{dV}{d\mathbf{r}_i} \tag{27}$$

and the norm is given by

$$|\mathbf{g}|^2 = \sum_{i=1}^{N}\frac{1}{m_i}\left\{\left(\frac{dV}{dx_i}\right)^2 + \left(\frac{dV}{dy_i}\right)^2 + \left(\frac{dV}{dz_i}\right)^2\right\}. \tag{28}$$

Eq. (26) is equivalent to

$$m_i\frac{d\mathbf{r}_i}{dt} = -\frac{1}{|\mathbf{g}|}\frac{dV}{d\mathbf{r}_i}. \tag{29}$$

In the PIMD code, Eq. (29) is solved using a fourth-order Runge–Kutta algorithm with step size dt. Note that dt has the dimension $(\sqrt{\text{mass}} \times \text{length})$ or $(\sqrt{\text{energy}} \times \text{time})$.

13.3 Box optimization

The box matrix, $h_{\alpha\beta}$, can be optimized using the limited-memory BFGS algorithm [3].

13.3.1 Zero pressure condition

At zero pressure, the bare potential V is minimized. The potential gradient with respect to the box

$$G_{\alpha\beta} = \frac{\partial V}{\partial h_{\alpha\beta}} \tag{30}$$

is related to the virial,

$$A_{\alpha\beta} = -\sum_{\gamma=1}^{3}\frac{\partial V}{\partial h_{\alpha\gamma}}h_{\beta\gamma} = -\sum_{\gamma=1}^{3}G_{\alpha\gamma}h_{\beta\gamma}, \tag{31}$$

and thus,

$$G_{\alpha\beta} = -\sum_{\gamma=1}^{3}A_{\alpha\gamma}h^{-1}_{\beta\gamma}. \tag{32}$$

Note that the virial $A_{\alpha\beta}$ is associated with the internal pressure tensor at zero temperature,

$$P_{\alpha\beta}(0) = V^{-1}_{\text{cell}}A_{\alpha\beta}. \tag{33}$$

13.3.2 Finite pressure condition

In the presence of an external pressure, P_{ext}, the effective potential

$$\tilde{V} = V + P_{\text{ext}}V_{\text{cell}} \tag{34}$$

is minimized. Then, the gradient of the effective potential is given by

$$\tilde{G}_{\alpha\beta} = \frac{\partial\tilde{V}}{\partial h_{\alpha\beta}} = \frac{\partial V}{\partial h_{\alpha\beta}} + P_{\text{ext}}V_{\text{cell}}h^{-1}_{\beta\alpha}. \tag{35}$$

13.3.3 Finite tension condition

In the presence of an external thermodynamic tension, $\overleftrightarrow{\mathbf{t}}$, the effective potential

$$\tilde{V} = V - \bar{V}_{\text{cell}} \sum_{\alpha,\beta=1}^{3} \left(t_{\alpha\beta} \epsilon_{\alpha\beta} \right) \tag{36}$$

is minimized.

$$\epsilon_{\alpha\beta} = \frac{1}{2} \left\{ \sum_{\mu,\nu,\lambda=1}^{3} \left(\bar{h}^{-1}_{\mu\alpha} h_{\lambda\mu} h_{\lambda\nu} \bar{h}^{-1}_{\nu\beta} \right) - \delta_{\alpha\beta} \right\} \tag{37}$$

is the strain tensor, where $\bar{\mathbf{h}}$ and $\bar{V}_{\text{cell}} = \det(\bar{\mathbf{h}})$ are the box matrix and the volume, respectively, in the absence of external tension. Then, the gradient of the effective potential is given by

$$\tilde{G}_{\alpha\beta} = \frac{\partial \tilde{V}}{\partial h_{\alpha\beta}} = \frac{\partial V}{\partial h_{\alpha\beta}} - \bar{V}_{\text{cell}} \sum_{\mu,\nu,\lambda=1}^{3} \left(h_{\alpha\lambda} \bar{h}^{-1}_{\lambda\mu} t_{\mu\nu} \bar{h}^{-1}_{\beta\nu} \right). \tag{38}$$

13.4 Static elastic constants

The static elastic constants are defined by

$$C_{\alpha\beta,\gamma\delta} = \frac{1}{\Omega} \frac{\partial^2 V}{\partial \epsilon_{\alpha\beta} \partial \epsilon_{\gamma\delta}}, \tag{39}$$

where the strain tensor is given by

$$\epsilon_{\alpha\beta} = \frac{1}{2} \left(\sum_{\nu,\gamma,\lambda=x,y,z} \bar{h}^{-1}_{\gamma\alpha} h_{\lambda\gamma} h_{\lambda\nu} \bar{h}^{-1}_{\nu\beta} - \delta_{\alpha\beta} \right). \tag{40}$$

The elastic constants are invariant under permutations of the indices between α and β, between γ and δ, and between (α, β) and (γ, δ). In Voigt notation, the constants are abbreviated as C_{ij}, where $1 \le i, j \le 6$, and 1–6 correspond to (x,x), (y,y), (z,z), (y,z), (z,x), and (x,y), respectively.

13.5 Normal mode analysis

The mass-weighted Hessian matrix is defined as a $(3N \times 3N)$ real symmetric matrix:

$$\tilde{H}_{i\alpha,j\beta} = \frac{1}{\sqrt{m_i m_j}} \frac{\partial^2 V}{\partial r_{i\alpha} \partial r_{j\beta}}. \tag{41}$$

By diagonalizing $\tilde{H}$ at a stationary point, one obtains the eigenvalues ε_k and the corresponding eigenvectors $U_{i\alpha,k}$ for each mode k $(1 \le k \le 3N)$. The vibrational frequencies, converted from atomic units to wavenumbers (cm^{-1}), are given by

$$\nu_k = \frac{1}{100} \times \frac{E_{\text{h}}}{hc} \sqrt{\varepsilon_k}, \tag{42}$$

where c is the speed of light, h is Planck's constant, and $E_{\text{h}} = 4.35974417 \times 10^{-18}$ J is one hartree in SI units. The reduced mass of the k-th mode in atomic mass units is defined as

$$\mu_k = \frac{1}{1822.88853} \times \frac{1}{\sum_{i=1}^{N} \sum_{\alpha=x,y,z} m_i^{-1} U_{i\alpha,k}^2}, \tag{43}$$

and the normalized mass-weighted normal mode vector is

$$\tilde{U}_{i\alpha,k} = \frac{m_i^{-1/2} U_{i\alpha,k}}{\sqrt{\sum_{i=1}^{N} \sum_{\alpha=x,y,z} m_i^{-1} U_{i\alpha,k}^2}}. \tag{44}$$

For free boundary systems, there are six (five) zero-frequency modes, corresponding to three translations and three (two) rotations for nonlinear (linear) molecules. For periodic boundary systems, there are three zero-frequency modes corresponding to pure translations. In the PIMD code, numerical errors in the Hessian are corrected by applying a projection operator to remove contributions from zero-frequency translational and rotational modes.

13.6 Phonon calculations

Let the vector of three integers, $\mathbf{n} = (n_1, n_2, n_3)$, denote the numbering of the respective unit cells in a crystal. Let N atoms be contained in the unit cell defined by the matrix of three lattice vectors, $\overleftrightarrow{\mathbf{h}} = (\mathbf{a}, \mathbf{b}, \mathbf{c})$. Let $\mathbf{r}_{i,\mathbf{0}}$ be the position of the i-th atom in the unit cell at the origin, $\mathbf{0} = (0, 0, 0)$. Then, the atomic positions in the unit cell labeled by $\mathbf{n}$ are given by

$$\mathbf{r}_{i,\mathbf{n}} = \mathbf{r}_{i,\mathbf{0}} + \mathbf{h}^{-1}\mathbf{n}. \tag{45}$$

For a given wave vector $\mathbf{K}$ in reciprocal space, the $(3N \times 3N)$ dynamical matrix is defined as

$$D_{j\alpha,k\beta}(\mathbf{K}) = \frac{1}{\sqrt{m_j m_k}} \sum_{\mathbf{n}} \frac{\partial^2 V}{\partial r_{j\alpha,\mathbf{0}} \partial r_{k\beta,\mathbf{n}}} \exp\left(i\mathbf{K} \cdot \left(\mathbf{r}_{j,\mathbf{0}} - \mathbf{r}_{k,\mathbf{n}}\right)\right). \tag{46}$$

By diagonalizing this matrix, one obtains the eigenvalues $\varepsilon_k(\mathbf{K})$, where k is the index of the vibrational mode. The phonon frequencies are obtained by converting from atomic units to cm^{-1},

$$\nu_k(\mathbf{K}) = \frac{1}{100} \times \frac{E_\mathrm{h}}{hc} \sqrt{\varepsilon_k(\mathbf{K})}. \tag{47}$$

The dynamical matrix is evaluated and diagonalized for each wave vector $\mathbf{K}$ within the first Brillouin zone,

$$\left(K_x, K_y, K_z\right)_{l_1,l_2,l_3} = \left(\sum_{\beta=1}^{3} \frac{\pi}{L_\beta} l_\beta h^{-1}_{\beta x}, \sum_{\beta=1}^{3} \frac{\pi}{L_\beta} l_\beta h^{-1}_{\beta y}, \sum_{\beta=1}^{3} \frac{\pi}{L_\beta} l_\beta h^{-1}_{\beta z}\right). \tag{48}$$

The evenly spaced grid points with $-L_\beta \le l_\beta \le L_\beta$ $(1 \le \beta \le 3)$ are used to obtain the phonon density of states,

$$\rho(\nu) = \sum_{\mathbf{K}} \left(\nu - \nu_k(\mathbf{K})\right). \tag{49}$$

The classical and quantum internal energies per unit cell are given by

$$U^\mathrm{cl} = (3N - 3)\, k_\mathrm{B} T, \tag{50}$$

and

$$U^\mathrm{qm} = \frac{1}{N_\mathbf{K}} \sum_{\mathbf{K}} \sum_{m=4}^{3N} \frac{\hbar\omega_m(\mathbf{K})}{2} \coth\left(\frac{\beta\hbar\omega_m(\mathbf{K})}{2}\right), \tag{51}$$

respectively. The classical and quantum Helmholtz free energies per unit cell are given by

$$A^\mathrm{cl} = \frac{1}{N_\mathbf{K}} \sum_{\mathbf{K}} \sum_{m=4}^{3N} k_\mathrm{B} T \log\left(\beta\hbar\omega_m(\mathbf{K})\right), \tag{52}$$

and

$$A^\mathrm{qm} = \frac{1}{N_\mathbf{K}} \sum_{\mathbf{K}} \sum_{m=4}^{3N} k_\mathrm{B} T \log\left(1 - \exp\left(-\beta\hbar\omega_m(\mathbf{K})\right)\right), \tag{53}$$

respectively. The entropy per unit cell is defined as

$$S^\mathrm{cl} = \frac{U^\mathrm{cl} - A^\mathrm{cl}}{T} \tag{54}$$

and

$$S^\mathrm{qm} = \frac{U^\mathrm{qm} - A^\mathrm{qm}}{T}, \tag{55}$$

respectively. In the PIMD code, numerical errors in the dynamical matrix are removed by applying a projection operator with respect to the zero-frequency translational modes.

13.7 String method

The String Method is an efficient algorithm for obtaining the minimum energy path or intrinsic reaction coordinate that connects the reactant and the product.

The path is described in terms of P discrete images along the path in mass-weighted coordinates, denoted as $\mathbf{q}$ in Eq. (25).

Starting from an initial guess, the images are optimized by iterating the following two-step procedure:

- Step 1: Move the positions of all images $\mathbf{q}$ along the negative gradient direction with an increment parameter Δt as follows:

$$\mathbf{q}_i^{(l)} \longleftarrow \mathbf{q}_i^{(l)} - \frac{\mathbf{g}_i^{(l)}}{|\mathbf{g}^{(l)}|}\Delta t \tag{56}$$

  for all images l and for all atoms i, where $\mathbf{g}_i^{(l)}$ and $|\mathbf{g}^{(l)}|$ are defined in Eqs. (27) and (28), respectively, for the l-th image.
- Step 2: While keeping the positions of the terminal images $\mathbf{q}^{(1)}$ and $\mathbf{q}^{(P)}$ fixed, rearrange all other images along the path such that all neighboring images are equidistant:

$$\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\left(q_{i\alpha}^{(l+1)} - q_{i\alpha}^{(l)}\right)^2 = \text{const.} \tag{57}$$

If the number of images P is sufficiently large, the iteration of Steps 1 and 2 will converge to a minimum energy path that is closest to the initial guess.

13.8 OM action

The path-integral formulation of overdamped Langevin dynamics was introduced by Onsager and Machlup (OM). The transition probability from the initial position, $\mathbf{r}(0) = \mathbf{r}^{(1)}$, to the final position, $\mathbf{r}(T) = \mathbf{r}^{(L)}$, over a time interval of T is given by

$$P\left(\mathbf{r}^{(L)}\middle|\mathbf{r}^{(1)}\right) \propto \int_{\mathbf{r}^{(1)}}^{\mathbf{r}^{(L)}} \mathrm{D}\mathbf{r}\exp(-\beta S[\mathbf{r}]) \tag{58}$$

where the OM action is defined by

$$S[\mathbf{r}(t)] = \int_0^T dt\left\{\sum_{i=1}^{N}\frac{m_i\gamma}{4}\left(\dot{\mathbf{r}}_i(t) + \frac{\nabla_i V(\mathbf{r}(t))}{m_i\gamma}\right)^2\right\}. \tag{59}$$

Here, γ is the friction constant in Langevin dynamics. For a smooth path, one obtains

$$S \simeq \frac{V(\mathbf{r}(T)) - V(\mathbf{r}(0))}{2} + \int_0^T dt\ \sum_{i=1}^{N}\frac{m_i\gamma}{4}\left\{\left(\dot{\mathbf{r}}_i(t)\right)^2 + \left(\frac{\nabla_i V(\mathbf{r}(t))}{m_i\gamma}\right)^2\right\}. \tag{60}$$

A path $\mathbf{r}(t)$ that minimizes the action S is the most probable path. Such a path can be obtained by optimizing the value of S under the constraint that the path starts from a reactant minimum and ends at a product minimum. The path depends on the parameter T/γ. If T/γ is sufficiently small, the path becomes straight. If T/γ is sufficiently large, it becomes the minimum energy path, i.e., the intrinsic reaction coordinate.

In the PIMD code, the OM action with L discrete images,

$$S = \frac{V(\mathbf{r}^{(L)}) - V(\mathbf{r}^{(1)})}{2} + \sum_{l=1}^{L-1}\Delta t\sum_{i=1}^{N}\frac{m_i\gamma}{4}\left\{\left(\frac{\mathbf{r}_i^{(l+1)} - \mathbf{r}_i^{(l)}}{\Delta t}\right)^2 + \left(\frac{\nabla_i V(\mathbf{r}^{(l)})}{m_i\gamma}\right)^2\right\}, \tag{61}$$

is optimized using the limited-memory BFGS algorithm with respect to the normal-mode variables, where $\Delta t = T/(L-1)$. Note that S can be expressed as a function of $\Delta\tau = \Delta t/\gamma$ and $\mathbf{q}_i = \sqrt{m_i}\mathbf{r}_i$, since

$$S = \frac{\tilde{V}(\mathbf{q}^{(L)}) - \tilde{V}(\mathbf{q}^{(1)})}{2} + \sum_{l=1}^{L-1}\sum_{i=1}^{N}\left\{\frac{\left(\mathbf{q}_i^{(l+1)} - \mathbf{q}_i^{(l)}\right)^2}{4\Delta\tau} + \frac{\Delta\tau}{4}\left(\nabla_i\tilde{V}(\mathbf{q}^{(l)})\right)^2\right\}. \tag{62}$$

Here, $\tilde{V}(\mathbf{q}) = V(\mathbf{r})$. In the PIMD code, the gradient of S is evaluated numerically using the finite-difference technique.

13.9 NVE molecular dynamics

In NVE molecular dynamics, the microcanonical ensemble is naturally generated by integrating Newton's equations of motion

$$\dot{r}_{i\alpha} = \frac{p_{i\alpha}}{m_i}, \quad \dot{p}_{i\alpha} = -\frac{\partial V}{\partial r_{i\alpha}} \tag{63}$$

for ergodic systems. The total energy is given by

$$H = \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_i^2}{2m_i} + V \tag{64}$$

and is conserved, since

$$\begin{aligned} \dot{H} &= \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \left(\frac{\partial H}{\partial p_{i\alpha}} \dot{p}_{i\alpha} + \frac{\partial H}{\partial r_{i\alpha}} \dot{r}_{i\alpha} \right) \\ &= \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \left\{ \frac{p_{i\alpha}}{m_i} \left(-\frac{\partial V}{\partial r_{i\alpha}} \right) + \frac{\partial V}{\partial r_{i\alpha}} \frac{p_{i\alpha}}{m_i} \right\} = 0. \end{aligned} \tag{65}$$

The Jacobian, J, satisfies

$$\dot{J} = -J \left\{ \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \left(\frac{\partial \dot{r}_{i\alpha}}{\partial r_{i\alpha}} + \frac{\partial \dot{p}_{i\alpha}}{\partial p_{i\alpha}} \right) \right\} = 0, \tag{66}$$

and therefore J is constant. This result is known as Liouville's theorem. Since H is constant ($= E$), the phase-space volume of the trajectory can be written as

$$W \propto \int dr^{3N} \int dp^{3N} J \delta(H - E) \propto \int dr^{3N} \int dp^{3N} \delta(H - E), \tag{67}$$

assuming ergodicity. This corresponds to the number of microscopic states in the microcanonical ensemble. The probability density of finding the system at the point $(r_{1x}, \cdots, r_{Nz}, p_{1x}, \cdots, p_{Nz})$ is

$$P(r_{1x}, \cdots, r_{Nz}, p_{1x}, \cdots, p_{Nz}) = \frac{\delta(H - E)}{\int dr^{3N} \int dp^{3N} \delta(H - E)}. \tag{68}$$

13.9.1 Velocity-Verlet algorithm

The velocity-Verlet algorithm is used to integrate Newton's equations of motion,

$$\begin{aligned} p_{i\alpha}\left(t + \frac{\Delta t}{2}\right) &= p_{i\alpha}(t) - \frac{\partial V}{\partial r_{i\alpha}(t)} \frac{\Delta t}{2} \\ r_{i\alpha}(t + \Delta t) &= r_{i\alpha}(t) + \frac{p_{i\alpha}(t + \frac{\Delta t}{2})}{m_i} \Delta t \\ p_{i\alpha}(t + \Delta t) &= p_{i\alpha}\left(t + \frac{\Delta t}{2}\right) - \frac{\partial V}{\partial r_{i\alpha}(t + \Delta t)} \frac{\Delta t}{2}, \end{aligned} \tag{69}$$

or equivalently,

$$\begin{aligned} r_{i\alpha}(t + \Delta t) &= r_{i\alpha}(t) + \frac{p_{i\alpha}(t)}{m_i} \Delta t - \frac{1}{m_i} \frac{\partial V}{\partial r_{i\alpha}(t)} \frac{\Delta t^2}{2} \\ p_{i\alpha}(t + \Delta t) &= p_{i\alpha}(t) - \left(\frac{\partial V}{\partial r_{i\alpha}(t)} + \frac{\partial V}{\partial r_{i\alpha}(t + \Delta t)} \right) \frac{\Delta t}{2}. \end{aligned} \tag{70}$$

Note that this algorithm is time-reversible. Liouville's theorem is satisfied for any value of Δt.

13.10 NVT molecular dynamics

In NVT molecular dynamics, the equations of motion are designed to generate the canonical ensemble for ergodic systems. In the PIMD code, the temperature is controlled using a family of Nosé–Hoover thermostats. This technique is also used in path-integral molecular dynamics, centroid molecular dynamics, and metadynamics.

13.10.1 Nosé–Hoover method

In the Nosé–Hoover method [8, 10], N atoms interact with a thermostat. The equations of motion for the system of N atoms are

$$\begin{aligned} \dot{r}_{i\alpha} &= \frac{p_{i\alpha}}{m_i}, \\ \dot{p}_{i\alpha} &= -\frac{\partial V}{\partial r_{i\alpha}} - p_{i\alpha}\frac{p_\eta}{Q_\eta}, \end{aligned} \tag{71}$$

which are coupled to the equations of motion for the thermostat,

$$\begin{aligned} \dot{\eta} &= \frac{p_\eta}{Q_\eta}, \\ \dot{p}_\eta &= \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^2}{m_i} - 3NkT, \end{aligned} \tag{72}$$

where η and p_η are the thermostat coordinate and momentum, respectively, and Q_η is the thermostat mass. The total energy,

$$H = \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^2}{2m_i} + V + \frac{p_\eta^2}{2Q_\eta} + 3NkT\eta, \tag{73}$$

is conserved because

$$\begin{aligned} \dot{H} &= \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\left(\frac{\partial H}{\partial p_{i\alpha}}\dot{p}_{i\alpha} + \frac{\partial H}{\partial r_{i\alpha}}\dot{r}_{i\alpha}\right) + \frac{\partial H}{\partial p_\eta}\dot{p}_\eta + \frac{\partial H}{\partial \eta}\dot{\eta} \\ &= \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\left\{\frac{p_{i\alpha}}{m_i}\left(-\frac{\partial V}{\partial r_{i\alpha}} - p_{i\alpha}\frac{p_\eta}{Q_\eta}\right) + \frac{\partial V}{\partial r_{i\alpha}}\frac{p_{i\alpha}}{m_i}\right\} \\ &+ \frac{p_\eta}{Q_\eta}\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^2}{m_i} - 3NkT\right) + 3NkT\frac{p_\eta}{Q_\eta} = 0. \end{aligned} \tag{74}$$

It can be shown that the canonical ensemble is generated assuming ergodicity. Since H is a constant ($= E$), the phase-space volume of the combined system (atoms plus thermostat) can be written as

$$W \propto \int dr^{3N}\int dp^{3N}\int d\eta\int dp_\eta\, J\,\delta(H-E), \tag{75}$$

where the Jacobian J satisfies

$$\begin{aligned} \dot{J} &= -J\left\{\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\left(\frac{\partial \dot{r}_{i\alpha}}{\partial r_{i\alpha}} + \frac{\partial \dot{p}_{i\alpha}}{\partial p_{i\alpha}}\right) + \frac{\partial \dot{\eta}}{\partial \eta} + \frac{\partial \dot{p}_\eta}{\partial p_\eta}\right\} \\ &= J\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_\eta}{Q_\eta} = 3N\dot{\eta}\,J, \end{aligned} \tag{76}$$

and thus

$$J = \exp\left(3N\eta\right). \tag{77}$$

Now Eq. (75) can be rewritten as

$$\begin{aligned} W &\propto \int dr^{3N} \int dp^{3N} \int d\eta \int dp_\eta \exp\left(3N\eta\right) \\ &\times \delta\left(\sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_{i\alpha}^2}{2m_i} + V + \frac{p_\eta^2}{2Q_\eta} + 3NkT\eta - E\right). \end{aligned} \tag{78}$$

Using the identity

$$\delta[h(s)] = \frac{\delta\left(s - s_0\right)}{\left|h'(s_0)\right|} \quad \text{where} \quad h\left(s_0\right) = 0, \tag{79}$$

one obtains

$$\begin{aligned} W &\propto \frac{e^{\beta E}}{3NkT} \int dr^{3N} \int dp^{3N} \int dp_\eta \exp\left[-\beta\left(\sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_{i\alpha}^2}{2m_i} + V + \frac{p_\eta^2}{2Q_\eta}\right)\right] \\ &\propto \int dr^{3N} \int dp^{3N} \exp\left[-\beta\left(\sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_{i\alpha}^2}{2m_i} + V\right)\right] \propto Z, \end{aligned} \tag{80}$$

where $\beta = 1/kT$. This shows that W is proportional to the partition function Z of the canonical ensemble for the system of N atoms. In the same manner, it can be shown that the probability density of finding the system at the phase point $(r_{1x}, \cdots, r_{Nz}, p_{1x}, \cdots, p_{Nz})$ is identical to that of the canonical ensemble:

$$\begin{aligned} &P(r_{1x}, \cdots, r_{Nz}, p_{1x}, \cdots, p_{Nz}) \\ &= \frac{\exp\left[-\beta\left(\sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_{i\alpha}^2}{2m_i} + V\right)\right]}{\int dr^{3N} \int dp^{3N} \exp\left[-\beta\left(\sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_{i\alpha}^2}{2m_i} + V\right)\right]}. \end{aligned} \tag{81}$$

13.10.2 Nosé–Hoover chain method

In the Nosé–Hoover chain method [11], the system interacts with a chain of Nosé–Hoover thermostats to control the temperature more strongly. The system interacts with a thermostat which, in turn, interacts with the second thermostat, and so on. The equations of motion for the system of N atoms and a Nosé–Hoover thermostat chain of length L are expressed as follows:

$$\begin{aligned} \dot{r}_{i\alpha} &= \frac{p_{i\alpha}}{m_i} \\ \dot{p}_{i\alpha} &= -\frac{\partial V}{\partial r_{i\alpha}} - p_{i\alpha} \frac{p_{\eta,1}}{Q_{\eta,1}} \\ \dot{\eta}_l &= \frac{p_{\eta,l}}{Q_{\eta,l}} \\ \dot{p}_{\eta,1} &= \left(\sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_{i\alpha}^2}{m_i} - 3NkT\right) - p_{\eta,1} \frac{p_{\eta,2}}{Q_{\eta,2}} \\ \dot{p}_{\eta,l} &= \left(\frac{p_{\eta,l-1}^2}{Q_{\eta,l-1}} - kT\right) - p_{\eta,l} \frac{p_{\eta,l+1}}{Q_{\eta,l+1}} \quad (2 \le l \le L-1) \\ \dot{p}_{\eta,L} &= \frac{p_{\eta,L-1}^2}{Q_{\eta,L-1}} - kT, \end{aligned} \tag{82}$$

where η_l, $p_{\eta,l}$, and $Q_{\eta,l}$ are the coordinate, momentum, and mass of the lth thermostat in the chain, respectively. The total energy,

$$H = \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_{i\alpha}^2}{2m_i} + V + \frac{p_{\eta,1}^2}{2Q_{\eta,1}} + 3NkT\eta_1 + \sum_{l=2}^{L} \left(\frac{p_{\eta,l}^2}{2Q_{\eta,l}} + kT\eta_l\right), \tag{83}$$

is conserved, since

$$\begin{aligned}
\dot{H} &= \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\left(\frac{\partial H}{\partial p_{i\alpha}}\dot{p}_{i\alpha}+\frac{\partial H}{\partial r_{i\alpha}}\dot{r}_{i\alpha}\right)+\sum_{l=1}^{L}\frac{\partial H}{\partial p_{\eta,l}}\dot{p_{\eta,l}}+\sum_{l=1}^{L}\frac{\partial H}{\partial \eta_l}\dot{\eta}_l \\
&= \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\left\{\frac{p_{i\alpha}}{m_i}\left(-\frac{\partial V}{\partial r_{i\alpha}}-p_{i\alpha}\frac{p_{\eta,1}}{Q_{\eta,1}}\right)+\frac{\partial V}{\partial r_{i\alpha}}\frac{p_{i\alpha}}{m_i}\right\} \\
&+ \frac{p_{\eta,1}}{Q_{\eta,1}}\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^2}{m_i}-3NkT-p_{\eta,1}\frac{p_{\eta,2}}{Q_{\eta,2}}\right) \\
&+ \sum_{l=2}^{L-1}\left\{\frac{p_{\eta,l}}{Q_{\eta,l}}\left(\frac{p_{\eta,l-1}^2}{Q_{\eta,l-1}}-kT-p_{\eta,l}\frac{p_{\eta,l+1}}{Q_{\eta,l+1}}\right)\right\} \\
&+ \frac{p_{\eta,L}}{Q_{\eta,L}}\left(\frac{p_{\eta,L-1}^2}{Q_{\eta,L-1}}-kT\right)+3NkT\frac{p_{\eta,1}}{Q_{\eta,1}}+\sum_{l=2}^{L}kT\frac{p_{\eta,l}}{Q_{\eta,l}}=0. \qquad (84)
\end{aligned}$$

It can be shown that the canonical ensemble is generated assuming ergodicity. Since H is a constant ($= E$), the phase-space volume of the combined system (atoms plus thermostat chain) can be written as

$$W \propto \int dr^{3N}\int dp^{3N}\int d\eta^{L}\int dp_{\eta}^{L}\, J\,\delta(H-E), \qquad (85)$$

where the Jacobian J satisfies

$$\begin{aligned}
\dot{J} &= -J\left\{\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\left(\frac{\partial \dot{r}_{i\alpha}}{\partial r_{i\alpha}}+\frac{\partial \dot{p}_{i\alpha}}{\partial p_{i\alpha}}\right)+\sum_{l=1}^{L}\left(\frac{\partial \dot{\eta}_l}{\partial \eta_l}+\frac{\partial \dot{p}_{\eta,l}}{\partial p_{\eta,l}}\right)\right\} \\
&= J\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{\eta,1}}{Q_{\eta,1}}+\sum_{l=2}^{L}\frac{p_{\eta,l}}{Q_{\eta,l}}\right)=\left(3N\dot{\eta}_1+\sum_{l=2}^{L}\dot{\eta}_l\right)J, \qquad (86)
\end{aligned}$$

and thus

$$J = \exp\left(3N\eta_1+\sum_{l=2}^{L}\eta_l\right). \qquad (87)$$

Now Eq. (85) can be rewritten as

$$\begin{aligned}
W \propto & \int dr^{3N}\int dp^{3N}\int d\eta^{L}\int dp_{\eta}^{L}\exp\left(3N\eta_1+\sum_{l=2}^{L}\eta_l\right) \\
& \times\delta\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^2}{2m_i}+V+\frac{p_{\eta,1}^2}{2Q_{\eta,1}}+3NkT\eta_1+\sum_{l=2}^{L}\left(\frac{p_{\eta,l}^2}{2Q_{\eta,l}}+kT\eta_l\right)-E\right). \qquad (88)
\end{aligned}$$

Using the identity in Eq. (79), one obtains

$$\begin{aligned}
W &\propto \frac{e^{\beta E}}{3NkT}\int dr^{3N}\int dp^{3N}\int d\eta^{L-1}\int dp_{\eta}^{L} \\
&\quad \exp\left[-\beta\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^2}{2m_i}+V\right)\right]\exp\left(-\beta\sum_{l=1}^{L}\frac{p_{\eta,l}^2}{2Q_{\eta,l}}\right) \\
&\propto \int dr^{3N}\int dp^{3N}\exp\left[-\beta\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^2}{2m_i}+V\right)\right]\propto Z, \qquad (89)
\end{aligned}$$

where $\beta = 1/kT$. This shows that W is proportional to the partition function Z of the canonical ensemble for the system of N atoms. In the same manner, it can be shown that the probability density of finding the system at the phase point $(r_{1x},\cdots,r_{Nz},p_{1x},\cdots,p_{Nz})$ is identical to that of the canonical ensemble, Eq. (81).

13.10.3 Massive Nosé–Hoover chain method

In the massive Nosé–Hoover chain method, a Nosé–Hoover thermostat chain is attached to each of the $3N$ degrees of freedom. To control the temperature more strongly, multiple chains can be attached to each degree of freedom. In the latter case, the thermostat masses are chosen to control vibrational modes with different frequencies present in the system. In the PIMD code, the masses are chosen such that the characteristic frequencies of the thermostats form a Fourier series. The equations of motion for N nuclei coupled to a massive Nosé–Hoover chain thermostat, in which M chains of length L are attached to each degree of freedom, are

$$
\begin{aligned}
\dot{r}_{i\alpha} &= \frac{p_{i\alpha}}{m_i} \\
\dot{p}_{i\alpha} &= -\frac{\partial V}{\partial r_{i\alpha}} - p_{i\alpha} \sum_{m=1}^{M} \frac{p_{\eta,i\alpha,1,m}}{Q_{\eta,i\alpha,1,m}} \\
\dot{\eta}_{i\alpha,l,m} &= \frac{p_{\eta,i\alpha,l,m}}{Q_{\eta,i\alpha,l,m}} \\
\dot{p}_{\eta,i\alpha,1,m} &= \left(\frac{p_{i\alpha}^2}{m_i} - kT \right) - p_{\eta,i\alpha,1,m} \frac{p_{\eta,i\alpha,2,m}}{Q_{\eta,i\alpha,2,m}} \\
\dot{p}_{\eta,i\alpha,l,m} &= \left(\frac{p_{\eta,i\alpha,l-1,m}^2}{Q_{\eta,i\alpha,l-1,m}} - kT \right) - p_{\eta,i\alpha,l,m} \frac{p_{\eta,i\alpha,l+1,m}}{Q_{\eta,i\alpha,l+1,m}} \quad (2 \le l \le L-1) \\
\dot{p}_{\eta,i\alpha,L,m} &= \frac{p_{\eta,i\alpha,L-1,m}^2}{Q_{\eta,i\alpha,L-1,m}} - kT,
\end{aligned}
\tag{90}
$$

where $\eta_{i\alpha,l,m}$, $p_{\eta,i\alpha,l,m}$, and $Q_{\eta,i\alpha,l,m}$ are the coordinate, momentum, and mass of the lth thermostat of the mth chain attached to the α coordinate of the ith atom, respectively. The total energy,

$$
H = \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_{i\alpha}^2}{2m_i} + V + \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{l=1}^{L} \sum_{m=1}^{M} \left(\frac{p_{\eta,i\alpha,l,m}^2}{2Q_{\eta,i\alpha,l,m}} + kT\eta_{i\alpha,l,m} \right),
\tag{91}
$$

is conserved, since

$$
\begin{aligned}
\dot{H} &= \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \left(\frac{\partial H}{\partial p_{i\alpha}} \dot{p}_{i\alpha} + \frac{\partial H}{\partial r_{i\alpha}} \dot{r}_{i\alpha} \right) \\
&+ \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{l=1}^{L} \sum_{m=1}^{M} \frac{\partial H}{\partial p_{\eta,i\alpha,l,m}} \dot{p}_{\eta,i\alpha,l,m} + \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{l=1}^{L} \sum_{m=1}^{M} \frac{\partial H}{\partial \eta_{i\alpha,l,m}} \dot{\eta}_{i\alpha,l,m} \\
&= \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \left\{ \frac{p_{i\alpha}}{m_i} \left(-\frac{\partial V}{\partial r_{i\alpha}} - p_{i\alpha} \sum_{m=1}^{M} \frac{p_{\eta,i\alpha,1,m}}{Q_{\eta,i\alpha,1,m}} \right) + \frac{\partial V}{\partial r_{i\alpha}} \frac{p_{i\alpha}}{m_i} \right\} \\
&+ \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{m=1}^{M} \frac{p_{\eta,i\alpha,1,m}}{Q_{\eta,i\alpha,1,m}} \left(\frac{p_{i\alpha}^2}{m_i} - kT - p_{\eta,i\alpha,1,m} \frac{p_{\eta,i\alpha,2,m}}{Q_{\eta,i\alpha,2,m}} \right) \\
&+ \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{l=2}^{L-1} \sum_{m=1}^{M} \left\{ \frac{p_{\eta,i\alpha,l,m}}{Q_{\eta,i\alpha,l,m}} \left(\frac{p_{\eta,i\alpha,l-1,m}^2}{Q_{\eta,i\alpha,l-1,m}} - kT - p_{\eta,i\alpha,l,m} \frac{p_{\eta,i\alpha,l+1,m}}{Q_{\eta,i\alpha,l+1,m}} \right) \right\} \\
&+ \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{m=1}^{M} \frac{p_{\eta,i\alpha,L,m}}{Q_{\eta,i\alpha,L,m}} \left(\frac{p_{\eta,i\alpha,L-1,m}^2}{Q_{\eta,i\alpha,L-1,m}} - kT \right) \\
&+ \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{l=1}^{L} \sum_{m=1}^{M} kT \frac{p_{\eta,i\alpha,l,m}}{Q_{\eta,i\alpha,l,m}} = 0.
\end{aligned}
\tag{92}
$$

It can be shown that the canonical ensemble is generated assuming ergodicity. Since H is a constant ($= E$), the phase-space volume of the combined system (atoms plus thermostats) can be written as

$$
W \propto \int dr^{3N} \int dp^{3N} \int d\eta^{3NLM} \int dp_{\eta}^{3NLM} \, J \, \delta(H - E),
\tag{93}
$$

where the Jacobian J satisfies

$$\begin{aligned} \dot{J} &= -J\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\left\{\left(\frac{\partial \dot{r}_{i\alpha}}{\partial r_{i\alpha}}+\frac{\partial \dot{p}_{i\alpha}}{\partial p_{i\alpha}}\right)+\sum_{l=1}^{L}\sum_{m=1}^{M}\left(\frac{\partial \dot{\eta}_{i\alpha,l,m}}{\partial \eta_{i\alpha,l,m}}+\frac{\partial \dot{p}_{\eta,i\alpha,l,m}}{\partial p_{\eta,i\alpha,l,m}}\right)\right\} \\ &= J\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\sum_{l=1}^{L}\sum_{m=1}^{M}\frac{p_{\eta,i\alpha,l,m}}{Q_{\eta,i\alpha,l,m}}\right)=\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\sum_{l=1}^{L}\sum_{m=1}^{M}\dot{\eta}_{i\alpha,l,m}\right)J, \end{aligned} \tag{94}$$

and thus

$$J = \exp\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\sum_{l=1}^{L}\sum_{m=1}^{M}\eta_{i\alpha,l,m}\right). \tag{95}$$

Now Eq. (93) can be rewritten as

$$\begin{aligned} W \propto & \int dr^{3N}\int dp^{3N}\int d\eta^{3NLM}\int dp_{\eta}^{3NLM}\exp\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\sum_{l=1}^{L}\sum_{m=1}^{M}\eta_{i\alpha,l,m}\right) \\ & \times\delta\left[\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\left\{\frac{p_{i\alpha}^{2}}{2m_i}+\sum_{l=1}^{L}\sum_{m=1}^{M}\frac{p_{\eta,i\alpha,l,m}^{2}}{2Q_{\eta,i\alpha,l,m}}+kT\eta_{i\alpha,l,m}\right\}+V-E\right]. \end{aligned} \tag{96}$$

Using the identity in Eq. (79), one obtains

$$\begin{aligned} W &\propto \frac{e^{\beta E}}{kT}\int dr^{3N}\int dp^{3N}\int d\eta^{3NLM-1}\int dp_{\eta}^{3NLM} \\ &\times \exp\left[-\beta\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^{2}}{2m_i}+V+\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\sum_{l=1}^{L}\sum_{m=1}^{M}\frac{p_{\eta,i\alpha,l,m}^{2}}{2Q_{\eta,i\alpha,l,m}}\right)\right] \\ &\propto \int dr^{3N}\int dp^{3N}\exp\left[-\beta\left(\sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^{2}}{2m_i}+V\right)\right]\propto Z, \end{aligned} \tag{97}$$

where $\beta = 1/kT$. This shows that W is proportional to the partition function Z of the canonical ensemble for the system of N atoms. In the same manner, it can be shown that the probability density of finding the system at the phase point $(r_{1x},\cdots,r_{Nz},p_{1x},\cdots,p_{Nz})$ is identical to that of the canonical ensemble, Eq. (81).

13.11 Path integral methods

In PI methods, each atom is represented by P replicas in imaginary time slices. The parameter P is known as the number of beads. The equation of motion of a $3NP$-dimensional system is designed such that the distribution is proportional to $\exp\left(-\beta V^{\text{eff}}\right)$, where the effective potential V^{eff} is given by:

$$V^{\text{eff}} = V^{\text{harm}}+V^{\text{phys}}+V^{\text{cor}}. \tag{98}$$

where the first and second terms are the contributions of the harmonic and physical potentials, respectively:

$$V^{\text{harm}} \equiv \sum_{l=1}^{P}\sum_{I=1}^{N}\sum_{\alpha=x,y,z}\left\{\frac{m_i\omega_P^2}{2}\left(r_{i\alpha}^{(l)}-r_{i\alpha}^{(l+1)}\right)^2\right\} \tag{99}$$

$$V^{\text{phys}} \equiv \frac{1}{P}\sum_{l=1}^{P}V\left(r_{1x}^{(l)},\cdots,r_{Nz}^{(l)}\right), \tag{100}$$

The periodicity $r_{i\alpha}^{(P+1)} = r_{i\alpha}^{(1)}$ is assumed, and the constant $\omega_P = \sqrt{P}/\beta\hbar$ is defined. The third term is the correction for the fourth order Trotter expansion:

$$V^{\text{cor}} \equiv \frac{\beta^2\hbar^2}{24P^3}\sum_{i=1}^{N}\frac{1}{m_i}\sum_{l=1}^{P}\left\{\left(\frac{\partial V}{\partial r_{ix}^{(l)}}\right)^2+\left(\frac{\partial V}{\partial r_{iy}^{(l)}}\right)^2+\left(\frac{\partial V}{\partial r_{iz}^{(l)}}\right)^2\right\}, \tag{101}$$

which is neglected in the second order Trotter expansion:

$$V^{\rm cor} = 0 \tag{102}$$

Note that the first term on the rhs of Eq.(98) connects replicas into a cyclic chain by harmonic interaction. It is useful to introduce a linear transformation from Cartesian coordinates $r_{i\alpha}^{(l)}$ to so-called normal mode coordinates $q_{i\alpha}^{(k)}$:

$$q_{i\alpha}^{(k)} = \frac{1}{\sqrt{P}} \sum_{l=1}^{P} U_{kl} r_{i\alpha}^{(l)}. \tag{103}$$

The unitary matrix U_{kl} corresponds to the eigenvector of the matrix $\overleftrightarrow{\mathbf{A}}$, where $A_{kl} = P\left(2\delta_{kl} - \delta_{k,l+1} - \delta_{k,l-1}\right)$ with periodicity $0 \to P$ and $P+1 \to 1$. The first mode with $k = 1$, known as the centroid variable, is given by

$$q_{i\alpha}^{(1)} = q_{i\alpha}^{(\rm cent)} = \frac{1}{P} \sum_{l=1}^{P} r_{i\alpha}^{(l)}. \tag{104}$$

After this transformation, we obtain

$$V^{\rm eff} = \sum_{k=1}^{P} \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \left\{ \frac{m_i \lambda^{(k)} \omega_P^2}{2} q_{i\alpha}^{(k)2} \right\} + \frac{1}{P} \sum_{l=1}^{P} V(r_{1x}^{(l)}, \cdots, r_{Nz}^{(l)}), \tag{105}$$

where $\lambda^{(k)}$ is the k-th eigenvalue of the matrix $\overleftrightarrow{\mathbf{A}}$, and $r_{i\alpha}^{(l)}$ is given as a function of $q_{i\alpha}^{(k)}$ by the backward transformation

$$r_{i\alpha}^{(l)} = \sqrt{P} \sum_{k=1}^{P} U_{lk}^{\dagger} q_{i\alpha}^{(k)}. \tag{106}$$

The force is given by

$$F_{i\alpha}^{(k)} = -\frac{\partial V^{\rm eff}}{\partial q_{i\alpha}^{(k)}} = -m_i \lambda^{(k)} \omega_P^2 q_{i\alpha}^{(k)} - \frac{1}{\sqrt{P}} \sum_{l=1}^{P} U_{lk}^{\dagger} \frac{\partial V}{\partial r_{i\alpha}^{(l)}}. \tag{107}$$

We can construct the effective Hamiltonian as

$$H^{\rm eff} = \sum_{k=1}^{P} \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \frac{p_{i\alpha}^{(k)2}}{2\mu_i^{(k)}} + V^{\rm eff} + H^{\rm bath} + H^{\rm baro}, \tag{108}$$

where $p_{i\alpha}^{(k)}$ is the momentum conjugate to $q_{i\alpha}^{(k)}$, $\mu_i^{(k)}$ is the fictitious mass, $H^{\rm bath}$ and $H^{\rm baro}$ are the energy contributions of the thermostats and barostats, respectively.

In PIMD, BCMD, CMD and (T)RPMD methods, the equations of motion are integrated with a finite time step Δt. In PIHMC, after the equations of motion are integrated for a few steps, the last move is either accepted or rejected according to the Metropolis algorithm with the acceptance probability $\min(\exp(-\Delta H^{\rm eff}), 1)$. When rejected, all the positions (atomic positions in the NVT ensemble, the atomic positions and the box vectors in NPT and NtT ensembles) are placed back, and all the velocities (the atomic velocities in the NVT ensemble, the atomic velocities and the box velocities in the NPT and NtT ensembles) are regenerated randomly according to the Maxwell-Boltzmann distribution. The step size Δt in PIHMC is usually taken to be larger than that of PIMD.

In BCMD and TRPMD, the deterministic equation of motion for the centroid variable is combined with the stochastic equation of motion for the non-centroid variables.

PIMD, PIHMC, BCMD, CMD and (T)RPMD methods are different in the the settings of the fictitious mass and the way of using the thermostats and the barostats, which are mentioned below.

13.11.1 Fictitious mass

The fictitious mass is set differently between the centroid modes ($k = 1$) and the non-centroid modes ($k \neq 1$).

PIMD and PIHMC..

$$\mu_i^{(1)} = \mu_i^{(\mathrm{cent})} = m_i, \quad \mu_i^{(k)} = \lambda^{(k)} m_i \ \ (2 \le k \le P). \tag{109}$$

BCMD..

$$\mu_i^{(1)} = \mu_i^{(\mathrm{cent})} = m_i, \quad \mu_i^{(k)} = \frac{P(\Delta t)}{2\tau} \lambda^{(k)} m_i \ \ (2 \le k \le P). \tag{110}$$

where Δt is the step size and $\tau = \beta\hbar$ is the decoherence time.

CMD..

$$\mu_i^{(1)} = \mu_i^{(\mathrm{cent})} = m_i, \quad \mu_i^{(k)} = \gamma^2 \lambda^{(k)} m_i \ \ (2 \le k \le P). \tag{111}$$

Theoretically, the adiabaticity parameter γ should be taken sufficiently small, $0 < \gamma \ll 1$, in CMD. However, the step size should be set smaller as γ becomes smaller, so a good compromise in the choice of γ is needed.

RPMD and TRPMD..

$$\mu_i^{(k)} = m_i \quad (1 \le k \le P). \tag{112}$$

13.11.2 Thermostats

PIMD for NVT ensemble.. In the present implementation of PIMD, the massive Nosé-Hoover chain thermostats are attached to both the centroid modes ($3N$) and the non-centroid modes ($3NP - 3N$). The energy of the thermostats is given by

$$\begin{aligned} H_{\mathrm{bath}} &= \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{l=1}^{L} \sum_{m=1}^{M} \left(\frac{{p_{\eta,i\alpha,l,m}^{(\mathrm{cent})}}^2}{2Q_{\eta,i\alpha,l,m}^{(\mathrm{cent})}} + kT\eta_{i\alpha,l,m}^{(\mathrm{cent})} \right) \\ &+ \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{k=2}^{P} \sum_{l=1}^{L} \left(\frac{{p_{\eta,i\alpha,l}^{(k)}}^2}{2Q_{\eta,i\alpha,l}^{(k)}} + kT\eta_{i\alpha,l}^{(k)} \right). \end{aligned} \tag{113}$$

The M thermostat chains with the length L are multiply attached to each centroid degree of freedom. A thermostat chain with the length L is attached to each non-centroid degree of freedom. The mass of the centroid thermostats is chosen to be

$$Q_{\eta,i\alpha,l,m}^{(\mathrm{cent})} = kT \left(M^{m-1} \frac{\tau_{\mathrm{bath}}}{2\pi} \right)^2, \tag{114}$$

where τ_{bath} is typically chosen to be the period of (the highest) vibrational frequency present in the system. The mass of the non-centroid thermostats is chosen to be

$$Q_{\eta,i\alpha,l}^{(k)} = \frac{kT}{\omega_P^2}. \tag{115}$$

PIMD for NPT and NtT ensembles with a cubic box.. The energy of the thermostats is given by

$$\begin{aligned} H_{\mathrm{bath}} &= \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{l=1}^{L} \sum_{m=1}^{M} \left(\frac{{p_{\eta,i\alpha,l,m}^{(\mathrm{cent})}}^2}{2Q_{\eta,i\alpha,l,m}^{(\mathrm{cent})}} + kT\eta_{i\alpha,l,m}^{(\mathrm{cent})} \right) \\ &+ \sum_{i=1}^{N} \sum_{\alpha=x,y,z} \sum_{k=2}^{P} \sum_{l=1}^{L} \left(\frac{{p_{\eta,i\alpha,l}^{(k)}}^2}{2Q_{\eta,i\alpha,l}^{(k)}} + kT\eta_{i\alpha,l}^{(k)} \right) \\ &+ \sum_{l=1}^{L} \sum_{m=1}^{M} \left(\frac{{p_{\eta,l,m}^{\mathrm{b}}}^2}{2Q_{\eta,l,m}^{\mathrm{b}}} + kT\eta_{l,m}^{\mathrm{b}} \right). \end{aligned} \tag{116}$$

The mass of the centroid thermostats is chosen to be

$$Q^{(\mathrm{cent})}_{\eta,i\alpha,l,m} = kT\left(M^{m-1}\frac{\tau_{\mathrm{bath}}}{2\pi}\right)^2, \tag{117}$$

where τ_{bath} is typically chosen to be the period of (the highest) vibrational frequency present in the system. The mass of the non-centroid thermostats is chosen to be

$$Q^{(k)}_{\eta,i\alpha,l} = \frac{kT}{\omega_P^2}. \tag{118}$$

The mass of the box thermostats is chosen to be

$$Q^{\mathrm{b}}_{\eta,l,m} = kT\left(M^{m-1}\frac{\tau_{\mathrm{baro}}}{2\pi}\right)^2. \tag{119}$$

PIMD for NPT and NtT ensembles with a hexahedron box.. The energy of the thermostats is given by

$$\begin{aligned} H_{\mathrm{bath}} = & \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\sum_{l=1}^{L}\sum_{m=1}^{M}\left(\frac{{p^{(\mathrm{cent})}_{\eta,i\alpha,l,m}}^2}{2Q^{(\mathrm{cent})}_{\eta,i\alpha,l,m}} + kT\eta^{(\mathrm{cent})}_{i\alpha,l,m}\right) \\ & + \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\sum_{k=2}^{P}\sum_{l=1}^{L}\left(\frac{{p^{(k)}_{\eta,i\alpha,l}}^2}{2Q^{(k)}_{\eta,i\alpha,l}} + kT\eta^{(k)}_{i\alpha,l}\right) \\ & + \sum_{\alpha=x,y,z}\sum_{\beta=x,y,z}\sum_{l=1}^{L}\sum_{m=1}^{M}\left(\frac{{p^{\mathrm{b}}_{\eta,\alpha\beta,l,m}}^2}{2Q^{\mathrm{b}}_{\eta,\alpha\beta,l,m}} + kT\eta^{\mathrm{b}}_{\alpha\beta,l,m}\right). \end{aligned} \tag{120}$$

The mass of the centroid thermostats is chosen to be

$$Q^{(\mathrm{cent})}_{\eta,i\alpha,l,m} = kT\left(M^{m-1}\frac{\tau_{\mathrm{bath}}}{2\pi}\right)^2, \tag{121}$$

where τ_{bath} is typically chosen to be the period of (the highest) vibrational frequency present in the system. The mass of the non-centroid thermostats is chosen to be

$$Q^{(k)}_{\eta,i\alpha,l} = \frac{kT}{\omega_P^2}. \tag{122}$$

The mass of the box thermostats is chosen to be

$$Q^{\mathrm{b}}_{\eta,\alpha\beta,1,m} = kT\left(6M^{m-1}\frac{\tau_{\mathrm{baro}}}{2\pi}\right)^2 \tag{123}$$

and

$$Q^{\mathrm{b}}_{\eta,\alpha\beta,l,m} = kT\left(M^{m-1}\frac{\tau_{\mathrm{baro}}}{2\pi}\right)^2 \tag{124}$$

for $l \neq 1$.

BCMD.. Non-centroid velocities are randomized at each step (i.e., an Andersen thermostat is attached to the non-centroid modes). The energy is conserved by keeping track of the jumps due to the randomization.

CMD.. In the present implementation of CMD, the massive Nosé-Hoover chain thermostats are attached only to the non-centroid modes ($3NP - 3N$). The energy of the thermostats is given by

$$H_{\mathrm{bath}} = \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\sum_{k=2}^{P}\sum_{l=1}^{L}\left(\frac{{p^{(k)}_{\eta,i\alpha,l}}^2}{2Q^{(k)}_{\eta,i\alpha,l}} + kT\eta^{(k)}_{i\alpha,l}\right). \tag{125}$$

A thermostat chain with length L is attached to each non-centroid degree of freedom. The mass of the non-centroid thermostats is chosen to be

$$Q^{(k)}_{\eta,i\alpha,l} = \frac{kT}{\omega_P^2}\gamma^2. \tag{126}$$

RPMD and TRPMD.. A thermostat is not used in the RPMD method. Thus, the energy of the thermostats is zero,

$$H_{\text{bath}} = 0. \tag{127}$$

In TRPMD, a Langevin thermostat is attached to the non-centroid modes, and thus the energy is not conserved. For an internal normal mode $k > 0$, PIMD 2.8.0 implements the Rossi–Ceriotti–Manolopoulos convention

$$\gamma_k = 2\lambda\widetilde{\omega}_k, \tag{128}$$

where $\lambda =$ <scale_trpmd>. The default $\lambda = 1/2$ is the near-optimal damping adopted in Ref. [34]; $\lambda = 1$ gives critical damping of the free-ring-polymer mode and $\lambda = 0$ recovers RPMD.

13.11.3 Barostats

PIMD and PIHMC for NVT ensemble.. The barostat is not used in the PIMD and PIHMC methods for the NVT ensemble since the volume is fixed. Thus, the energy of the barostats is zero,

$$H_{\text{baro}} = 0. \tag{129}$$

PIMD and PIHMC for NPT ensemble with a cubic box.. The energy of the barostat is given by

$$H_{\text{baro}} = \frac{{P^{\text{b}}}^2}{2Q^{\text{b}}} + P_{\text{ext}}V_{\text{cell}}, \tag{130}$$

where P^{b} is the box velocity conjugate to the volume V_{cell}. The box mass is chosen to be

$$Q^{\text{b}} = 3(N+1)kT\left(\frac{\tau_{\text{baro}}}{2\pi}\right)^2. \tag{131}$$

PIMD and PIHMC for NPT ensemble with a hexahedron box.. The energy of the barostats is given by

$$H_{\text{baro}} = \sum_{\alpha,\beta=x,y,z} \frac{{P^{\text{b}}_{\alpha\beta}}^2}{2Q^{\text{b}}_{\alpha\beta}} + P_{\text{ext}}V_{\text{cell}}. \tag{132}$$

where $P^{\text{b}}_{\alpha\beta}$ is the box velocity conjugate to the box matrix $h_{\alpha\beta}$. The box masses are chosen to be

$$Q^{\text{b}}_{\alpha\beta} = (N+1)kT\left(\frac{\tau_{\text{baro}}}{2\pi}\right)^2. \tag{133}$$

PIMD and PIHMC for NtT ensemble with a hexahedron box.. The energy of the barostats is given by

$$H_{\text{baro}} = \sum_{\alpha,\beta=x,y,z}\left(\frac{{P^{\text{b}}_{\alpha\beta}}^2}{2Q^{\text{b}}_{\alpha\beta}} - t_{\alpha\beta}\epsilon_{\alpha\beta}\bar{V}_{\text{cell}}\right), \tag{134}$$

The strain tensor is defined by

$$\epsilon_{\alpha\beta} = \frac{1}{2}\left(\sum_{\nu,\gamma,\lambda=x,y,z}\bar{h}^{-1}_{\gamma\alpha}h_{\lambda\gamma}h_{\lambda\nu}\bar{h}^{-1}_{\nu\beta} - \delta_{\alpha\beta}\right) \tag{135}$$

where $\overleftrightarrow{\mathbf{h}}$ is the box matrix under zero tension. The box masses are chosen to be

$$Q^{\text{b}}_{\alpha\beta} = (N+1)kT\left(\frac{\tau_{\text{baro}}}{2\pi}\right)^2. \tag{136}$$

Meanwhile, the stress tensor is given by

$$\sigma_{\alpha\beta} = \frac{\bar{V}_{\text{cell}}}{V_{\text{cell}}}P_{\alpha\beta} \tag{137}$$

where the pressure tensor is defined as

$$P_{\alpha\beta} = -\sum_{\gamma,\nu,\mu,\lambda=x,y,z} h_{\alpha\gamma}\bar{h}^{-1}_{\gamma\mu}t_{\mu\lambda}\bar{h}^{-1}_{\nu\lambda}h_{\beta\nu}. \tag{138}$$

CMD and RPMD.. A barostat is not used in the CMD and RPMD methods since the volume is fixed. Thus, the energy of the barostat is zero,

$$H_{\mathrm{baro}} = 0. \tag{139}$$

13.11.4 How to set up RPMD and CMD

In the RPMD and CMD calculations, the initial positions and the initial velocities could be taken from the previous PIMD calculation [31]. In such cases, since the fictitious masses of RPMD and CMD are different from those of PIMD, the initial velocities must be scaled appropriately such that the temperature is kept the same. This can be done by simply scaling the normal mode velocities for all the modes k,

$$\dot{q}_k^{\mathrm{X}} = \sqrt{\frac{\mu_k^{\mathrm{PIMD}}}{\mu_k^{\mathrm{X}}}}\dot{q}_k^{\mathrm{PIMD}}. \tag{140}$$

where X is either RPMD or CMD. In CMD, the initial positions and the initial velocities of the massive thermostats attached to the non-centroid modes can also be taken from the previous PIMD calculation. As the masses of these thermostats in CMD are different from those of PIMD by the factor $1/\gamma^2$, all the initial velocities of the thermostats must be scaled by $1/\gamma$ in CMD. All the above velocity scaling is automatically done in the code.

13.11.5 Dual-level hybrid Monte Carlo

In the dual-level HMC method, the low-level force, $-\nabla V^{\mathrm{lo}}$, is used for the trial move. Meanwhile the high-level potential, V^{hi}, is used for the Metropolis acceptance probability, i.e., $P = \min\left[1, \exp\left(-\beta H^{\mathrm{hi}}\right)\right]$. In this way, the computation of the high-level force, $-\nabla V^{\mathrm{hi}}$, can be avoided. This method works well if the difference between ΔV^{hi} and ΔV^{lo} in each trial move is marginal compared to the thermal energy, $k_{\mathrm{B}}T$.

13.11.6 Fourth order path integral hybrid Monte Carlo

In the fourth order PIHMC method, the force without the correction, $-\nabla\left(V^{\mathrm{harm}} + V^{\mathrm{phys}}\right)$, is used for the trial move. Meanwhile, the full effective potential, V^{eff}, is used for the Metropolis acceptance probability, i.e., $P = \min\left[1, \exp\left(-\beta H^{\mathrm{eff}}\right)\right]$. In this way, the computation of the Hessian matrix, $\nabla^2 V$, can be avoided. This method works well if the change of the fourth order correction, ΔV^{cor}, in each trial move is marginal compared to the thermal energy, $k_{\mathrm{B}}T$.

13.11.7 Translational and rotational corrections

In order to obtain the vibrational spectra of isolated molecules from RPMD and CMD calculations, it would be better to eliminate the overall translation and the overall rotation of the system [31]. To do this, the correction to the centroid velocity should be applied at the initial step. In addition, the correction to the centroid force should be applied at every step during the run. Note, however, that as a trade-off of applying these translational and rotational corrections, the energy of the extended system will not be strictly conserved. For convenience, the origin of coordinates is chosen to be the center of mass of the whole system.

correction of centroid velocity.. First, the centroid velocity vector of i-th atom is

$$\dot{\mathbf{q}}_i^{(\mathrm{cent})} = \frac{1}{P}\sum_{k=1}^{P}\dot{\mathbf{r}}_i^{(k)}. \tag{141}$$

The translational correction is applied using

$$\dot{\mathbf{q}}_i^{(\mathrm{cent})} \longleftarrow \dot{\mathbf{q}}_i^{(\mathrm{cent})} - \Delta\dot{\mathbf{q}}'. \tag{142}$$

The correction term is calculated as the center-of-mass velocity of the whole system,

$$\Delta\dot{\mathbf{q}}' = \frac{\sum_{j=1}^{3N}\mu_j^{(\mathrm{cent})}\dot{\mathbf{q}}_j^{(\mathrm{cent})}}{\sum_{j=1}^{3N}\mu_j^{(\mathrm{cent})}}. \tag{143}$$

Next, the rotational correction is applied using

$$\dot{\mathbf{q}}_i^{(\mathrm{cent})} \longleftarrow \dot{\mathbf{q}}_i^{(\mathrm{cent})} - \Delta\dot{\mathbf{q}}_i''. \tag{144}$$

The correction term is calculated as the contribution of velocity to the angular momentum of the whole system,

$$\Delta\dot{\mathbf{q}}_i'' = \boldsymbol{\theta} \times \mathbf{q}_i^{(\mathrm{cent})}, \tag{145}$$

where the centroid angular velocity vector is defined as

$$\boldsymbol{\theta} = \overleftrightarrow{\mathbf{I}}^{(\mathrm{cent})^{-1}} \mathbf{L}^{(\mathrm{cent})}, \tag{146}$$

the centroid angular momentum vector as

$$\mathbf{L}^{(\mathrm{cent})} = \sum_{i=1}^{N} \left(\mathbf{q}_i^{(\mathrm{cent})} \times \mu_i^{(\mathrm{cent})} \dot{\mathbf{q}}_i^{(\mathrm{cent})} \right), \tag{147}$$

and the centroid moment of inertia tensor as

$$I_{xx}^{(\mathrm{cent})} = \sum_{i=1}^{N} \mu_i^{(\mathrm{cent})} (q_{iy}^{(\mathrm{cent})} q_{iy}^{(\mathrm{cent})} + q_{iz}^{(\mathrm{cent})} q_{iz}^{(\mathrm{cent})}), \quad I_{xy}^{(\mathrm{cent})} = -\sum_{i=1}^{N} \mu_i^{(\mathrm{cent})} q_{ix}^{(\mathrm{cent})} q_{iy}^{(\mathrm{cent})}, \quad \text{etc.} \tag{148}$$

correction of centroid force.. The centroid force of i-th atom is

$$\dot{\mathbf{F}}_i^{(\mathrm{cent})} = \frac{1}{P} \sum_{k=1}^{P} \dot{\mathbf{F}}_i^{(k)}. \tag{149}$$

The translational correction is applied using

$$\mathbf{F}_i^{(\mathrm{cent})} \longleftarrow \mathbf{F}_i^{(\mathrm{cent})} - \Delta\mathbf{F}_i'. \tag{150}$$

The correction term is calculated as the force applied to the whole system,

$$\Delta\mathbf{F}_i' = \frac{1}{N} \sum_{i=1}^{N} \mathbf{F}_i^{(\mathrm{cent})}. \tag{151}$$

Next, the rotational correction is applied using

$$\mathbf{F}_i^{(\mathrm{cent})} \longleftarrow \mathbf{F}_i^{(\mathrm{cent})} - \Delta\mathbf{F}_i''. \tag{152}$$

The correction term is calculated as the contribution of force to torque applied to the whole system,

$$\Delta\mathbf{F}_i'' = \mu_i^{(\mathrm{cent})} \dot{\boldsymbol{\omega}} \times \mathbf{r}_i^{(\mathrm{cent})}. \tag{153}$$

Here the time derivative of centroid angular momentum,

$$\dot{\boldsymbol{\omega}} = \overleftrightarrow{\mathbf{I}}^{(\mathrm{cent})^{-1}} \mathbf{N}^{(\mathrm{cent})}, \tag{154}$$

has been introduced using the centroid torque

$$\mathbf{N}^{(\mathrm{cent})} = \sum_{i=1}^{N} \mathbf{r}_i^{(\mathrm{cent})} \times \mathbf{F}_i^{(\mathrm{cent})}. \tag{155}$$

For the special case of diatomic molecules, the component parallel to the bond in the $\dot{\boldsymbol{\omega}}$ vector is eliminated to avoid the redundant rotation.

13.12 Replica exchange hybrid Monte Carlo

13.12.1 Temperature replica exchange

Consider P replicas of a system of N atoms with different temperatures. If the replicas do not interact with each other, the s-th replica constitutes the canonical ensemble at temperature $T^{(s)}$. The whole system of all replicas combined together will then constitute the ensemble corresponding to the probability density

$$\rho\left(\mathbf{\Gamma}^{(1)},\cdots,\mathbf{\Gamma}^{(P)}\right)=\rho\left(\mathbf{\Gamma}^{(1)}\right)\times\cdots\times\rho\left(\mathbf{\Gamma}^{(P)}\right)\propto\exp\left\{-\sum_{s=1}^{P}\beta^{(s)}H\left(\mathbf{\Gamma}^{(s)}\right)\right\} \tag{156}$$

where $\beta^{(s)}=\frac{1}{k_{\mathrm{B}}T^{(s)}}$, and the temperature of the s-th replica is set to

$$T^{(s)}=T^{(1)}\left(\frac{T^{(P)}}{T^{(1)}}\right)^{\frac{s-1}{P-1}}. \tag{157}$$

This probability density has the following symmetry under the exchange of replicas s and t:

$$\rho\left(\cdots,\mathbf{\Gamma}^{(t)},\cdots,\mathbf{\Gamma}^{(s)},\cdots\right)=\rho\left(\cdots,\mathbf{\Gamma}^{(s)},\cdots,\mathbf{\Gamma}^{(t)},\cdots\right)\exp\left\{\left(\beta^{(s)}-\beta^{(t)}\right)\left(H\left(\mathbf{\Gamma}^{(s)}\right)-H\left(\mathbf{\Gamma}^{(t)}\right)\right)\right\}. \tag{158}$$

Therefore, according to the detailed balance condition, this ensemble is not changed if the configurations are exchanged at random, with the Metropolis acceptance probability given by

$$P\left(\cdots,\mathbf{\Gamma}^{(s)},\cdots,\mathbf{\Gamma}^{(t)},\cdots\rightarrow\cdots,\mathbf{\Gamma}^{(t)},\cdots,\mathbf{\Gamma}^{(s)},\cdots\right)=\min(1,\exp(-\Delta)) \tag{159}$$

where

$$\Delta=-\left(\beta^{(s)}-\beta^{(t)}\right)\left(H\left(\mathbf{\Gamma}^{(s)}\right)-H\left(\mathbf{\Gamma}^{(t)}\right)\right). \tag{160}$$

In the REHMC method, this exchange attempt is performed at the Monte Carlo stage, which occurs every few steps of the integration of the equations of motion.

13.12.2 Hamiltonian replica exchange

Consider P replicas of a system of N atoms with different potentials, each describing an intermediate alchemical stage between the physical systems A and B, with energies H_{A} and H_{B}, respectively. In this case, the energy of the s-th intermediate alchemical stage for the t-th replica is given by

$$\tilde{H}^{(s)}(\mathbf{\Gamma}^{(t)}) \quad = \quad \left(\frac{P-s}{P-1}\right)H_{\mathrm{A}}(\mathbf{\Gamma}^{(t)})+\left(\frac{s-1}{P-1}\right)H_{\mathrm{B}}(\mathbf{\Gamma}^{(t)}). \tag{161}$$

If the replicas do not interact with each other, the s-th replica constitutes the canonical ensemble of the corresponding system. The whole system of all replicas combined together will then constitute the ensemble corresponding to the probability density

$$\rho\left(\mathbf{\Gamma}^{(1)},\cdots,\mathbf{\Gamma}^{(P)}\right)=\rho\left(\mathbf{\Gamma}^{(1)}\right)\times\cdots\times\rho\left(\mathbf{\Gamma}^{(P)}\right)\propto\exp\left\{-\beta\sum_{s=1}^{P}\tilde{H}_{s}\left(\mathbf{\Gamma}_{s}\right)\right\} \tag{162}$$

where $\beta=\frac{1}{k_{\mathrm{B}}T}$. Upon exchanging replicas s and t, the Metropolis acceptance probability that preserves this distribution is

$$P\left(\cdots,\mathbf{\Gamma}^{(s)},\cdots,\mathbf{\Gamma}^{(t)},\cdots\rightarrow\cdots,\mathbf{\Gamma}^{(t)},\cdots,\mathbf{\Gamma}^{(s)},\cdots\right) \quad = \quad \min(1,\exp(-\Delta)) \tag{163}$$

where

$$\Delta \quad = \quad \beta\left\{\tilde{H}^{(t)}(\mathbf{\Gamma}^{(s)})+\tilde{H}^{(s)}(\mathbf{\Gamma}^{(t)})-\tilde{H}^{(s)}(\mathbf{\Gamma}^{(s)})-\tilde{H}^{(t)}(\mathbf{\Gamma}^{(t)})\right\}. \tag{164}$$

In the REHMC method, this exchange attempt is performed at the Monte Carlo stage, which occurs every few steps of the integration of the equations of motion.

13.13 Metadynamics

The metadynamics method was proposed by Laio and Parrinello [36]. We use ξ_d and π_d for the coordinate and conjugate momentum of a fictitious collective-variable particle, μ_d for its fictitious mass, κ_d for its harmonic coupling constant, and h and σ_d for the height and width of a metadynamics hill. In the PIMD code, the implementation follows closely the version described in reference [38]. The total energy is expressed as

$$\begin{aligned} H &= \sum_{l=1}^{P}\left[H_{\rm sys}^{(l)}\left(\{\mathbf{r}^{(l)},\mathbf{p}^{(l)}\}\right)+T_{\rm cv}^{(l)}\left(\{\boldsymbol{\pi}^{(l)}\}\right)+V_{\rm cv}^{(l)}\left(\{\boldsymbol{\xi}^{(l)},\mathbf{r}^{(l)}\}\right)\right] \\ &+ V^{\rm hills}\left(\{\boldsymbol{\xi}^{(1)},\cdots,\boldsymbol{\xi}^{(P)},\mathbf{r}^{(1)},\cdots,\mathbf{r}^{(P)}\}\right)+V^{\rm cor}+H^{\rm bath}. \end{aligned} \tag{165}$$

Here, $\{\mathbf{r}^{(l)}\}$ and $\{\mathbf{p}^{(l)}\}$ denote the sets of $3N$-dimensional Cartesian coordinates and momenta of the l-th walker, while $\{\boldsymbol{\xi}^{(l)}\}=\left(\xi_1^{(l)},\cdots,\xi_D^{(l)}\right)$ and $\{\boldsymbol{\pi}^{(l)}\}=\left(\pi_1^{(l)},\cdots,\pi_D^{(l)}\right)$ are the sets of D-dimensional coordinates and momenta of the fictitious particle for the l-th walker. The energy of the l-th walker is given by

$$H_{\rm sys}^{(l)} = \sum_{i=1}^{N}\sum_{\alpha=x,y,z}\frac{p_{i\alpha}^{(l)2}}{2m_i}+V\left(\mathbf{r}^{(l)}\right). \tag{166}$$

The kinetic energy of the fictitious particle is

$$T_{\rm cv}^{(l)} = \sum_{d=1}^{D}\frac{\pi_d^{(l)2}}{2\mu_d}, \tag{167}$$

where μ_d is the mass of the d-th component of the fictitious particle. The harmonic potential between the fictitious particle $\xi_d^{(l)}$ and the collective variable $R_d^{(l)}$ is

$$V_{\rm cv}^{(l)} = \sum_{d=1}^{D}\frac{\kappa_d}{2}\left\{\xi_d^{(l)}-R_d^{(l)}\left(\mathbf{r}^{(l)}\right)\right\}^2, \tag{168}$$

where κ_d is the force constant and $R_d^{(l)}$ is the d-th collective variable of the l-th walker. The sum of Gaussian hill potentials is

$$V^{\rm hills}(t) = \sum_{l=1}^{P}\sum_{a=1}^{N_{\rm hills}}h(t_a)\Theta(t-t_a)\prod_{d=1}^{D}\exp\left\{-\frac{\left(\xi_d^{(l)}(t)-\xi_d^{(l)}(t_a)\right)^2}{2\sigma_d^2(t_a)}\right\}, \tag{169}$$

where $h(t_a)$ and $\sigma_d(t_a)$ are the height and width of each Gaussian hill added at time t_a, and $\Theta(t)$ is the Heaviside step function. Since the hills are added where the fictitious particle has passed, this term is history-dependent. The correction term,

$$V^{\rm cor}(t) = -\sum_{l=1}^{P}\sum_{a=1}^{N_{\rm hills}}h(t_a)\Theta(t-t_a), \tag{170}$$

is required to ensure total energy conservation. Note that $V^{\rm hills}(t)$ cancels with $V^{\rm cor}(t)$ at $t=t_a$, making the potential continuous at the position of the evolving fictitious particle. This correction term was not included in the original metadynamics paper. However, it does not affect the metadynamics trajectory, since the derivative of $V^{\rm cor}(t)$ with respect to $\xi_d^{(l)}(t)$ is always zero. Thus, this term is needed only when monitoring the energy conservation of the trajectory. Finally, $H^{\rm bath}$ represents the energy contribution from massive Nosé–Hoover chain thermostats. Thermostat chains are attached to each degree of freedom of the system and to the fictitious particle. See below for the procedure to set thermostat masses.

The RESPA integrator is useful when the forces can be separated into slow and fast components. In the PIMD code, this technique is applied to metadynamics. The forces arising from $V_{\rm cv}$ and $V^{\rm hills}$ are updated with a smaller timestep than those from the physical potential V.

Several parameters control metadynamics. The dimension of the collective variables D, the hill height h, and the hill width σ_d ($1\le d\le D$) must be determined first. During the simulation, a new hill is added whenever the

collective variables move by more than 1.5 times the hill width from the last hill, or when a specified time limit has passed since the previous hill. The latter time is defined by the parameter $\tau_{\rm lim}$. The motion of the fictitious particle is related to the parameter $\tau_{\rm cv}$, which represents the characteristic time for the particle to move 1.5 times the hill width via thermal diffusion at temperature T. This determines the mass of the collective variables by

$$\mu_d = kT\left(\frac{\tau_{\rm cv}}{1.5\sigma_d}\right)^2. \tag{171}$$

The parameter $\tau_{\rm fc}$ is the oscillation period for $V^{\rm cv}$. This determines the force constant by

$$\kappa_d = \mu_d\left(\frac{2\pi}{\tau_{\rm fc}}\right)^2. \tag{172}$$

The following is a recipe to start metadynamics simulations.

1. Decide the dimension of collective variables, D. For each collective variable $1 \le d \le D$, choose the type that properly describes the chemical reaction. Currently, six types are implemented:

   (1) bond distance of atoms A–B,
   (2) bond angle of atoms A–B–C,
   (3) dihedral angle of atoms A–B–C–D,
   (4) difference between A–B and B–C distances,
   (5) coordination number between A and B species,

   (6) difference in coordination numbers between A and B species and C and D species.

   Set the collective variables using the keyword **<nmeta>**.
2. Set the hill height h using the keyword **<gh_meta>**.
3. Set the hill width σ_d for each collective variable using the keyword **<gw_meta>**.
4. Set the characteristic time $\tau_{\rm cv}$ for the fictitious particle to move 1.5 times the hill width, using the keyword **<time_cv_meta>**.
5. Set the time limit for adding hills, $\tau_{\rm lim}$, using the keyword **<time_limit_meta>**.
6. Set the oscillation period of the harmonic potential, $\tau_{\rm fc}$, using the keyword **<time_fc_meta>**.
7. Set the timestep dt for updating the physical potential V using **<dt>**. Also set the number of updates of the rapidly varying potentials $V^{\rm hills}$ and $V^{\rm cv}$ per step dt using **<nref_meta>**.
8. (Optional) Set the maximum number of Gaussian hills using **<mg_meta>**.
9. (Optional) Set the print interval for energy components using **<iprint_meta>**.
10. (Optional) Set the print interval for hill reconstruction using **<iprint_rec_meta>**. Set the parameters for hill reconstruction using **<params_rec_meta>** and **<ndim_rec_meta>**.
11. (Optional) Set the print interval of the collective variable trajectory using **<iprint_cv_meta>**.
12. Set the characteristic timescale for determining the mass of the thermostat attached to the fictitious particle using **<time_cv_bath>**.

In some cases, it may be convenient to change the Gaussian hill height and width during a simulation. To do so, first interrupt the run with a soft exit, and then restart it with new settings of **<gh_meta>** and **<gw_meta>**. During the restarted run, Gaussian hill functions specified in `input.dat` are simply appended to the list `hills.ini`.

13.14 Conventional and well-tempered metadynamics

In conventional metadynamics, the bias potential $V^{\rm hills}$ is equal to the deposited hills $U^{\rm hills}$, i.e.,

$$V^{\rm hills} = U^{\rm hills}, \tag{173}$$

and the free energy is estimated as

$$A^{\rm hills} = -U^{\rm hills}. \tag{174}$$

Meanwhile, in well-tempered metadynamics, the bias potential is

$$V^{\text{hills}} = k_{\text{B}}\Delta T \log\left(1 + \frac{U^{\text{hills}}}{k_{\text{B}}\Delta T}\right), \tag{175}$$

and the free energy landscape is estimated as

$$A^{\text{hills}} = -k_{\text{B}}(T + \Delta T) \log\left(1 + \frac{U^{\text{hills}}}{k_{\text{B}}\Delta T}\right), \tag{176}$$

and therefore

$$A^{\text{hills}} = -\gamma V^{\text{hills}}, \tag{177}$$

where $\gamma = \frac{T+\Delta T}{\Delta T}$. Well-tempered metadynamics with $\Delta T = \infty$ corresponds to conventional metadynamics. The deposited hills U^{hills} are given in the file `hills.ini`. The free energy landscape A^{hills} is given in the file `rec.out`.

The relation between the unbiased probability

$$P_{\text{unbiased}}(\xi) = \frac{e^{-\beta A^{\text{hills}}(\xi)}}{\int d\xi\, e^{-\beta A^{\text{hills}}(\xi)}} \tag{178}$$

and the biased probability

$$P_{\text{biased}}(\xi, t) = \frac{e^{-\beta(A^{\text{hills}}(\xi)+V^{\text{hills}}(\xi,t))}}{\int d\xi e^{-\beta(A^{\text{hills}}(\xi)+V^{\text{hills}}(\xi,t))}} \tag{179}$$

is

$$P_{\text{unbiased}}(\xi) = P_{\text{biased}}(\xi, t) e^{\beta(V^{\text{hills}}(\xi,t)-c(t))}, \tag{180}$$

where

$$c(t) = \beta^{-1} \log \frac{\int d\xi\, e^{-\beta A^{\text{hills}}(\xi)}}{\int d\xi\, e^{-\beta(A^{\text{hills}}(\xi)+V^{\text{hills}}(\xi,t))}} = \beta^{-1} \log \frac{\int d\xi\, e^{\beta\gamma V^{\text{hills}}(\xi,t)}}{\int d\xi\, e^{\beta(\gamma-1)V^{\text{hills}}(\xi,t)}}. \tag{181}$$

Assuming the local equilibrium approximation and ergodicity, the unbiased average of A is given by

$$\langle A \rangle_{\text{unbiased}} \approx \frac{\left\langle A\, e^{\beta(V^{\text{hills}}(\xi,t)-c(t))} \right\rangle_{\text{biased}}}{\left\langle e^{\beta(V^{\text{hills}}(\xi,t)-c(t))} \right\rangle_{\text{biased}}} = \frac{\int_0^T A(t)\, e^{\beta(V^{\text{hills}}(\xi,t)-c(t))}\, dt}{\int_0^T e^{\beta(V^{\text{hills}}(\xi,t)-c(t))}\, dt}, \tag{182}$$

where the last expression describes the time average of the biased metadynamics trajectory.

13.14.1 Coordination number

The coordination number between species A and species B, using the rational function, is given by

$$R_d = \sum_{i\in A} \sum_{j\in B} \frac{1 - \left(\frac{r_{ij}}{d_{ij}}\right)^{\nu_{ij}}}{1 - \left(\frac{r_{ij}}{d_{ij}}\right)^{\mu_{ij}}}. \tag{183}$$

where μ_{ij}, ν_{ij}, and d_{ij} are three parameters that determine the functional form. The coordination number within species A is given by

$$R_d = \sum_{i\in A} \sum_{j>i\in A} \frac{1 - \left(\frac{r_{ij}}{d_{ij}}\right)^{\nu_{ij}}}{1 - \left(\frac{r_{ij}}{d_{ij}}\right)^{\mu_{ij}}}. \tag{184}$$

13.15 Well-sliced metadynamics

The total Hamiltonian is [83, 84]

$$H = H^{\rm cons} + H^{\rm meta} + H^{\rm afed} + H^{\rm bath}, \tag{185}$$

where

$$H^{\rm cons} = \sum_{l=1}^{P} \left[H_{\rm sys}^{(l)} \left(\{\mathbf{r}^{(l)}, \mathbf{p}^{(l)}\} \right) + V_{\rm cv}^{(l)} \left(\left\{ \xi_c^{(l)}, \mathbf{r}^{(l)} \right\} \right) \right] \tag{186}$$

is the contribution from constrained MD,

$$\begin{aligned} H^{\rm meta} &= \sum_{l=1}^{P} \left[H_{\rm sys}^{(l)} \left(\{\mathbf{r}^{(l)}, \mathbf{p}^{(l)}\} \right) + T_{\rm cv}^{(l)} \left(\left\{ \boldsymbol{\pi}_m^{(l)} \right\} \right) + V_{\rm cv}^{(l)} \left(\left\{ \boldsymbol{\xi}_m^{(l)}, \mathbf{r}^{(l)} \right\} \right) \right] \\ &+ V^{\rm hills} \left(\left\{ \boldsymbol{\xi}_m^{(1)}, \cdots, \boldsymbol{\xi}_m^{(P)}, \mathbf{r}^{(1)}, \cdots, \mathbf{r}^{(P)} \right\} \right) + V^{\rm cor} \end{aligned} \tag{187}$$

is the contribution from metadynamics,

$$H^{\rm afed} = \sum_{l=1}^{P} \left[H_{\rm sys}^{(l)} \left(\{\mathbf{r}^{(l)}, \mathbf{p}^{(l)}\} \right) + T_{\rm cv}^{(l)} \left(\left\{ \boldsymbol{\pi}_a^{(l)} \right\} \right) + V_{\rm cv}^{(l)} \left(\left\{ \boldsymbol{\xi}_a^{(l)}, \mathbf{r}^{(l)} \right\} \right) \right] \tag{188}$$

is the contribution from adiabatic free energy dynamics, and $H^{\rm bath}$ is the contribution from thermostats. Note that, in the implementation in the PIMD code, the temperature is always set equal to the physical temperature. The free energy with respect to the constrained MD collective variable, ξ_c, and the metadynamics collective variable, ξ_m, is defined as

$$A(\xi_c, \xi_m) = -\beta^{-1} \log \int d\mathbf{r} \exp(-\beta V_{\rm sys}(\mathbf{r})). \tag{189}$$

The reweighting scheme employed in the PIMD code is different from that of the original method. We use the equivalence for the free energy:

$$A(\xi_c, \xi_m) = A'_c(\xi_c) + \Delta A_{\xi_c}(\xi_m), \tag{190}$$

where we have defined the free energy in the reduced space

$$A'_c(\xi_c) = -\beta^{-1} \log \int d\xi_m \exp(-\beta A(\xi_c, \xi_m)), \tag{191}$$

and the shift from it

$$\Delta A_{\xi_c}(\xi_m) = -\beta^{-1} \log \frac{\exp(-\beta A(\xi_c, \xi_m))}{\int d\xi_m \exp(-\beta A(\xi_c, \xi_m))}. \tag{192}$$

The first term is evaluated by thermodynamic integration along the set of different constraints $(\xi_c^{(1)}, \cdots, \xi_c^{(P)})$,

$$A'_c(\xi_c) = \int d\xi_c \nabla A'_c(\xi_c) = \sum_l \left(\xi_c^{(l+1)} - \xi_c^{(l)} \right) \left(\frac{\nabla A'_c(\xi_c^{(l+1)}) + \nabla A'_c(\xi_c^{(l)})}{2} \right), \tag{193}$$

where

$$\begin{aligned} \nabla A'_c(\xi_c) &= \frac{\nabla \int d\xi_m \exp(-\beta A(\xi_c, \xi_m))}{\int d\xi_m \exp(-\beta A(\xi_c, \xi_m))} = \frac{\int d\mathbf{r} \exp(-\beta V_{\rm sys}(\mathbf{r})) \nabla \delta(\xi_c - R_c(\mathbf{r}))}{\int d\mathbf{r} \exp(-\beta V_{\rm sys}(\mathbf{r})) \delta(\xi_c - R_c(\mathbf{r}))} \\ &\approx \frac{\int d\xi_c \int d\mathbf{r} \exp(-\beta(V_{\rm sys}(\mathbf{r}) + V_{\rm cv}(\mathbf{r}, \xi_c))) [-\beta \kappa_c (\xi_c - R_c(\mathbf{r}))]}{\int d\xi_c \int d\mathbf{r} \exp(-\beta(V_{\rm sys}(\mathbf{r}) + V_{\rm cv}(\mathbf{r}, \xi_c)))} \\ &= \frac{\int d\xi_m \int d\xi_c \int d\mathbf{r} \exp(-\beta V) [-\beta \kappa_c (\xi_c - R_c(\mathbf{r}))] \exp(\beta(V^{\rm hills}(\xi_m) + V^{\rm cor}))}{\int d\xi_m \int d\xi_c \int d\mathbf{r} \exp(-\beta V) \exp(\beta(V^{\rm hills}(\xi_m) + V^{\rm cor}))} \\ &= \frac{\langle [-\beta \kappa_c (\xi_c - R_c(\mathbf{r}))] \exp(\beta(V^{\rm hills}(\xi_m) + V^{\rm cor}))] \rangle}{\langle \exp(\beta(V^{\rm hills}(\xi_m) + V^{\rm cor})) \rangle}, \end{aligned} \tag{194}$$

with

$$V = V_{\rm sys}(\mathbf{r}) + V_{\rm cv}(\mathbf{r}, \xi_c) + V^{\rm hills}(\xi_m) + V^{\rm cor}. \tag{195}$$

Note that the second term does not change upon the addition of any constant function of ξ_c. Explicitly,

$$\Delta A_{\xi_c}(\xi_m) = -\beta^{-1} \log \frac{\exp(-\beta A''_{\xi_c}(\xi_m))}{\int d\xi_m \exp(-\beta A''_{\xi_c}(\xi_m))}, \tag{196}$$

for

$$A''_{\xi_c}(\xi_m) = A(\xi_c, \xi_m) + c(\xi_c). \tag{197}$$

Thus, $A''_{\xi_c}(\xi_m)$ is taken as the metadynamics free energy surface at each constraint, ξ_c.

13.16 Mean force dynamics (AFED)

To keep the notation compact, ξ_d and π_d denote the coordinate and conjugate momentum of the fictitious CV particle, μ_d its fictitious mass, and κ_d the harmonic coupling constant. The Helmholtz free energy of the l-th bead (replica) is defined as

$$A_{\rm cv}^{(l)}\left(\xi_1, \cdots, \xi_D\right) = -\beta^{-1} \log \int dr_{1x}^{(l)} \cdots dr_{Nz}^{(l)} e^{-\beta V^{(l)}\left(r_{1x}^{(l)}, \cdots, r_{Nz}^{(l)}\right)} \prod_{d=1}^{D} \delta\left(\xi_d^{(l)} - R_d^{(l)}\left(\mathbf{r}^{(l)}\right)\right) \tag{198}$$

where $\{\mathbf{r}^{(l)}\}$ and $\{\mathbf{p}^{(l)}\}$ denote the sets of $3N$-dimensional Cartesian coordinates and momenta of the system for the l-th bead, while $\left\{\boldsymbol{\xi}^{(l)}\right\} = \left(\xi_1^{(l)}, \cdots, \xi_D^{(l)}\right)$ and $\left\{\boldsymbol{\pi}^{(l)}\right\} = \left(\pi_1^{(l)}, \cdots, \pi_D^{(l)}\right)$ denote the sets of D-dimensional coordinates and momenta of the fictitious CV particles, respectively, for the l-th bead. $R_d^{(l)}$ is the d-th collective variable of the l-th bead.

In driven AFED [79], the free energy can be approximated as

$$A_{\rm cv}^{(l)} \approx -\beta^{-1} \log \int dr_{1x} \cdots dr_{Nz} e^{-\beta\left(V^{(l)} + V_{\rm cv}^{(l)}\right)} + \text{const.} \tag{199}$$

where

$$V_{\rm cv}^{(l)} = \sum_{d=1}^{D} \frac{\kappa_d}{2} \left\{\xi_d^{(l)} - R_d^{(l)}\left(\mathbf{r}^{(l)}\right)\right\}^2 \tag{200}$$

is the sum of harmonic potentials acting between the fictitious particle $\xi_d^{(l)}$ and the collective variable $R_d^{(l)}$ with the force constant κ_d. The free energy gradient is expressed as

$$\frac{\partial A_{\rm cv}^{(l)}}{\partial \xi_d^{(l)}} = \frac{\int dr_{1x} \cdots dr_{Nz} \kappa_d \left\{\xi_d^{(l)} - R_d^{(l)}\left(\mathbf{r}^{(l)}\right)\right\} e^{-\beta\left(V^{(l)} + V_{\rm cv}^{(l)}\right)}}{\int dr_{1x} \cdots dr_{Nz} e^{-\beta\left(V^{(l)} + V_{\rm cv}^{(l)}\right)}} = \left\langle \kappa_d \left\{\xi_d^{(l)} - R_d^{(l)}\left(\mathbf{r}^{(l)}\right)\right\}\right\rangle_{\rm md}. \tag{201}$$

Except for the finite-difference version of gentlest ascent dynamics, the coordinates and momenta are identical for all beads $1 \le l \le P$, i.e.,

$$\xi_d = \xi_d^{(l)}, \quad \pi_d = \pi_d^{(l)} \tag{202}$$

In this case, the mean force in Eq.(201) can be estimated using parallel dynamics [77] as

$$\frac{\partial A_{\rm cv}}{\partial \xi_d} = \sum_{l=1}^{P} W_l \left\langle \kappa_d \left\{\xi_d - R_d^{(l)}\left(\mathbf{r}^{(l)}\right)\right\}\right\rangle_{\rm md}, \tag{203}$$

where

$$W_l = \frac{e^{-\beta w^l}}{\sum_l e^{-\beta w^l}}, \tag{204}$$

and

$$w^l = \sum_d \int_{\xi_d(0)}^{\xi_d(t)} \left\langle \frac{\partial V_{\rm cv}^{(l)}}{\partial \xi_d} \right\rangle_{\rm md} d\xi_d. \tag{205}$$

W_l in Eq.(204) is used when <ioption_afed> = 2, while W_l is set to "1" for any replica with <ioption_afed> = 1 (default setting). See ref.[77] for details of the parallel dynamics.

13.16.1 Steepest descent

Theory.. The minimum of the free energy is searched using the steepest descent method. The mass-weighted coordinate of a collective variable is defined as

$$q_d = \sqrt{\mu_d}\xi_d. \tag{206}$$

The steepest descent path is defined by the trajectory of the equation

$$\frac{dq_d}{dt} = -\frac{\nabla_d A_{\rm cv}}{|\nabla A_{\rm cv}|} \tag{207}$$

where the free energy gradient with respect to q_d is defined as

$$\nabla_d A_{\rm cv} = \frac{\partial A_{\rm cv}}{\partial q_d} = \frac{1}{\sqrt{\mu_d}} \frac{\partial A_{\rm cv}}{\partial \xi_d}, \tag{208}$$

and the norm is given by

$$|\nabla A_{\rm cv}|^2 = \sum_{d=1}^{D} \frac{1}{\mu_d} \left(\frac{\partial A_{\rm cv}}{\partial \xi_d} \right)^2. \tag{209}$$

Then Eq.(207) is equivalent to

$$\mu_d \frac{d\xi_d}{dt} = -\frac{1}{|\nabla A_{\rm cv}|} \frac{\partial A_{\rm cv}}{\partial \xi_d}. \tag{210}$$

In the PIMD code, the mass of the fictitious particle is set as

$$\mu_d = \frac{1}{\Delta \xi_d^{\rm ref\,2}} \tag{211}$$

where the reference shift $\Delta\xi_d^{\rm ref}$ is a parameter with the same dimension as the corresponding collective variable. Then $\Delta t_{\rm cv}$ becomes dimensionless.

Implementation..

- The Euler method

$$q_d(t + dt) = q_d(t) - \frac{\nabla_d A_{\rm cv}}{|\nabla A_{\rm cv}|} dt \tag{212}$$

  is employed.
- The PIMD code checks whether the position of the fictitious particle is inside the designated range

$$\xi_d^{\rm min} < \xi_d < \xi_d^{\rm max} \tag{213}$$

  and the free energy satisfies

$$A_{\rm cv} < A_{\rm cv}^{\rm max}. \tag{214}$$

  The calculation stops if these conditions are violated. The parameters $\xi_d^{\rm min}$ and $\xi_d^{\rm max}$ are specified by the keyword <params_afed>, while the parameter $A_{\rm cv}^{\rm max}$ is specified by the keyword <fenergy_max_afed>.

- The AFED step size is initially set by $\Delta t_{\rm cv}$ = (the value given by the keyword **<dt_descent_afed>**), but it is reduced when the system is judged to be close to a minimum. This judgement is based on the angle between the free energy gradients of the current AFED step (t) and the previous AFED step $(t - \Delta t_{\rm cv})$,

$$\theta = \arccos\left(\frac{\nabla A_{\rm cv}(t) \cdot \nabla A_{\rm cv}(t - \Delta t_{\rm cv})}{|\nabla A_{\rm cv}(t)|\,|\nabla A_{\rm cv}(t - \Delta t_{\rm cv})|}\right). \tag{215}$$

  If $\theta > 90°$, the system is judged to be converging toward a stationary point (a minimum in this case). Then $\Delta t_{\rm cv}$ is decreased by multiplying by a damping factor $\lambda^{\rm afed}$, i.e., $\Delta t_{\rm cv} \to \lambda^{\rm afed}\Delta t_{\rm cv}$, where $\lambda^{\rm afed}$ is specified by the keyword **<dt_damp_afed>**. On the other hand, if $\theta \le 90°$ persists for more than five consecutive AFED steps, the system is judged to be far from a stationary point. Then $\Delta t_{\rm cv}$ is increased again by dividing by the damping factor, $\Delta t_{\rm cv} \to \Delta t_{\rm cv}/\lambda^{\rm afed}$. This adjustment occurs only if $\Delta t_{\rm cv}$ is smaller than the initial value specified by the keyword **<dt_descent_afed>**. When $\Delta t_{\rm cv}$ becomes smaller than the converged AFED step size, $\Delta t_{\rm cv}$ < (the value given by the keyword **<dt_conv_afed>**), the calculation stops.

Simulation setup.. Here is a recipe for starting steepest descent simulations.

1. Decide the dimension of the collective variables, D. Then, for each collective variable $1 \le d \le D$, choose the type that properly describes the chemical reaction. Currently, five types are implemented, i.e.,

   (1) bond distance between atoms A–B,
   (2) bond angle associated with atoms A–B–C,
   (3) dihedral angle associated with atoms A–B–C–D,
   (4) difference between A–B and B–C distances,
   (5) coordination number between A and B species.

   Set the collective variables using the keyword **<ncons>**.
2. Set the force constant of the harmonic potential, κ_d, using the keyword **<params_cons>**.
3. Set the initial AFED step size, the damping factor, and the converged AFED step size using the keywords **<dt_descent_afed>**, **<dt_damp_afed>**, and **<dt_conv_afed>**, respectively.
4. Set the reference shift, $\Delta\xi_d^{\rm ref}$, as well as the lower bound $\xi_d^{\rm min}$ and the upper bound $\xi_d^{\rm max}$, for each type of collective variable using the keyword **<params_afed>**.
5. Set the upper bound of the free energy to be explored, $A_{\rm cv}^{\rm max}$, using the keyword **<fenergy_max_afed>**.
6. Set the number of preliminary and production MD steps using the keywords **<nstep_pre_afed>** and **<nstep_pro_afed>**, respectively.

13.16.2 Gentlest ascent dynamics

Theory.. The saddle point of the free energy is searched using gentlest ascent dynamics. The position of the fictitious particle is described in terms of the mass-weighted coordinates q_d. Gentlest ascent dynamics is defined by the trajectory of the equation

$$\frac{dq_d}{dt} = \frac{\tilde{F}_d}{\sqrt{\sum_{d=1}^{D}\tilde{F}_d^2}} \tag{216}$$

with

$$\tilde{F}_d = -\nabla_d A_{\rm cv} + \left(2\sum_{d'=1}^{D}\nabla_{d'}A_{\rm cv}\cdot n_{d'}\right)n_d, \tag{217}$$

where n_d is a vector that evolves according to

$$\frac{dn_d}{dt} = \frac{1}{\gamma^{\rm gad}}\left[-\sum_{d'=1}^{D}\nabla_{dd'}^2 A_{\rm cv}\cdot n_{d'} + \left(\sum_{d'=1}^{D}n_{d'}\sum_{d''=1}^{D}\nabla_{d'd''}^2 A_{\rm cv}\cdot n_{d''}\right)n_d\right] \tag{218}$$

with the Hessian

$$\nabla_{dd'}^2 A_{\rm cv} = \frac{\partial^2 A_{\rm cv}}{\partial q_d \partial q_{d'}} \tag{219}$$

and the parameter $\gamma^{\rm gad}$.

Implementation..

- The Euler method is used to update the position of the fictitious particle.

$$q_d(t+dt) = q_d + \frac{\tilde{F}_d}{\sqrt{\sum_{d=1}^{D} \tilde{F}_d^2}} dt \tag{220}$$

with

$$\tilde{F}_d = -a_d + \left(2 \sum_{d'=1}^{D} a_{d'} \cdot n_{d'} \right) n_d. \tag{221}$$

The vector a_d is computed either analytically or numerically (see below).

- The following equation is used to update the unit vector.

$$n_d(t+dt) = \frac{n_d(t) - b_d(t) dt/\gamma^{\mathrm{gad}}}{\sqrt{\sum_{d'=1}^{D} \left[n_{d'}(t) - b_{d'}(t) dt/\gamma^{\mathrm{gad}} \right]^2}} \tag{222}$$

where

$$b_d(t) = \sum_{d'=1}^{D} \nabla^2_{dd'} A_{\mathrm{cv}}(t) \cdot n_{d'}(t). \tag{223}$$

The vector b_d is computed either analytically or numerically (see below).

- The PIMD code ensures that the position of the fictitious particle is inside the designated range

$$\xi_d^{\min} < \xi_d < \xi_d^{\max}. \tag{224}$$

The calculation stops if this range is exceeded. The parameters $\xi_d^{\min}$ and $\xi_d^{\max}$ are given by the keyword **<params_afed>**, while the parameter $A_{\mathrm{cv}}^{\max}$ is given by the keyword **<fenergy_max_afed>**.

- The AFED step size is initially set by $\Delta t_{\mathrm{cv}} =$ (the value given by the keyword **<dt_ascent_afed>**), but it is reduced when the system is judged to be close to the saddle point. This judgement is based on the angle between the free energy gradients of the current AFED step (t) and the previous AFED step $(t - \Delta t_{\mathrm{cv}})$,

$$\theta = \arccos \left(\frac{\nabla A_{\mathrm{cv}}(t) \cdot \nabla A_{\mathrm{cv}}(t - \Delta t_{\mathrm{cv}})}{|\nabla A_{\mathrm{cv}}(t)| \, |\nabla A_{\mathrm{cv}}(t - \Delta t_{\mathrm{cv}})|} \right). \tag{225}$$

If $\theta > 90°$, the system is judged to be converging to a stationary point (a saddle point, in this case). Then Δt_{cv} is decreased by multiplying by a damping factor λ^{afed}, i.e., $\Delta t_{\mathrm{cv}} \to \lambda^{\mathrm{afed}} \Delta t_{\mathrm{cv}}$, where λ^{afed} is given by the keyword **<dt_damp_afed>**. On the other hand, if $\theta \leq 90°$ persists for more than five consecutive AFED steps, the system is judged to be far from a stationary point. Then Δt_{cv} is increased again by dividing by the damping factor, $\Delta t_{\mathrm{cv}} \to \Delta t_{\mathrm{cv}} / \lambda^{\mathrm{afed}}$. This adjustment occurs only if Δt_{cv} is smaller than the initial value given by the keyword **<dt_descent_afed>**. When Δt_{cv} becomes smaller than the converged AFED step size, $\Delta t_{\mathrm{cv}} <$ (the value given by the keyword **<dt_conv_afed>**), the calculation stops.

- In the case of **<hessian_ascent_afed>** = ANALYTICAL, the Hessian is calculated analytically.

$$\frac{\partial^2 A_{\mathrm{cv}}^{(l)}}{\partial \xi_d^{(l)} \partial \xi_{d'}^{(l)}} = \delta_{dd'} \kappa_d - \beta \left[\left\langle f_d^{(l)} f_{d'}^{(l)} \right\rangle_{\mathrm{md}} - \left\langle f_d^{(l)} \right\rangle_{\mathrm{md}} \left\langle f_{d'}^{(l)} \right\rangle_{\mathrm{md}} \right] \tag{226}$$

where

$$f_d^{(l)} = -\kappa_d \left\{ \xi_d^{(l)} - R_d^{(l)} \left(\mathbf{r}^{(l)} \right) \right\}. \tag{227}$$

- In the case of **<hessian_ascent_afed>** = NUMERICAL, the finite difference method is employed. This is done as follows. The free energy gradient, $\nabla_{d'} A_{\mathrm{cv}}$, is evaluated at each bead s, at the position $\boldsymbol{q}^{(s)} = \left\{ q_1^{(s)}, \cdots, q_D^{(s)} \right\}$ with $1 \leq s \leq P$. The P beads are centered at $\boldsymbol{q} = \{ q_1, \cdots, q_D \}$ and are distributed along the unit vector $\mathbf{n} = \{ n_1, \cdots, n_D \}$ with equal spacing from $-\epsilon^{\mathrm{gad}}$ to $+\epsilon^{\mathrm{gad}}$, i.e.,

$$q_d^{(s)} = q_d + n_d \epsilon_s^{\mathrm{gad}} \tag{228}$$

where

$$\epsilon_s^{\mathrm{gad}} = \epsilon^{\mathrm{gad}} \left(\frac{2s - P - 1}{P - 1} \right) \tag{229}$$

and ϵ^{gad} is a constant value given by the keyword **<fdiff_sampling_afed>**. From the first-order Taylor expansion of the gradient of each bead $\nabla_{d'} A_{\mathrm{cv}} \left(\boldsymbol{q}^{(s)} \right)$ around the center $\boldsymbol{q}$, one obtains

$$\nabla_{d'} A_{\mathrm{cv}} \left(\boldsymbol{q}^{(s)} \right) = a_{d'} + b_{d'} \epsilon_s^{\mathrm{gad}} + c_{d'} \epsilon_s^{\mathrm{gad}^2} + \cdots \tag{230}$$

where

$$a_{d'} = \nabla_{d'} A_{\mathrm{cv}} \left(\boldsymbol{q} \right) = \nabla_{d'} A_{\mathrm{cv}} \tag{231}$$

and

$$b_{d'} = \sum_{d=1}^{D} \nabla_{dd'}^2 A_{\mathrm{cv}} \left(\boldsymbol{q} \right) \cdot n_d = \sum_{d=1}^{D} \nabla_{dd'}^2 A_{\mathrm{cv}} \cdot n_d. \tag{232}$$

Thus, $a_{d'}$ and $b_{d'}$ can be obtained by a least-squares fitting of the function $\nabla_{d'} A_{\mathrm{cv}} \left(\boldsymbol{q}^{(s)} \right)$ with respect to $\epsilon_s^{\mathrm{gad}}$. In the PIMD code, the least-squares fitting is performed using a quadratic function.

The implementation has been modified from version 2.3 as follows with respect to the guiding vector $\mathbf{n}$.

- When restarted with the status “EQ”, a random displacement is applied in the first AFED step, and $\mathbf{n}$ is generated along the displacement vector.
- When restarted with the status “AS”, a calculation of the full Hessian $\mathbf{H}$ (which requires computational time equivalent to D AFED steps, where D is the CV dimension) is performed at the first AFED step, and $\mathbf{n}$ is set as the lowest eigenvector.
- A calculation of $\mathbf{H}$ is performed every 20 AFED steps, and $\mathbf{n}$ is reset as the lowest eigenvector.
- $\mathbf{n}$ is not updated at any AFED step while the value of $\omega^2 = \mathbf{n}^\dagger \cdot \mathbf{Hn}$ is positive (which means that the curvature is downward along the direction $\mathbf{n}$).

The AFED step size is controlled as follows.

- The AFED step size is adjusted after the fifth AFED step.
- The AFED step is accepted as long as the angle between the last two moves does not exceed 90 degrees.
- If the last AFED step is rejected, the AFED step size is reduced by 70 percent.
- However, if the last five moves are accepted, the AFED step size is increased by 30 percent.
- The run is terminated once the AFED step size falls below a threshold value.

Simulation setup.. Here is a recipe to start gentlest ascent dynamics simulations.

1. Decide the dimension of the collective variables, D. Then, for each collective variable $1 \leq d \leq D$, choose the type that properly describes the chemical reaction. So far, eight types are implemented, i.e.,

   (1) bond distance of atoms A–B,
   (2) bond angle associated with atoms A–B–C,
   (3) dihedral angle associated with atoms A–B–C–D,
   (4) difference between A–B and B–C distances,
   (5) coordination number between A and B species,

   (6) difference in coordination numbers between A–B and C–D species,

   (7) center of mass of A and B species,

   (8) difference of centers of mass of A and B species.

   Set the collective variables using the keyword **<ncons>**.

2. Set the force constant of the harmonic potential, κ_d, using the keyword <params_cons>.
3. Set the parameter of gentlest ascent dynamics, γ^{gad}, using the keyword <gamma_ascent_afed>.
4. Set the initial AFED step sizes for descent and ascent trajectories, the damping factor, and the converged AFED step size using the keywords <dt_descent_afed>, <dt_ascent_afed>, <dt_damp_afed>, and <dt_conv_afed>, respectively.
5. Set the reference shift, $\Delta\xi_d^{\mathrm{ref}}$, as well as the lower bound ξ_d^{min} and the upper bound ξ_d^{max}, for each collective variable using the keyword <params_afed>.
6. Set the keyword <hessian_ascent_afed>, either "NUMERICAL" or "ANALYTICAL". For the numerical case, set the finite difference parameter, ϵ^{gad}, using the keyword <fdiff_sampling_afed>.
7. Set the upper bound of the free energy to be explored, $A_{\mathrm{cv}}^{\mathrm{max}}$, using the keyword <fenergy_max_afed>.
8. Set the number of preliminary and productive MD steps using the keywords <nstep_pre_afed> and <nstep_pro_afed>, respectively.

13.16.3 Automated search

Implementation.. The automated search calculation proceeds as follows.

1. The calculation starts with an initial descent search (DE) to locate a free energy minimum point (EQ). The data are written to the list file `auto.ini`.
2. An ascent search (AS) is launched from this EQ in a random direction.
3. If a free energy saddle point (TS) is found, check the list file to determine whether this is a new one. If it is an old one (previously found TS), return to step 2.
4. Once a new TS is found, the data are added to the list file. Then start two descent searches in the forward and backward directions from the TS. These trajectories, referred to as D1 and D2, should lead to two minima. The minima are checked, and if they are new, the data are added to the list file.
5. The ascent search fails if it either finds an old TS, enters regions with free energy that is too high, or goes out of bounds.
6. If the ascent search is performed N_{shot} times from the same EQ, or if it does not find a new TS for N_{miss} consecutive attempts from the same EQ, the ascent search from that EQ is abandoned. In this case, another EQ in the list file is selected, and the process returns to step 2.
7. The calculation stops when all EQs have undergone the ascent search.

The output files are:

- `auto.ini`: The list of EQ and TS points that have been explored.
- `afed.ini`: The AFED restart file.
- *EQ.*.tar* and *TS.*.tar*: The tape archive containing the geometry file `geometry.ini` and the restart file `afed.ini` of EQ and TS, respectively, where * is the ID number.

A point in the mass-weighted CV space, q_d, is identified as a new point if it is distant from all other points in the list, q_d^{list}, i.e.,

$$\sqrt{\sum_{d=1}^{D}\left(q_d - q_d^{\mathrm{list}}\right)^2} > q^{\mathrm{rad}}. \tag{233}$$

Otherwise, q_d is identified as an old point.

Simulation setup.. Here is a recipe to start gentlest ascent dynamics simulations.

1. Decide the dimension of the collective variables, D. Then, for each collective variable $1 \le d \le D$, choose the type that properly describes the chemical reaction. So far, five types are implemented, i.e.,

   (1) bond distance of atoms A–B,
   (2) bond angle associated with atoms A–B–C,
   (3) dihedral angle associated with atoms A–B–C–D,
   (4) difference between A–B and B–C distances,

(5) coordination number between A and B species.

Set the collective variables using the keyword <ncons>.

2. Set the force constant of the harmonic potential, κ_d, using the keyword <params_cons>.
3. Set the parameter of gentlest ascent dynamics, γ^{gad}, using the keyword <gamma_ascent_afed>.
4. Set the initial AFED step size, the damping factor, and the converged AFED step size using the keywords <dt_ascent_afed>, <dt_damp_afed>, and <dt_conv_afed>, respectively.
5. Set the reference shift, $\Delta\xi_d^{\mathrm{ref}}$, as well as the lower bound ξ_d^{min} and the upper bound ξ_d^{max}, for each type of collective variable using the keyword <params_afed>.
6. Set the keyword <hessian_ascent_afed>, either "NUMERICAL" or "ANALYTICAL". For the numerical case, set the finite difference parameter, ϵ^{gad}, using the keyword <fdiff_sampling_afed>.
7. Set the upper bound of the free energy to be explored, $A_{\mathrm{cv}}^{\mathrm{max}}$, using the keyword <fenergy_max_afed>.
8. Set the number of preliminary and productive MD steps using the keywords <nstep_pre_afed> and <nstep_pro_afed>, respectively.
9. Set the reference radius, q^{rad}, using the keyword <radius_auto_afed>.
10. Set the maximum number of shots from a minimum, N_{shot}, using the keyword <nshot_auto_afed>.
11. Set the number of consecutive missed shots from a minimum, N_{miss}, using the keyword <nmiss_auto_afed>.

13.16.4 Temperature accelerated molecular dynamics

Theory.. In temperature accelerated molecular dynamics (TAMD) with the Nosé-Hoover thermostat [79], the pseudo Hamiltonian is defined as

$$H^{\mathrm{tamd}} = T_{\mathrm{cv}} + V_{\mathrm{cv}}^{\mathrm{tamd}} + H_{\mathrm{cv}}^{\mathrm{bath}} \tag{234}$$

where

$$T_{\mathrm{cv}} = \sum_{d=1}^{D} \frac{{\pi_d}^2}{2\mu_d}, \tag{235}$$

is the kinetic energy of the fictitious particles, and $H_{\mathrm{cv}}^{\mathrm{bath}}$ is the energy of the thermostat attached to the fictitious particles. The effective potential is defined as

$$V_{\mathrm{cv}}^{\mathrm{tamd}} = \left(\frac{T}{T_{\mathrm{tamd}}}\right) A_{\mathrm{cv}} \tag{236}$$

where the temperature T_{tamd} is an input parameter.

The mass of the collective variables is determined by

$$\mu_d = \frac{k_B T \Delta t_{\mathrm{cv}}^2}{{\Delta\xi_d^{\mathrm{ref}}}^2} \tag{237}$$

where the parameter $\Delta\xi_d^{\mathrm{ref}}$ is the reference displacement of the fictitious particle given by <params_afed> (see Sec.5.4.1).

When the AFED time step Δt_{cv} is set to be dimensionless, μ_d has the dimension of (energy)/(collective variable) (default is "$\Delta t_{\mathrm{cv}} = 1.0$").

Implementation.. For NVE-type TAMD, the thermostats are not applied, and thus $H_{\mathrm{cv}}^{\mathrm{bath}} = 0$. For NVT- and VS-type TAMD, the temperature is maintained at the physical temperature, T, by the Nosé-Hoover thermostat or the velocity scaling (Gaussian thermostat), respectively, which are attached to the whole set of fictitious variables (global thermostatting). The energy of the Nosé-Hoover thermostat is given by

$$H_{\mathrm{cv}}^{\mathrm{bath}} = \frac{p_{\eta,1}^2}{2Q_{\eta,1}} + D k_B T \eta_1. \tag{238}$$

where the thermostat mass is chosen to be $Q_{\eta,1} = Dk_BT\tau_{\rm cv}^2$, and $\tau_{\rm cv}$ is the typical time scale of the fictitious variables.

The parameters $\xi_d^{\rm min}$ and $\xi_d^{\rm max}$ are the lower and upper bounds, respectively, in the collective variable space to be explored. If the component d of the collective variables goes out of bounds, $\xi_d > \xi_d^{\rm max}$ or $\xi_d < \xi_d^{\rm min}$, then the velocity of the component d is reflected, i.e., $v_d \to -v_d$.

The parameter $A_{\rm cv}^{\rm max}$ sets the upper limit of the free energy. If $A_{\rm cv}$ exceeds $A_{\rm cv}^{\rm max}$, then the velocity is reversed, i.e., $v_d \to -v_d$ for all d.

Simulation setup.. Here is a recipe to start TAMD simulations.

1. Decide the dimension of the collective variables, D. Then, for each collective variable $1 \le d \le D$, choose the type that properly describes the chemical reaction. So far, five types are implemented, i.e.,

   (1) bond distance of atoms A–B,
   (2) bond angle associated with atoms A–B–C,
   (3) dihedral angle associated with atoms A–B–C–D,
   (4) difference between A–B and B–C distances,
   (5) coordination number between A and B species.

   Set the collective variables using the keyword **<ncons>**.
2. Set the TAMD type, **<tamd_type>** = `NVE, NVT, VS`.
3. Set the TAMD parameter $T_{\rm tamd}$, using the keyword **<temperature_tamd>**.
4. Set the reference displacement, $\Delta\xi_d^{\rm ref}$, as well as the lower bound $\xi_d^{\rm min}$ and the upper bound $\xi_d^{\rm max}$, for each type of collective variable using the keyword **<params_afed>**. Note that $\Delta\xi_d^{\rm ref}$ is used to determine μ_d using Eq.(237). Additionally, one can use **<mass_afed>** to further tune μ_d.
5. Set the force constant of the harmonic potential, κ_d, using the keyword **<params_cons>**.
6. Optionally, the maximum free energy value ($A_{\rm cv}^{\rm max}$) to be sampled and the AFED step size $\Delta t_{\rm cv}$ can be manually set using **<fenergy_max_afed>** and **<dt_tamd>**, respectively. Also, the mass of the thermostat should be set using **<niter_bath_afed>** for **<tamd_type>** = `NVT`.

13.16.5 Logarithmic mean force dynamics

Theory.. In logarithmic mean force dynamics (LogMFD) with the Nosé-Hoover thermostat, the pseudo Hamiltonian is defined as

$$H^{\rm log} = T_{\rm cv} + V_{\rm cv}^{\rm log} + H_{\rm cv}^{\rm bath} \tag{239}$$

where

$$T_{\rm cv} = \sum_{d=1}^{D} \frac{{\pi_d}^2}{2\mu_d}, \tag{240}$$

is the kinetic energy of the fictitious particles, and $H_{\rm cv}^{\rm bath}$ is the energy of the thermostat attached to the fictitious particles. The effective potential is defined as

$$V_{\rm cv}^{\rm log} = {\rm sgn}\left(A_{\rm cv}\right)\gamma_{\rm log}\log\left\{\alpha_{\rm log}\left|A_{\rm cv}\right| + 1\right\} \tag{241}$$

where $\alpha_{\rm log}$ and $\gamma_{\rm log}$ are positive input parameters. If the largest free-energy barrier in the CV space, ΔF, can be roughly estimated, $\alpha_{\rm log}$ is given as $\alpha_{\rm log} \sim 1.5\log\left(\frac{\Delta F}{k_BT}\right)/k_BT$ (see ref. [77]). Note that $\alpha_{\rm log}$ can take any positive value as long as the fictitious variables sufficiently explore the CV space. Also, $\gamma_{\rm log}$ is normally set as $\gamma_{\rm log} = 1/\alpha_{\rm log}$ (default setting).

The free energy is calculated on-the-fly as

$$A_{\rm cv} = \frac{{\rm sgn}\left(H - T_{\rm cv} - H_{\rm cv}^{\rm bath}\right)}{\alpha_{\rm log}}\left\{\exp\left(\frac{\left|H^{\rm log} - T_{\rm cv} - H_{\rm cv}^{\rm bath}\right|}{\gamma_{\rm log}}\right) - 1\right\} \tag{242}$$

from the conservation of $H^{\log}$. The origin of the free energy is set by specifying the value of $A_{\rm cv}(0)$ (default is "0.003"). When restarted, $A_{\rm cv}(0)$ is automatically set by `afed.ini`, which contains the final value of $A_{\rm cv}$ in the previous LogMFD run. In the same way as Eq. (242), $A_{\rm cv}$ is calculated at each AFED step in NVE and VS LogMFD calculations (for the latter, more details will be discussed in a forthcoming paper by Morishita *et al.*).

The mass of the collective variables is determined by

$$\mu_d = \frac{k_B T \Delta t_{\rm cv}^2}{\Delta \xi_d^{\rm ref\,2}} \tag{243}$$

where the parameter $\Delta\xi_d^{\rm ref}$ is the reference displacement of the fictitious particle given by **<params_afed>** (see Sec.5.4.1). When the AFED time step $\Delta t_{\rm cv}$ is set to be dimensionless, μ_d has the dimension of (energy)/(collective variable) (default is "$\Delta t_{\rm cv} = 1.0$").

Implementation.. For NVE-type LogMFD, the thermostat is not applied, and thus $H_{\rm cv}^{\rm bath} = 0$. For NVT- and VS-type LogMFD, the temperature is maintained at the physical temperature, T, by the Nosé-Hoover thermostat or the velocity scaling (Gaussian thermostat), respectively, which are attached to the whole set of fictitious variables (global thermostatting). The energy of the Nosé-Hoover thermostat is given by

$$H_{\rm cv}^{\rm bath} = \frac{p_{\eta,1}^2}{2Q_{\eta,1}} + D k_B T \eta_1. \tag{244}$$

where the thermostat mass is chosen to be $Q_{\eta,1} = D k_B T \tau_{\rm cv}^2$, and $\tau_{\rm cv}$ is the typical time scale of the fictitious variables.

The parameters $\xi_d^{\rm min}$ and $\xi_d^{\rm max}$ are the lower and upper bounds, respectively, in the collective variable space to be explored. If the component d of the collective variables goes out of bounds, $\xi_d > \xi_d^{\rm max}$ or $\xi_d < \xi_d^{\rm min}$, then the velocity of the component d is reflected, i.e., $v_d \to -v_d$.

The parameter $A_{\rm cv}^{\rm max}$ sets the upper limit of the free energy. If $A_{\rm cv}$ exceeds $A_{\rm cv}^{\rm max}$, then the velocity is reversed, i.e., $v_d \to -v_d$ for all d.

Simulation setup.. Here is a recipe to start LogMFD simulations.

1. Decide the dimension of the collective variables, D. Then, for each collective variable $1 \le d \le D$, choose the type that properly describes the chemical reaction. So far, five types are implemented, i.e.,

   (1) bond distance of atoms A–B,
   (2) bond angle associated with atoms A–B–C,
   (3) dihedral angle associated with atoms A–B–C–D,
   (4) difference between A–B and B–C distances,
   (5) coordination number between A and B species.

   Set the collective variables using the keyword **<ncons>**.
2. Set the LogMFD type, **<logmfd_type>** = `NVE, NVT, VS`.
3. Set the LogMFD parameter α' ($\alpha_{\log} = \alpha'/k_B T$) using the keyword **<alpha_logmfd>** ($\gamma_{\log}$ can be independently set using **<gamma_logmfd>**, but normally the default setting, $\gamma_{\log} = 1/\alpha_{\log}$, is sufficient).
4. Set the reference displacement, $\Delta\xi_d^{\rm ref}$, as well as the lower bound $\xi_d^{\rm min}$ and the upper bound $\xi_d^{\rm max}$, for each type of collective variable using the keyword **<params_afed>**. Note that $\Delta\xi_d^{\rm ref}$ is used to determine μ_d using Eq.(243). Additionally, one can use **<mass_afed>** to further tune μ_d.
5. Set the force constant of the harmonic potential, κ_d, using the keyword **<params_cons>**.
6. Optionally, the origin of the free energy [$A_{\rm cv}(0)$], the maximum free energy value ($A_{\rm cv}^{\rm max}$) to be sampled, and the AFED step size $\Delta t_{\rm cv}$ can be manually set using **<fenergy_ini_logmfd>**, **<fenergy_max_afed>**, and **<dt_logmfd>**, respectively. Also, the mass of the thermostat should be set using **<niter_bath_afed>** for **<logmfd_type>** = `NVT`.

13.17 Nonadiabatic dynamics

In nonadiabatic dynamics, the time-dependent coefficients of the adiabatic electronic states, $C_k(t)$, are introduced.

$$|\Psi(t)\rangle = \sum_k^{\text{state}} C_k(t) \left|\Phi_k(\mathbf{r}(t))\right\rangle \tag{245}$$

where $\Psi(t)$ is the total electronic wave function and $\Phi_k(\mathbf{r}(t))$ is the adiabatic wave function at time t, which depends on t implicitly through the instantaneous nuclear configuration $\mathbf{r}(t)$. The adiabatic states are defined as the eigenvectors of the Born-Oppenheimer Hamiltonian:

$$\hat{H}_{\text{BO}} \left|\Phi_k(\mathbf{r}(t))\right\rangle = V_k(\mathbf{r}(t)) \left|\Phi_k(\mathbf{r}(t))\right\rangle, \tag{246}$$

where the eigenvalue

$$V_k(\mathbf{r}(t)) = \langle\Phi_k(\mathbf{r}(t))|\hat{H}_{\text{BO}}(\mathbf{r}(t))|\Phi_k(\mathbf{r}(t))\rangle \tag{247}$$

corresponds to the adiabatic potential energy of the k-th state. The density matrix is defined as

$$\rho_{kl}(t) = C_k^*(t) C_l(t), \tag{248}$$

and its diagonal element, $\rho_{kk}(t)$, represents the population of the k-th state.

The electronic equations of motion are given by

$$i\hbar \dot{C}_l(t) = C_l(t) V_l(\mathbf{r}(t)) - i\hbar \sum_{k\neq l}^{\text{state}} C_k(t) \sum_{j=1}^{N} \sum_{\alpha=x,y,z} \dot{r}_{j\alpha}(t) \cdot d_{j\alpha,kl}(\mathbf{r}(t)), \tag{249}$$

where

$$d_{j\alpha,kl}(\mathbf{r}(t)) = \left\langle \Phi_k(\mathbf{r}(t)) \right| \frac{\partial}{\partial r_{j\alpha}} \left| \Phi_l(\mathbf{r}(t)) \right\rangle \tag{250}$$

is the nonadiabatic coupling vector.

Mean field dynamics. In Ehrenfest mean-field dynamics, the nuclei evolve under the average force weighted by the state populations. The nuclear equations of motion are expressed as

$$\dot{p}_{j\alpha}(t) = -\sum_k^{\text{state}} |C_k(t)|^2 \frac{\partial V_k(\mathbf{r}(t))}{\partial r_{j\alpha}} - \sum_k^{\text{state}} \sum_l^{\text{state}} C_l^*(t) C_k(t) \left[V_l(\mathbf{r}(t)) - V_k(\mathbf{r}(t))\right] \cdot d_{lk,j\alpha}(\mathbf{r}(t)). \tag{251}$$

Surface hopping dynamics. In Tully's surface hopping dynamics, the nuclei propagate on the potential energy surface of the currently occupied state, $k(t)$. The nuclear equations of motion are expressed as

$$\dot{p}_{j\alpha}(t) = -\frac{\partial V_{k(t)}(\mathbf{r}(t))}{\partial r_{j\alpha}}. \tag{252}$$

Here, $k(t)$ is a stochastic variable that evolves in time. According to the fewest switches algorithm, the probability for a transition (hop) from state l to state k during a time step Δt is given by

$$P^{l\to k}(t) = \max\left[\frac{f_{kl}(t)}{\rho_{ll}(t)}, 0\right] \Delta t, \tag{253}$$

where

$$f_{kl}(t) = -2\,\text{Re}\left[\rho_{kl}(t) \left(\sum_{j=1}^{N} \sum_{\alpha=x,y,z} \dot{r}_{j\alpha}(t) \cdot d_{j\alpha,kl}(\mathbf{r}(t))\right)\right]. \tag{254}$$

13.18 Hybrid potentials: ONIOM and QM/MM

In the ONIOM method, the total potential is composed of three parts

$$V^{\mathrm{ONIOM}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}) \quad = \quad V^{\mathrm{hi}}_{\mathrm{AL}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{L}}) - V^{\mathrm{lo}}_{\mathrm{AL}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{L}}) + V^{\mathrm{lo}}_{\mathrm{AB}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}). \tag{255}$$

The first two terms describe the energy difference of the subsystem consisting of layer A and the link atoms L, calculated at high- and low-level methods. The last term is the energy of the system composed of layers A and B (the whole system), calculated at a low-level method. As shown below, the positions of the link atoms are determined once the positions of the bridging atoms are fixed. The ONIOM energy has the correct limits when the high- and low-level methods are identical,

$$\lim_{\mathrm{hi}\to\mathrm{lo}} V^{\mathrm{ONIOM}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}) \quad = \quad V^{\mathrm{lo}}_{\mathrm{AB}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}) \tag{256}$$

and

$$\lim_{\mathrm{lo}\to\mathrm{hi}} V^{\mathrm{ONIOM}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}) \quad = \quad V^{\mathrm{hi}}_{\mathrm{AB}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}). \tag{257}$$

In the absence of link atoms, it can be shown that the interaction between layer A and layer B is given by the low-level energy. Formally, the A–B interaction can be expressed as

$$V^{\mathrm{lo}}_{int}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}) \quad = \quad V^{\mathrm{lo}}_{\mathrm{AB}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}) - V^{\mathrm{lo}}_{\mathrm{A}}(\mathbf{r}_{\mathrm{A}}) - V^{\mathrm{lo}}_{\mathrm{B}}(\mathbf{r}_{\mathrm{B}}), \tag{258}$$

and thus, the total potential energy can be rewritten as

$$V^{\mathrm{ONIOM}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}) \quad = \quad V^{\mathrm{hi}}_{\mathrm{A}}(\mathbf{r}_{\mathrm{A}}) + V^{\mathrm{lo}}_{\mathrm{B}}(\mathbf{r}_{\mathrm{B}}) + V^{\mathrm{lo}}_{int}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}). \tag{259}$$

This clearly indicates that the ONIOM energy is composed of the high-level energy of layer A, the low-level energy of layer B, and the low-level energy of the interaction between A and B.

The mechanical embedding version of the QM/MM energy is a special case of the ONIOM energy, where the high-level method corresponds to QM and the low-level calculation corresponds to MM:

$$V^{\mathrm{QM/MM}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}) \quad = \quad V^{\mathrm{QM}}_{\mathrm{AL}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{L}}) - V^{\mathrm{MM}}_{\mathrm{AL}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{L}}) + V^{\mathrm{MM}}_{\mathrm{AB}}(\mathbf{r}_{\mathrm{A}},\mathbf{r}_{\mathrm{B}}). \tag{260}$$

Let the link atom, l, be placed along the bond i–j, between atom i in layer A and atom j in layer B. Let the ratio of i–l and i–j interatomic distances be a constant,

$$\alpha = \frac{\left|\mathbf{r}_i - \mathbf{r}_l\right|}{\left|\mathbf{r}_i - \mathbf{r}_j\right|}. \tag{261}$$

Then, the position of the link atom is determined as

$$\mathbf{r}_l = \mathbf{r}_i + \alpha(\mathbf{r}_j - \mathbf{r}_i) \tag{262}$$

once the positions of atoms i and j, $\mathbf{r}_i$ and $\mathbf{r}_j$, are fixed. In this case, the forces acting on atoms i and j are

$$\mathbf{F}_i = -\frac{\partial V^{\mathrm{ONIOM}}}{\partial \mathbf{r}_i} - (1-\alpha)\frac{\partial (V^{\mathrm{hi}}_{\mathrm{AL}} - V^{\mathrm{lo}}_{\mathrm{AL}})}{\partial \mathbf{r}_l} \tag{263}$$

and

$$\mathbf{F}_j = -\frac{\partial V^{\mathrm{ONIOM}}}{\partial \mathbf{r}_j} - \alpha\frac{\partial (V^{\mathrm{hi}}_{\mathrm{AL}} - V^{\mathrm{lo}}_{\mathrm{AL}})}{\partial \mathbf{r}_l}, \tag{264}$$

respectively. Note that the sum of forces

$$-\sum_{i=1}^{N}\left(\frac{\partial V^{\mathrm{ONIOM}}}{\partial \mathbf{r}_i}\right) = \mathbf{0} \tag{265}$$

and the total torque

$$-\sum_{i=1}^{N}\left(\mathbf{r}_i \times \frac{\partial V^{\mathrm{ONIOM}}}{\partial \mathbf{r}_i}\right) = \mathbf{0} \tag{266}$$

are preserved.

13.19 Multiple time scale method

The multiple time scale (MTS) method is implemented in the PIMD code. This technique allows for efficient sampling in molecular dynamics (MD) and path integral molecular dynamics (PIMD) when a hybrid potential, such as QM/MM or ONIOM, is employed.

In general, the MTS method is an algorithm to integrate the equations of motion using different time steps for forces that vary on different time scales. For two-layer systems, the MTS method is applicable only under the following conditions:

- Only properties at thermal equilibrium are of interest. Nonequilibrium properties, such as time-correlation functions, are not considered.
- Layer A is much smaller than layer B, so that the phase space of layer A is much smaller than that of layer B.
- Force calculations for layer A are significantly more expensive than those for layer B.

Under these conditions, the MTS algorithm is suitable.

- **Mass scaling:** Phase space sampling of layer B can be accelerated by scaling the masses of its atoms. In classical MD, all masses in layer B are scaled by a factor n_{ref}^{-2}. Although this mass scaling alters dynamical information, it does not affect thermodynamic properties, because the classical NVT ensemble distribution, $\propto \exp(-\beta V)$, remains unchanged. For PIMD, the fictitious masses of layer B can also be scaled without changing the quantum NVT ensemble distribution, $\propto \exp(-\beta V^{\mathrm{eff}})$.
- **Force decomposition:** In the two-layer ONIOM method and mechanical embedding QM/MM, the total force can be decomposed as

$$-\nabla V_{\mathrm{ONIOM}}(\mathbf{r}_{\mathrm{A}}, \mathbf{r}_{\mathrm{B}}) = -\nabla V_{\mathrm{high}}(\mathbf{r}_{\mathrm{A}}, \mathbf{r}_{\mathrm{B}}) + \nabla V_{\mathrm{low}}(\mathbf{r}_{\mathrm{A}}, \mathbf{r}_{\mathrm{B}}) - \nabla V_{\mathrm{low}}(\mathbf{r}_{\mathrm{A}}, \mathbf{r}_{\mathrm{B}}). \quad (267)$$

  To utilize the MTS algorithm, the first term on the right-hand side is updated with the larger time step, Δt, while the second and third terms are updated more frequently, using a smaller time step, $\Delta t / n_{\mathrm{ref}}$. Rapidly varying forces in layer B require more frequent updates than those in layer A.

The basic MD cycle is then constructed as follows:

```
[1] force of system A at high level
[2] force of system A at low level
[3] force of system B at low level

do i = 1, nstep

  update velocity due to forces [2] by -dt/2
  update velocity due to forces [3] by +dt/2

  do j = 1, nref

     update velocity due to force [1] by dt/nref/2

     update position by dt/nref

     [1] force of system A at high level

     update velocity due to force [1] by dt/nref/2

  end do

  [2] force of system A at low level
  [3] force of system B at low level

  update velocity due to forces [2] by -dt/2
  update velocity due to forces [3] by +dt/2
```

```
end do
```

Here, updates of thermostats, harmonic forces in PIMD, and other details have been omitted for simplicity. The parameter `nref` can be increased until the computational costs of the inner and outer cycles are balanced.
Note: The MTS method can be combined with the BEST method.

13.20 BEST method

The BEST (Boundary based on Exchange Symmetry Theory) method is a treatment of hybrid potentials for open boundary systems [53]. A bias potential is imposed to enforce a natural phase separation of the layers introduced in hybrid potentials. The form of the bias potential has been determined according to the exchange symmetry between the atoms in different layers. The bias potential is designed to have no influence on the thermodynamics of the system if the atoms were identical. (Of course, the atoms in different layers are not identical in any hybrid potentials, but at least the bias potential preserves the correct limit as if they were identical.) The details will be described elsewhere [53].

13.21 Ewald sum

Let us consider the electrostatic interaction of N particles with charges q_i at positions $\mathbf{r}_i$ for $1 \le i \le N$, confined in a parallelepiped box under periodic boundary conditions. For a vector of three integers, $\mathbf{n} = (n_1, n_2, n_3)$, the lattice vectors are defined by

$$(\mathbf{R}_\mathbf{n})_\alpha = \sum_{\beta=1}^{3} h_{\alpha\beta} n_\beta \tag{268}$$

where $\alpha = (x, y, z)$, $\beta = (1, 2, 3)$, and $\mathbf{h}$ is the box matrix. Then, the electrostatic interaction is

$$V^{\mathrm{es}} = \sum_{\mathbf{n}} \sum_{j=1}^{N} \sum_{k=1}^{N}{}' \frac{q_j q_k}{\left|\mathbf{r}_j - \mathbf{r}_k + \mathbf{R}_\mathbf{n}\right|}. \tag{269}$$

Here, $\sum'$ indicates that the contribution of $k = j$ for the special case of $\mathbf{n} = \mathbf{0}$ has been excluded from the sum to avoid self-interactions. Meanwhile, for a vector of three integers, $\mathbf{k} = (k_1, k_2, k_3)$, the reciprocal lattice vector is defined by

$$(\mathbf{G}_\mathbf{k})_\beta = 2\pi \sum_{\alpha=1}^{3} k_\alpha h^{-1}_{\alpha\beta}. \tag{270}$$

Then, the Ewald sum can be expressed as

$$V^{\mathrm{es}} = V^{\mathrm{short}} + V^{\mathrm{long}} + V^{\mathrm{self}} + V^{\mathrm{charge}} (+V^{\mathrm{dipole}}) \tag{271}$$

where the short-range, real-space contribution is

$$V^{\mathrm{short}} = \frac{1}{2} \sum_{\mathbf{n}} \sum_{j=1}^{N} \sum_{k \ne j}^{N}{}' \frac{q_j q_k \mathrm{erfc}\left(\eta \left|\mathbf{r}_j - \mathbf{r}_k + \mathbf{R}_\mathbf{n}\right|\right)}{\left|\mathbf{r}_j - \mathbf{r}_k + \mathbf{R}_\mathbf{n}\right|}, \tag{272}$$

the long-range, reciprocal-space contribution is

$$V^{\mathrm{long}} = \frac{2\pi}{V_{\mathrm{cell}}} \sum_{\mathbf{k} \ne \mathbf{0}} \frac{\exp\left(-\frac{|\mathbf{G}_\mathbf{k}|^2}{4\eta^2}\right)}{|\mathbf{G}_\mathbf{k}|^2} \left| \sum_{j=1}^{N} q_j e^{i\left(\mathbf{G}_\mathbf{k} \cdot \mathbf{r}_j\right)} \right|^2, \tag{273}$$

the correction of self-interaction in the reciprocal-space contribution is

$$V^{\mathrm{self}} = -\frac{\eta}{\sqrt{\pi}} \sum_{i=1}^{N} q_i^2, \tag{274}$$

and the correction for a charged system is

$$V^{\text{charge}} = -\frac{\pi}{2\eta^2 V_{\text{cell}}}\left(\sum_{i=1}^{N} q_i\right)^2, \tag{275}$$

The parameter η can be determined so that the calculation is both accurate and efficient by balancing the computational cost between the real-space and reciprocal-space contributions. The elimination of the $\mathbf{k} = \mathbf{0}$ contribution from the long-range term corresponds to the introduction of a uniform background charge that maintains the charge neutrality of the system. The dipole correction is

$$V^{\text{dipole}} = \frac{2\pi}{3V_{\text{cell}}}\left|\sum_{i=1}^{N} q_i \mathbf{r}_i\right|^2. \tag{276}$$

In the PIMD code, this term can be either included or neglected by option.

13.22 Polarizable force field

The polarizable force field is described by

$$V_{\text{pol}} = V_{\text{mm}} + V_{\text{ind}} + V_{\text{cd}} + V_{\text{dd}}, \tag{277}$$

where V_{mm} is the non-polarizable molecular mechanics contribution represented by the bonded, nonbonded, and electrostatic terms selected through CFF (Section 10). With the dipole moment $\boldsymbol{\mu}_i$ and the polarizability α_i of each atom i, the polarization energy is given by

$$V_{\text{ind}} = \sum_{i=1}^{N} \frac{\boldsymbol{\mu}_i^2}{2\alpha_i}. \tag{278}$$

Under free boundary conditions, the charge–dipole interaction is described by

$$V_{\text{cd}} = \sum_{i=1}^{N}\sum_{j\neq i}^{N} q_i \mathbf{t}_{ij}\boldsymbol{\mu}_j = \sum_{i=1}^{N}\sum_{j>i}^{N}\left(q_i \mathbf{t}_{ij}\boldsymbol{\mu}_j - q_j \mathbf{t}_{ij}\boldsymbol{\mu}_i\right), \tag{279}$$

and the dipole–dipole interaction by

$$V_{\text{dd}} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N} \boldsymbol{\mu}_i \mathbf{T}_{ij}\boldsymbol{\mu}_j = -\sum_{i=1}^{N}\sum_{j>i}^{N} \boldsymbol{\mu}_i \mathbf{T}_{ij}\boldsymbol{\mu}_j. \tag{280}$$

Here, we have defined a vector

$$\left(\mathbf{t}_{ij}\right)_\alpha = \frac{s_3^{\text{cd}}\left(r_{ij}\right)}{r_{ij}^3}\left(\mathbf{r}_{ij}\right)_\alpha, \tag{281}$$

and a matrix

$$\left(\mathbf{T}_{ij}\right)_{\alpha\beta} = -\frac{s_3^{\text{dd}}\left(r_{ij}\right)}{r_{ij}^3}\delta_{\alpha\beta} + \frac{3s_5^{\text{dd}}\left(r_{ij}\right)}{r_{ij}^5}\left(\mathbf{r}_{ij}\right)_\alpha\left(\mathbf{r}_{ij}\right)_\beta. \tag{282}$$

These quantities are functions of the interatomic separation $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and the interatomic distance $r_{ij} = |\mathbf{r}_{ij}|$. The damping functions $s_3^{\text{cd}}(r)$, $s_3^{\text{dd}}(r)$, and $s_5^{\text{dd}}(r)$ monotonically increase from 0 to 1 with increasing r. The dipole moments are determined by the variational condition

$$\frac{\partial V^{\text{pol}}}{\partial \boldsymbol{\mu}_i} = 0. \tag{283}$$

Thus,

$$\boldsymbol{\mu}_i = \alpha_i \sum_{j\neq i}^{N}\left(-\mathbf{t}_{ij} q_j + \mathbf{T}_{ij}\boldsymbol{\mu}_j\right). \tag{284}$$

Under periodic boundary conditions, the polarizable force field is described by

$$V_{\mathrm{pol}} = V_{\mathrm{mm}} + V_{\mathrm{ind}} + V_{\mathrm{cd}}^{\mathrm{short}} + V_{\mathrm{cd}}^{\mathrm{long}} + V_{\mathrm{dd}}^{\mathrm{short}} + V_{\mathrm{dd}}^{\mathrm{long}} + V_{\mathrm{dd}}^{\mathrm{self}} + V_{\mathrm{dd}}^{\mathrm{net}}. \tag{285}$$

The real-space contributions of the charge–dipole and dipole–dipole interactions are given by

$$V_{\mathrm{cd}}^{\mathrm{short}} = \sum_{n} \sum_{i=1}^{N} \sum_{j=1}^{N}{}' q_i \mathbf{t}_{ij}^{\mathrm{short}} \boldsymbol{\mu}_j, \tag{286}$$

and

$$V_{\mathrm{dd}}^{\mathrm{short}} = \frac{1}{2} \sum_{n} \sum_{i=1}^{N} \sum_{j=1}^{N}{}' \boldsymbol{\mu}_i \mathbf{T}_{ij}^{\mathrm{short}} \boldsymbol{\mu}_j, \tag{287}$$

respectively, where we have defined a vector

$$\left(\mathbf{t}_{ij}^{\mathrm{short}}\right)_{\alpha} = \left(\frac{s_3^{\mathrm{cd}}\left(r_{ij}\right) - \mathrm{erf}(\eta r_{ij}) + \frac{2\eta r_{ij}}{\sqrt{\pi}} e^{-\eta^2 r_{ij}^2}}{r_{ij}^3} \right) \left(\mathbf{r}_{ij}\right)_{\alpha}, \tag{288}$$

and a matrix

$$\begin{aligned} \left(\mathbf{T}_{ij}^{\mathrm{short}}\right)_{\alpha\beta} = & - \left(\frac{s_3^{\mathrm{dd}}\left(r_{ij}\right) - \mathrm{erf}(\eta r_{ij}) + \frac{2\eta r_{ij}}{\sqrt{\pi}} e^{-\eta^2 r_{ij}^2}}{r_{ij}^3} \right) \delta_{\alpha\beta} \\ & + \left(\frac{3\left(s_5^{\mathrm{dd}}\left(r_{ij}\right) - \mathrm{erf}\left(\eta r_{ij}\right)\right) + \left(3 + 2\eta^2 r_{ij}^2\right) \frac{2\eta r_{ij}}{\sqrt{\pi}} e^{-\eta^2 r_{ij}^2}}{r_{ij}^5} \right) \left(\mathbf{r}_{ij}\right)_{\alpha} \left(\mathbf{r}_{ij}\right)_{\beta}. \end{aligned} \tag{289}$$

The reciprocal-space contributions of the charge–dipole and dipole–dipole interactions are given by

$$V_{\mathrm{cd}}^{\mathrm{long}} = \frac{4\pi}{V_{\mathrm{cell}}} \sum_{\mathbf{k}\neq\mathbf{0}} \frac{\exp\left(-\frac{|\mathbf{G}_{\mathbf{k}}|^2}{4\eta^2}\right)}{|\mathbf{G}_{\mathbf{k}}|^2} \sum_{i=1}^{N} q_i \sum_{j=1}^{N} (\boldsymbol{\mu}_j \cdot \mathbf{G}_{\mathbf{k}}) \sin\left(\mathbf{G}_{\mathbf{k}} \cdot \mathbf{r}_{ij}\right), \tag{290}$$

and

$$V_{\mathrm{dd}}^{\mathrm{long}} = \frac{2\pi}{V_{\mathrm{cell}}} \sum_{\mathbf{k}\neq\mathbf{0}} \frac{\exp\left(-\frac{|\mathbf{G}_{\mathbf{k}}|^2}{4\eta^2}\right)}{|\mathbf{G}_{\mathbf{k}}|^2} \left| \sum_{j=1}^{N} (\boldsymbol{\mu}_j \cdot \mathbf{G}_{\mathbf{k}}) e^{i\mathbf{G}_{\mathbf{k}} \cdot \mathbf{r}_j} \right|^2, \tag{291}$$

respectively. The dipole self-interaction correction in the reciprocal-space contribution is

$$V_{\mathrm{dd}}^{\mathrm{self}} = -\frac{2\eta^3}{3\sqrt{\pi}} \sum_{i=1}^{N} \boldsymbol{\mu}_i^2. \tag{292}$$

The correction with respect to the net dipole moment is

$$V_{\mathrm{dd}}^{\mathrm{net}} = \frac{2\pi}{3V_{\mathrm{cell}}} \left| \sum_{i=1}^{N} \boldsymbol{\mu}_i \right|^2. \tag{293}$$

The dipole moments $\boldsymbol{\mu}_i$ are obtained from the variational condition

$$\frac{\partial V^{\mathrm{pol}}}{\partial \boldsymbol{\mu}_i} = 0, \tag{294}$$

and thus

$$\boldsymbol{\mu}_i = \alpha_i \left(- \sum_{j\neq i}^{N} \left(\mathbf{t}_{ij}^{\mathrm{short}} q_j - \mathbf{T}_{ij}^{\mathrm{short}} \boldsymbol{\mu}_j \right) - \frac{\partial V_{\mathrm{cd}}^{\mathrm{long}}}{\partial \boldsymbol{\mu}_i} - \frac{\partial V_{\mathrm{dd}}^{\mathrm{long}}}{\partial \boldsymbol{\mu}_i} + \frac{4\eta^3}{3\sqrt{\pi}} \boldsymbol{\mu}_i - \frac{4\pi}{3V_{\mathrm{cell}}} \sum_{j=1}^{N} \boldsymbol{\mu}_j \right), \tag{295}$$

which leads to a set of $3N$-dimensional linear equations of the form $\mathbf{A}\boldsymbol{\mu} = \mathbf{b}$. This system can be solved using the LAPACK `dgesv` routine.

The damping functions can be chosen from the linear Thole (LIN), exponential Thole (EXP), Gaussian (GAU) [88], and Ojamae–Shavitt–Singer (OSS) types [89]. The OSS-type damping function is defined with two parameters, denoted by a_x^y and b_x^y, as

$$s_1^{\mathrm{cc}} = \frac{r^2}{r^2 + a_1^{\mathrm{cc}} e^{-b_1^{\mathrm{cc}} r}}, \quad s_3^{\mathrm{cd/dd}} = \frac{r^2}{r^2 + a_3^{\mathrm{cd/dd}} e^{-b_3^{\mathrm{cd/dd}} r}}, \quad s_3^{\mathrm{dd}} = \frac{r^2}{r^2 + a_3^{\mathrm{dd}} e^{-b_3^{\mathrm{dd}} r}}. \tag{296}$$

Other types use a single parameter a, and the functions are described in terms of $z = r/a$. The linear Thole-type damping functions are given by

$$s_1^{\mathrm{cc}} = \begin{cases} 2z - 2z^3 + z^4 (0 \le z \le 1) \\ 1 (z > 1) \end{cases} \quad s_3^{\mathrm{cd/dd}} = \begin{cases} 4z^3 - 3z^4 (0 \le z \le 1) \\ 1 (z > 1) \end{cases} \quad s_5^{\mathrm{dd}} = \begin{cases} z^4 (0 \le z \le 1) \\ 1 (z > 1). \end{cases} \tag{297}$$

The exponential Thole-type damping functions are given by

$$s_1^{\mathrm{cc}} = 1 - e^{-z^3} + z\Gamma\left(\frac{2}{3}, z^3\right), \quad s_3^{\mathrm{cd/dd}} = 1 - e^{-z^3}, \quad s_5^{\mathrm{dd}} = 1 - (1 + z^3) e^{-z^3}, \tag{298}$$

using the incomplete gamma function

$$\Gamma(a, x) = \int_x^\infty e^{-t} t^{a-1} dt. \tag{299}$$

The Gaussian-type damping functions are given by

$$s_1^{\mathrm{cc}} = \mathrm{erf}(z), \quad s_3^{\mathrm{cc/dd}} = \mathrm{erf}(z) - \frac{2z}{\sqrt{\pi}} e^{-z^2}, \quad s_5^{\mathrm{cc}} = \mathrm{erf}(z) - \left(1 + \frac{2z^2}{3}\right) \frac{2z}{\sqrt{\pi}} e^{-z^2}. \tag{300}$$

13.23 Dihedral bond potential

For the dihedral form used by the classical force field, the dihedral angle potential takes the form

$$V = \frac{v}{2} \{1 + \mu \cos(\nu\chi)\} \tag{301}$$

where v is the rotational barrier height, $\mu = \pm 1$ is the phase, and χ is the dihedral angle with respect to atoms i, j, k, and l, which can be expressed as

$$\chi = \mathrm{sgn}\left[\left(\mathbf{r}_{ijk} \times \mathbf{r}_{jkl}\right) \cdot \mathbf{r}_{jk}\right] \cos^{-1}\left(\frac{\mathbf{r}_{ijk} \cdot \mathbf{r}_{jkl}}{\left|\mathbf{r}_{ijk}\right| \left|\mathbf{r}_{jkl}\right|}\right) \tag{302}$$

with the vectors normal to the $i - j - k$ and $j - k - l$ planes given by

$$\mathbf{r}_{ijk} = \mathbf{r}_{ij} \times \mathbf{r}_{kj}, \quad \mathbf{r}_{jkl} = \mathbf{r}_{lj} \times \mathbf{r}_{kj}. \tag{303}$$

The gradients of V with respect to each atom can be expressed as

$$\frac{\partial V}{\partial \mathbf{r}_i} = \frac{v\mu\nu s_\nu}{2} \left[\frac{\mathbf{r}_{kj} \times \mathbf{r}_{jkl}}{\left|\mathbf{r}_{ijk}\right| \left|\mathbf{r}_{jkl}\right|} - \frac{\mathbf{r}_{kj} \times \mathbf{r}_{ijk}}{\left|\mathbf{r}_{ijk}\right|^2} \cos\chi\right], \tag{304}$$

$$\frac{\partial V}{\partial \mathbf{r}_j} = \frac{v\mu\nu s_\nu}{2} \left[\left(\frac{\mathbf{r}_{kl} \times \mathbf{r}_{jkl}}{\left|\mathbf{r}_{jkl}\right|^2} + \frac{\mathbf{r}_{ki} \times \mathbf{r}_{ijk}}{\left|\mathbf{r}_{ijk}\right|^2}\right) \cos\chi - \frac{\mathbf{r}_{kl} \times \mathbf{r}_{ijk} + \mathbf{r}_{ik} \times \mathbf{r}_{jkl}}{\left|\mathbf{r}_{ijk}\right| \left|\mathbf{r}_{jkl}\right|}\right], \tag{305}$$

$$\frac{\partial V}{\partial \mathbf{r}_k} = \frac{v\mu\nu s_\nu}{2}\left[\left(\frac{\mathbf{r}_{ij}\times\mathbf{r}_{ijk}}{\left|\mathbf{r}_{ijk}\right|^2}+\frac{\mathbf{r}_{lj}\times\mathbf{r}_{jkl}}{\left|\mathbf{r}_{jkl}\right|^2}\right)\cos\chi-\frac{\mathbf{r}_{ij}\times\mathbf{r}_{jkl}+\mathbf{r}_{lj}\times\mathbf{r}_{ijk}}{\left|\mathbf{r}_{ijk}\right|\left|\mathbf{r}_{jkl}\right|}\right], \tag{306}$$

and

$$\frac{\partial V}{\partial \mathbf{r}_l} = \frac{v\mu\nu s_\nu}{2}\left[\frac{\mathbf{r}_{kj}\times\mathbf{r}_{ijk}}{\left|\mathbf{r}_{ijk}\right|\left|\mathbf{r}_{jkl}\right|}-\frac{\mathbf{r}_{kj}\times\mathbf{r}_{jkl}}{\left|\mathbf{r}_{jkl}\right|^2}\cos\chi\right], \tag{307}$$

where the function

$$s_\nu = \frac{1}{\nu}\frac{d\cos(\nu\chi)}{d\cos\chi} = \frac{1}{\nu}\frac{d\cos(\nu\chi)}{d\chi}\left(\frac{d\cos\chi}{d\chi}\right)^{-1} = \frac{\sin(\nu\chi)}{\sin\chi} \tag{308}$$

can be calculated using the recursive relation

$$s_\nu = s_{\nu-2} - \sin((\nu-2)\chi)\sin\chi + \cos((\nu-2)\chi)\cos\chi + \cos((\nu-1)\chi) \tag{309}$$

with $s_1 = 1$ and $s_2 = 2\cos\chi$.

Part VII

Publications and References

14 Publications

The following is the list of publications in which the PIMD code or its primitive version has been used.

- K. Nishioka, S. Dekura, T. Fujino, M. Mizuno, R. Kumai, B. Thomsen, M. Shiga, Y. Hori, Y. Shigeta, H. Mori, Angew. Chem. Int. Ed., 65, e20785 (2026). "Proton Tautomerism for Anhydrous Superprotonic Conduction in 1,2,3-Triazolium Dihydrogen Phosphate Crystal"
- Y. Liu, J. Lu, X. Yuan, X. Liu, F. Tian, X. Li, Y. Ma, P. Hou, M. Pang, Physica B: Condens. Matter 734, 418596 (2026). "Prediction of isotope influenced negative thermal expansion in 14C diamond via PIMD simulation"
- M. Shiga, J. Elsner, J. Behler, B. Thomsen, J. Chem. Phys., 163, 134119 (2025). "Computation of the Heat Capacity of Water from First Principles"
- H. Kimizuka, S. Ogata, B. Thomsen, M. Shiga, J. Phys. Condens. Matter, 37, 193001 (2025). "Topical Review: Ab initio path-integral simulations of hydrogen-isotope diffusion in face-centered cubic metals"
- H. Matsubuchi, D. Hayashi, D. Okamoto, A. Noguchi, S. Nakagawa, T. Takayanagi, T. Murakami, ChemPhysChem 26 e202500494 (2025). "Solvent Effects on the Selectivity of Ambimodal Dipolar/Diels–Alder Cycloadditions: A Study Using Explicit Solvation Models"
- T. Murakami, H. Ota, S. Nakagawa, K. Okada, M. Tachikawa, T. Takayanagi, J. Phys. Chem. A, 29 2308 (2025). "Attractive Force-Induced Isotope Effects through Ring-Polymer Molecular Dynamics Simulations for the Barrierless Reaction between HNCO and H_3^+ Isotopologues: H_3^+, H_2D^+, HD_2^+, and D_3^+"

- T. Murakami, H. Ueno, Y. Kikuma, T. Takayanagi, Molecules 30, 442 (2025). "Nuclear Quantum Effects in the Ionic Dissociation Dynamics of HCl on the Water Ice Cluster"
- B. Thomsen, Y. Nagai, K. Kobayashi, I. Hamada, M. Shiga, J. Chem. Phys., 161, 204109 (2024). "Self-Learning Path Integral Hybrid Monte Carlo with Mixed Ab Initio and Machine Learning Potentials for Modelling Nuclear Quantum Effects in Water"
- H. Li, Y. Kataoka, S. Tanaka, J. Haruyama, O. Sugino, J. Yoshinobu, J. Phys. Chem. C, 128, 15393 (2024). "Elucidation of CO Oxidation and CO_2 Desorption Dynamics on Pt(111) by van der Waals DFT Calculations: Hyperthermal KineticEnergy, Sharp Desorption Angle, and Excited Vibrational States"
- Y. Kataoka, J. Haruyama, O. Sugino, M. Shiga, Phys. Rev. Res., 6, 043224 (2024). "Predictive evaluation of hydrogen diffusion coefficient on Pd(111) surface by path integral simulations using neural network potential"
- J. Tsuchiya, M. Shiga, S. Tsuneyuki, E. C. Thompson, Phys. Rev. Res., 6, 023302 (2024). "Nuclear quantum effect on the elasticity of ice VII under pressure: A path-integral molecular dynamics study"
- Y. Nagai, Y. Iwasaki, K. Kitahara, Y. Takagiwa, K. Kimura, M. Shiga, Phys. Rev. Lett., 132, 196301 (2024). "High-temperature atomic diffusion and specific heat in quasicrystals"
- Y. Nakata, T. Sasaki, B. Thomsen, M. Shiga, Chem. Phys. Lett., 845, 141285 (2024). "Theoretical Study of Cellobiose Conversion by Supported Metal Catalysts"
- M. Shiga, B. Thomsen, H. Kimizuka, Phys. Rev. B. 109, 054303 (2024). "Inelastic Neutron Scattering of Hydrogen in Palladium Studied by Semiclassical Dynamics"
- T. Murakami, Y. Kikuma, D. Hayashi, S. Ibuki, S. Nakagawa, H. Ueno, T. Takayanagi, J. Phys. Org. Chem., 37, e4611 (2024). "Molecular dynamics simulation study of post-transition state bifurcation: A case study on the ambimodal transition state of dipolar/Diels-Alder cycloaddition"
- T. Murakami, S. Takahashi, Y. Kikuma, T. Takayanagi, Molecules, 29, 2789 (2024). "Theoretical Study of the Thermal Rate Coefficients of the $H_3^+ + C_2H_4$ Reaction: Dynamics Study on a Full-Dimensional Potential Energy Surface"
- T. Murakami, T. Takayanagi, Phys. Chem. Chem. Phys., 26, 19195 (2024). "Computational study of the post-transition state dynamics for the OH + CH_3OH reaction probed by photodetachment of the $CH_3O^-(H_2O)$ anion"
- T. Murakami, D. Hayashi, Y. Kikuma, K. Yamaki, T. Takayanagi, J. Comput. Chem., 45, 2778 (2024). "Temperature effects on the branching dynamics in the model ambimodal (6 + 4)/(4 + 2) intramolecular cycloaddition reaction"
- T. Murakami, N. Matsumoto, T. Takayanagi, Comput. Theor. Chem., 1240, 114835 (2024). "Computational study on the bifurcation mechanism in the $H_2CO_- + CH_3Cl \rightarrow CH_3CH_2O + Cl^- / H_2CO + CH_3 + Cl^-$ reaction: The importance of intramolecular vibrational redistributions"
- T. Murakami, S. Nakagawa, H. Ota, K. Okada,T. Takayanagi, ACS Earth Space Chem., 8(11), 2294 (2024). "Kinetics and Ring-Polymer Molecular Dynamics Studies of the $H_3^+ + HNCO \rightarrow H_2 + HNCOH^+ / H_2NCO^+$ Branching Reaction on a Developed Full-Dimensional Potential Energy Surface"
- Y. Hashimoto, T. Takayanagi, T. Murakami, ACS Earth Space Chem., 7, 623 (2023). "Theoretical Calculations of the Thermal Rate Coefficients for the Interstellar $NH_3^+ + H_2 \rightarrow NH_4^+ + H$ Reaction on a New Δ-Machine Learning Potential Energy Surface"
- T. Murakami, S. Ibuki, Y. Hashimoto,Y. Kikuma, T. Takayanagi, Phys. Chem. Chem. Phys., 25, 14016(2023). "Dynamics study of the post-transition-state-bifurcation process of the $(HCOOH)H^+ \rightarrow CO + H3O^+ / HCO^+ + H_2O$ dissociation: Application of machine-learning techniques"
- T. Murakami, S. Ibuki, T. Takayanagi, Comput. Theor. Chem., 1227, 114239 (2023). "Molecular dynamics simulations and machine-learning assisted study of the reaction path bifurcation: Application to the intramolecular Diels-Alder cycloaddition between cyclobutadiene and butadiene"
- T. Murakami, Y. Kikuma, S. Ibuki, N. Matsumoto, K. Ogino, Y. Hashimoto, T. Takayanagi, J. Phys. Org. Chem., 36, e4561 (2023). "Machine learning-assisted study of correlation between post-transition-state bifurcation and initial phase information at the ambimodal transition state"
- H. Wang, P. T. Salzbrenner, I. Errea, F. Peng, Z. Lu, H. Liu, L. Zhu, C. J. Pickard, Y. Yao, Nature Commun. 14, 1674 (2023). "Quantum structural fluxion in superconducting lanthanum polyhydride"
- H. Kwon, M. Shiga, H. Kimizuka, T. Oda, Acta Mater, 247, 118739 (2023). "Accurate Description of Hydrogen Diffusivity in bcc Metals Using Machine-Learning Moment Tensor Potentials and Path-Integral Methods"

- K. Ogino, Y. Hashimoto, T. Takayanagi, ChemPhysChem, in press (2023). Ring-polymer Molecular Dynamics Simulation for the Adsorption of H_2 on Ice Clusters $(H_2O)_n$ ($n = 8, 10$, and 12)"
- T. Murakami, R. Iida, Y. Hashimoto, Y. Takahashi, S. Takahashi, T. Takayanagi, J. Phys. Chem. A 126, 9244 (2022). "Ring-Polymer Molecular Dynamics and Kinetics for the $H^- + C_2H_2 \rightarrow H_2 + C_2H^-$ Reaction Using the Full-Dimensional Potential Energy Surface"
- H. Suzuki, T. Otomo, K. Ogino, Y. Hashimoto, T. Takayanagi, ACS Earth Space Chem. 6, 1390 (2022). "Nuclear Quantum Effects in H_2 Adsorption Dynamics on a Small Water Cluster Studied with Ring-Polymer Molecular Dynamics Simulations"
- M. Shiga, J. Comput. Chem., 43, 1864 (2022). "Path Integral Brownian Chain Molecular Dynamics: A Simple Approximation of Quantum Vibrational Dynamics"
- H. Kimizuka, B. Thomsen, M. Shiga, J. Phys. Energy 4, 034334 (2022) (13 pages). "Artificial neural network-based path integral simulations of hydrogen isotope diffusion in palladium"
- B. Thomsen, M. Shiga, Phys. Chem. Chem. Phys., 24, 10851-10859 (2022). "Structure of Liquid and Aqueous Water Isotopologues at Ambient Temperature from ab initio Path Integral Simulations"
- D. Akazawa, T. Sasaki, M. Nagasaka, M. Shiga, J. Chem. Phys., 56, 044202 (2022) (7 pages). "X-ray Absorption Spectra of Aqueous Cellobiose: Experiment and Theory"
- Y. Watanabe, T. Nomoto, R. Arita, Phys. Rev. B, 105, 174111 (2022). "Quantum and temperature effects on the crystal structure of superhydride LaH_{10}: A path integral molecular dynamics study"
- K. Yeo, K. Park, S. Jeong, Curr. Appl. Phys., 39, 62-69 (2022). "Neural network approach to diffusion of B and N adatoms on the Pt (111) surface"
- K. Saito, Y. Sugiura, T. Miyazaki, Y. Takahashi, T. Takayanagi, Phys. Chem. Chem. Phys., 23, 6950-6958 (2021). "Quantum calculations of the photoelectron spectra of the OH^-NH_3 anion: implications for $OH+NH_3 \rightarrow H_2O+NH_2$ reaction dynamics"
- Y. Hashimoto, K. Saito, T. Takayanagi, H. Tachikawa, Phys. Chem. Chem. Phys., 23, 16958 (2021). "Theoretical study of the dissociative photodetachment dynamics of the hydrated superoxide anion cluster"
- K. Saito, Y. Hashimoto, T. Takayanagi, J. Phys. Chem. A, 125, 10750-10756 (2021). "Ring-Polymer Molecular Dynamics Calculations of Thermal Rate Coefficients and Branching Ratios for the Interstellar $H_3^+ + CO \rightarrow H_2^+ + HCO^+/HOC^+$ Reaction and Its Deuterated Analogue."
- Y. Takahashi, Y. Hashimoto, K. Saito, T. Takayanagi, Molecules, 26, 7250 (2021). "On-the-Fly Ring-Polymer Molecular Dynamics Calculations of the Dissociative Photodetachment Process of the Oxalate Anion."
- B. Thomsen, M. Shiga, J. Chem. Phys., 155, 194107 (2021) (11 pages). "Ab Initio Study of Nuclear Quantum Effects on Sub- and Supercritical Water"
- A. Pal, S. Pal, S. Verma, M. Shiga, N. N. Nair, J. Comput. Chem., 42, 1996-2003 (2021). "Mean Force Based Temperature Accelerated Sliced Sampling: Efficient Reconstruction of High Dimensional Free Energy Landscapes"
- K. Kobayashi, Y. Nagai, M. Itakura, M. Shiga, J. Chem. Phys., 155, 034106 (2021) (9 pages). "Self-learning hybrid Monte Carlo for isothermal-isobaric ensemble: Application to liquid silica"
- T. Kondo, T. Sasaki, M. Shiga, J. Comput. Chem., 42, 1783-1791 (2021). The Mechanism of Sorbitol Dehydration in Hot Acidic Solutions"
- H. Wang, Y. Yao, F. Peng, H. Liu, R. J. Hemley, Phys. Rev. Lett., 126, 117002 (2021). "Quantum and Classical Proton Diffusion in Superconducting Clathrate Hydrides"
- B. Thomsen, M. Shiga, J. Chem. Phys., 154, 084117 (2021). "Nuclear Quantum Effects on Autoionization of Water Isotopologues Studied by Ab Initio Path Integral Molecular Dynamics"
- T. Kondo, T. Sasaki, S. Ruiz Barragan, J. Ribas, M. Shiga, J. Comput. Chem., 42, 156-165 (2021). "Refined metadynamics through canonical sampling using time-invariant bias potential: A study of polyalcohol dehydration in hot acidic solutions"
- L. Yan, Y. Yamamoto, M. Shiga, O. Sugino, Phys. Rev. B, 101, 165414 (2020). "Nuclear quantum effect for hydrogen adsorption on Pt(111)"
- Y. Nagai, M. Okumura, K. Kobayashi, M. Shiga, Phys. Rev. B (Rapid Commun.), 102, 041124 (2020). "Self-learning hybrid Monte Carlo: A first principles approach"
- T. Ishiyama, A. Morita, J. Phys. Chem. Lett. 10, 5070-5075 (2019). "Nuclear Quantum Effect on the chi(2) Band Shape of Vibrational Sum Frequency Generation Spectra of Normal and Deuterated Water Surfaces"

- Y. Kawashima, K. Ishimura, M. Shiga, J. Chem. Phys., 150, 124103 (2019). "Ab initio Quantum Mechanics/Molecular Mechanics Method with Periodic Boundaries Employing Ewald Summation Technique to Electron-Charge Interaction: Treatment of the Surface-Dipole Term"
- Y. L. Chang, T. Sasaki, J. Ribas Arino, M. Machida, M. Shiga, J. Phys. Chem. B, 23, 1662-1671 (2019). "Understanding Competition of Polyalcohol Dehydration Reactions in Hot Water"
- M. Shiga, M. E. Tuckerman, J. Phys. Chem. Lett., 9, 6207–6214 (2018). "Finding Free Energy Landmarks of Chemical Reactions"
- Y. Watabe, T. Miyazaki, E. Ozama, T. Takayanagi, Y. Suzuki, Comp. Theo. Chem., 1140, 56-62 (2018). "Theoretical interpretation of photoelectron spectrum of $(AuCO_2)^-$ anion"
- E. Ozama, S. Adachi, T. Takayanagi, M. Shiga, Chem. Eur. J., 24, 12716-12721 (2018). "Quantum simulation verifies the stability of an 18 coordinated actinium-helium complex"
- M. Machida, K. Kato, M. Shiga. J. Chem. Phys., 148, 102324 (2018). "Nuclear quantum effects of light and heavy water studied by all-electron first principles path integral simulations."
- Y. Seki, T. Takayanagi, M. Shiga, Phys. Chem. Chem. Phys., 19, 13798-13806, (2017). "Photoexcited Ag ejection from a low-temperature He cluster: A simulation study by nonadiabatic Ehrenfest ring-polymer molecular dynamics"
- S. Ruiz-Barragan, J. Ribas-Arino, M. Shiga, Phys. Chem. Chem. Phys., 18, 32438-32447 (2016). "The reaction mechanism of polyalcohol dehydration in hot pressurized water"
- Y. Minoshima, Y. Seki, T. Takayanagi, M. Shiga, Chem. Phys., 472, 1-8 (2016). "Effects of temperature and isotopic substitution on electron attachment dynamics of guanine-cytosine base pair: Ring-polymer and classical molecular dynamics simulations"
- S. Ruiz-Barragan, K. Ishimura, M. Shiga, Chem. Phys. Lett., 646, 130-135 (2016). "On the hierarchical parallelization of ab initio simulations"
- T. Honda, Y. Minoshima, Y. Yokoi, T. Takayanagi and M. Shiga, Chem. Phys. Lett., 625, 174-178 (2015). “Semiclassical dynamics of electron attachment to guanine-cytosine base pair”
- M. Shiga, M. Masia, J. Chem. Phys., 139, 144103 (2013) (14 pages). “Boundary based on Exchange Symmetry Theory for Multilevel Simulations II. Multiple Time Scale Approach”
- M. Shiga, M. Masia, J. Chem. Phys., 139, 044120 (2013) (8 pages); (E) J. Chem. Phys., 139, 119901 (2013). “Boundary based on Exchange Symmetry Theory for Multilevel Simulations I. Basic Theory”
- K. Suzuki, M. Tachikawa, M. Shiga, J. Chem. Phys., 138, 184307 (2013) (7 pages). “Temperature dependence on the structure of Zundel cation and its isotopomers”
- A. Koizumi, M. Tachikawa, M. Shiga, Chem. Phys., 419, 44-49 (2013). “Quantum fluctuation and vibrational dynamics of aqueous Cu^+ and Ag^+ clusters”
- T. Yoshikawa, T. Takayanagi, H. Kimizuka, M. Shiga, J. Phys. Chem. C, 116, 23113-23119 (2012). “Quantum-Thermal Crossover of Hydrogen and Tritium Diffusion in α-Iron”
- M. Shiga, H. Fujisaki, J. Chem. Phys., 136, 184103 (2012) (11 pages). “A quantum generalization of intrinsic reaction coordinate using path integral centroid coordinates”
- K. Suzuki, H. Ishibashi, K. Yagi, M.Shiga, M. Tachikawa, Progress in Theoretical Chemistry and Physics B 26: Quantum Systems in Chemistry and Physics: Progress in Methods and Applications edited by K. Nishikawa et al., Chapter 10, 207-216 (2012). “Ab initio path integral molecular dynamics simulations of F_2H^- and $F_2H_3^+$”
- T. Yoshikawa, S. Sugawara, T. Takayanagi, M. Shiga, M. Tachikawa, Chem. Phys., 394, 46-51 (2012). “Quantum tautomerization in porphycene and its isotopomers: Path-integral molecular dynamics simulations”
- A. Koizumi, K. Suzuki, M. Shiga, M. Tachikawa, Int. J. Quant. Chem., 112, 136-139 (2012). “Ab initio path integral simulation of $AgOH(H_2O)$”
- S. Sugawara, T. Yoshikawa, T. Takayanagi, M. Shiga, M. Tachikawa, J. Phys. Chem. A, 115, 11486–11494 (2011). “Quantum Proton Transfer in Hydrated Sulfuric Acid Clusters: A Perspective from Semiempirical Path Integral Simulations”
- M. Daido, A. Koizumi, M. Shiga, M. Tachikawa, Theoret. Chem. Acc., 130, 385-391 (2011). “Nuclear quantum effect on the structure of guanine cytosine pair”
- P. Dopieralski, P. Anjukandi, M. Rueckert, M. Shiga, J. Ribas-Arino, D. Marx, J. Mater. Chem., 21, 8309-8316 (2011). “On the role of polymer chains in transducing external mechanical forces to benzocyclobutene mechanophores”

- A. Koizumi, K. Suzuki, M. Shiga, M. Tachikawa, J. Chem. Phys., (communication), 134, 031101 (2011); (E) J. Chem. Phys. 134, 169901 (2011). “A concerted mechanism between protron transfer of Zundel anion and displacement of counter cation”
- M. Sugimoto, M. Shiga, M. Tachikawa, Comp. Theor. Chem., 975, 31-37 (2011). “Nuclear quantum effect on the dissociation energies of cationic hydrogen clusters”
- J. Ribas-Arino, M. Shiga, D. Marx, J. Am. Chem. Soc., 132, 10609-10614 (2010). “Mechanochemical Transduction of Externally Applied Forces to Mechanophores”
- T. Yoshikawa, S. Sugawara, T. Takayanagi, M. Shiga, M. Tachikawa, Chem. Phys. Lett., 496, 14-19 (2010). “Theoretical study on the mechanism of double proton transfer in porphycene by path-integral molecular dynamics simulations”
- M. Shiga, K. Suzuki, M. Tachikawa, J. Chem. Phys., 132, 114104 (2010) (7 pages). “The chemical shift of deprotonated water dimer: Ab initio path integral simulation”
- K. Suzuki, M. Tachikawa, M. Shiga, J. Chem. Phys., 132, 144108 (2010) (7 pages). “Efficient ab initio path integral hybrid Monte Carlo based on the fourth-order Trotter expansion – Application to fluoride ion-water cluster”
- S. D. Ivanov, A. Witt, M. Shiga, D. Marx, J. Chem. Phys., (communication), 132, 031101 (2010) (4 pages). “On Artificial Frequency Shifts in Infrared Spectra obtained from Centroid Molecular Dynamics: Quantum Liquid Water”
- A. Kakizaki, H. Motegi, T. Yoshikawa, T. Takayanagi, M. Shiga, M. Tachikawa, J. Molec. Struct. (THEOCHEM), 901, 1-8 (2009). “Path-integral molecular dynamics simulations of small hydrated sulfuric acid clusters $H_2SO_4(H_2O)_n$ (n = 1-6) on semiempirical PM6 potential surfaces”
- T. Takayanagi, K. Takahashi, A. Kakizaki, M. Shiga, M. Tachikawa, Chem. Phys., 358, 196-202 (2009). “Path-integral molecular dynamics simulations of hydrated hydrogen chloride cluster $HCl(H_2O)_4$ on a semiempirical potential energy surface”
- A. Kaczmarek, M. Shiga, D. Marx, J. Phys. Chem. A, 113, 1985-1994 (2009). “Quantum effects on vibrational and electronic spectra of hydrazine studied by on the fly ab initio ring polymer molecular dynamics”
- J. Ribas-Arino, M. Shiga, D. Marx, Angew. Chem. Int. Ed. (communication), 48, 4190-4193 (2009). “Understanding Covalent Mechanochemistry”
- A. Witt, S. D. Ivanov, M. Shiga, H. Forbert, D. Marx, J. Chem. Phys., 130, 194510 (2009) (15 pages). “On the applicability of centroid and ring polymer path integral molecular dynamics for vibrational spectroscopy”
- A. Nakayama, T. Taketsugu, M. Shiga, Chem. Lett., 38, 976-977 (2009). “Speed-up of ab initio hybrid Monte Carlo and ab initio path integral hybrid Monte Carlo simulations using an auxiliary potential energy surface”
- J. Ribas-Arino, M. Shiga, D. Marx, Chem. Eur. J. (communication), 15, 13331-13335 (2009). “Unravelling the mechanism of force-induced ring-opening of benzocyclobutenes”
- T. Takayanagi, T. Yoshikawa, H. Motegi, M. Shiga, Chem. Phys. Lett., 482, 195-200 (2009). “Path-integral molecular dynamics simulations for water anion clusters $(H_2O)_5^-$ and $(D_2O)_5^-$”
- T. Yoshikawa, H. Motegi, A. Kakizaki, T. Takayanagi, M. Shiga, M. Tachikawa, Chem. Phys., 365, 60-68 (2009). “Path-integral molecular dynamics simulations of glycine-$(H_2O)_n$ (n = 1-7) clusters on semiempirical PM6 potential energy surfaces”
- T. Takayanagi, T. Yoshikawa, A. Kakizaki, M. Shiga, M. Tachikawa, J. Molec. Struct. (THEOCHEM), 869, 29-36 (2008). “Molecular dynamics simulations of small glycine-$(H_2O)_n$ (n = 2-7) clusters on semiempirical PM6 potential energy surfaces”
- H. Ishibashi, A. Hayashi, M. Shiga, M. Tachikawa, ChemPhysChem (communication), 9, 383-387 (2008). “Geometric isotope effect on $N_2H_7^+$ cation and $N_2H_5^-$ anion by ab initio path integral molecular dynamics simulation”
- M. Shiga, A. Nakayama, Chem. Phys. Lett., 451, 175-181 (2008). “ab initio path integral ring polymer molecular dynamics: vibrational spectra of molecules”
- H. Motegi, A. Kakizaki, T. Takayanagi, Y. Taketsugu, T. Takatsugu, M. Shiga, Chem. Phys., 454, 1-6 (2008). “Path-integral molecular dynamics simulations of BeO embedded in helium clusters: Formation of the stable HeBeO complex”
- K. Suzuki, M. Shiga, M. Tachikawa, J. Chem. Phys., 129, 144310 (8 pages) (2008); (E) J. Chem. Phys., 131, 039903 (2009). “Temperature and isotope effects on water cluster ions with path integral molecular dynamics based on 4th order Trotter expansion”

- A. Hayashi, M. Shiga, M. Tachikawa, Mol. Simul., 33, 185-188 (2007). “H/D isotope effect on the dihydrogen bond by ab initio path integral molecular dynamics simulation”
- M. Shiga, M. Tachikawa, Mol. Simul., 33, 171-184 (2007). “Ab initio quantum mechanical/molecular mechanical molecular dynamics using multiple-time-scale approach and perturbation theory”
- T. Kobayashi, M. Shiga, A. Murakami, S. Nakamura, J. Am. Chem. Soc., 129, 6405-6424 (2007). “Ab initio Study of Ultrafast Photochemical Reaction Dynamics of Phenol Blue”
- A. Kakizaki, T. Takayanagi, M. Shiga, Chem. Phys. Lett., 449, 28-32 (2007). “Path integral molecular dynamics calculations of the H_6^+ and D_6^+ clusters on an ab initio potential energy surface”
- A. Hayashi, M. Shiga, M. Tachikawa, J. Chem. Phys., 125, 204310 (7 pages) (2006). “H/D isotope effect on the dihydrogen bond of $NH_4^+BeH_2$ by ab initio path integral molecular dynamics simulation”
- M. Shiga, W. Shinoda, J. Chem. Phys., 123, 134502 (8 pages) (2005). “Calculation of heat capacities of light and heavy water by path integral molecular dynamics”
- M. Tachikawa, M. Shiga, J. Am. Chem. Soc., (communication) 127, 11906-11909 (2005). “Geometrical Isotope Effect on Hydrogen Bond in Charged Water Clusters”
- M. Tachikawa, M. Shiga, J. Theor. Comput. Chem., 4, 175-181 (2005). “Ab initio path integral study on isotope effect of ammonia molecule”
- T. Takayanagi, M. Shiga, T. Taketsugu, J. Theor. Comput. Chem., 4, 197-207 (2005). “Development of a three-dimensional ab initio potential energy surface for the He-Cl_2 (X) system and its application to solvation structures in the He_nCl_2 cluster”
- W. Shinoda, M. Shiga, Phys. Rev. E, 71, 041204 (4 pages) (2005). “Quantum simulation of the heat capacity of water”
- M. Tachikawa, M. Shiga, Chem. Phys. Lett., 407 135-138 (2005). “Ab initio path integral simulation study on $^{16}O/^{18}O$ isotope effect in water and hydronium ion”
- A. Hayashi, M. Shiga, M. Tachikawa, Chem. Phys. Lett., 407 135-138 (2005). “Ab initio path integral simulation study on the dihydrogen bond of $NH_4^+BH_2$”
- M. Tachikawa, M. Shiga, J. Chem. Phys. 121, 5985-5991 (2004). “Theoretical study on isotope and temperature effect in hydronium ion using ab initio path integral simulation”
- T. Takayanagi, M. Shiga, Phys. Chem. Chem. Phys., 6, 3241-3247 (2004). “Theoretical study on photoabsorption dynamics of the K atom attached to helium clusters and the solvation structures of K^*He_n exciplexes”
- M. Shiga, M. Tachikawa, Chem. Phys. Lett., 374, 229-234 (2003). “Ab initio path integral study of hydronium ion”
- A. Wada, T. Takayanagi, M. Shiga, J. Chem. Phys., 119, 5478-5486 (2003). “Theoretical simulations on photoexcitation dyamics of the silver atoms embedded in helium clusters”
- T. Takayanagi, M. Shiga, Chem. Phys. Lett., 372, 90-96 (2003). “Photodissociation of Cl_2 in helium clusters: an application of hybrid method of quantum wavepacket dynamics and path integral centroid molecular dynamics”
- M. Shiga, T. Takayanagi, Chem. Phys. Lett., 378, 539-547 (2003). “Quantum path-integral molecular dynamics calculations of the dipole-bound state of the water dimer anion”
- M. Shiga, M. Yamaguchi, H. Kaburaki, Phys. Rev. B, 68, 245402 (8 pages) (2003). “Structure and energetics of clean and hydrogenated Ni surfaces and symmetric tilt grain boundaries using embedded-atom method”
- T. Takayanagi, M. Shiga, Chem. Phys. Lett., 362, 504-510 (2002). “Path integral molecular dynamics combined with discrete-variable-representation approach: the effect of solvation structures on vibrational spectra of Cl_2 in helium clusters”
- M. Shiga, M. Tachikawa, S. Miura, J. Chem. Phys., 115, 9149-9159 (2001). “A unified scheme for ab initio molecular orbital theory and path integral molecular dynamics”
- M. Shiga, M. Tachikawa, S. Miura, Chem. Phys. Lett., 332, 396-402 (2000). “Ab initio molecular orbital calculation considering the quantum mechanical effect of nuclei by path integral molecular dynamics”

References

[1] PIMD code, version 2.0, M. Shiga, 2016; PIMD code, version 2.1, M. Shiga, 2017; PIMD code, version 2.2, M. Shiga, 2018; PIMD code, version 2.3, M. Shiga, 2019; PIMD code, version 2.4, M. Shiga, 2020; PIMD code, version 2.5, M. Shiga, 2022; PIMD code, version 2.6, M. Shiga, 2023; PIMD code, version 2.7, M. Shiga, 2025; PIMD code, version 2.8.0, M. Shiga, 2026.

[2] C. G. Broyden, IMA Journal of Applied Mathematics 6, 76–90 (1970), doi:10.1093/imamat/6.1.76; R. Fletcher, The Computer Journal 13, 317–322 (1970), doi:10.1093/comjnl/13.3.317; D. Goldfarb, Mathematics of Computation 24, 23–26 (1970), doi:10.1090/S0025-5718-1970-0258249-6; D. F. Shanno, Mathematics of Computation 24, 647–656 (1970), doi:10.1090/S0025-5718-1970-0274029-X.

[3] Limited-memory BFGS: J. Nocedal, "Updating quasi-Newton matrices with limited storage", Mathematics of Computation 35, 773–782 (1980), doi:10.1090/S0025-5718-1980-0572855-7; D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization", Mathematical Programming 45, 503–528 (1989), doi:10.1007/BF01589116. http://www.ece.northwestern.edu/~nocedal/lbfgs.html.

[4] W. Humphrey, A. Dalke and K. Schulten, "VMD—Visual Molecular Dynamics", J. Mol. Graphics 14, 33–38 (1996), doi:10.1016/0263-7855(96)00018-5; available from website. http://www.ks.uiuc.edu/Research/vmd/.

[5] K. Fukui, Acc. Chem. Res. 14, 363–368 (1981), doi:10.1021/ar00072a001.

[6] M. P. Allen and D. J. Tildesley, Computer Simulation of Liquids, Oxford University Press, Oxford, 1987.

[7] D. Frenkel and B. Smit, Understanding Molecular Simulation: From Algorithms to Applications, 2nd ed., Academic Press, San Diego, 2002, doi:10.1016/B978-0-12-267351-1.X5000-7.

[8] S. Nosé, J. Chem. Phys. 81, 511–519 (1984), doi:10.1063/1.447334; Mol. Phys. 52, 255–268 (1984), doi:10.1080/00268978400101201.

[9] S. Nosé and M. L. Klein, Mol. Phys. 50, 1055–1076 (1983), doi:10.1080/00268978300102851.

[10] W. G. Hoover, Phys. Rev. A 31, 1695–1697 (1985), doi:10.1103/PhysRevA.31.1695.

[11] G. J. Martyna, M. L. Klein, and M. E. Tuckerman, J. Chem. Phys. 97, 2635–2643 (1992), doi:10.1063/1.463940.

[12] M. E. Tuckerman, B. J. Berne, and G. J. Martyna, J. Chem. Phys. 97, 1990–2001 (1992), doi:10.1063/1.463137.

[13] G. J. Martyna, M. E. Tuckerman, D. J. Tobias, and M. L. Klein, Mol. Phys. 87, 1117–1157 (1996), doi:10.1080/00268979600100761.

[14] G. J. Martyna, A. Hughes, and M. E. Tuckerman, J. Chem. Phys. 110, 3275–3290 (1999), doi:10.1063/1.478193.

[15] W. Shinoda, M. Shiga, and M. Mikami, Phys. Rev. B 69, 134103 (2004), doi:10.1103/PhysRevB.69.134103.

[16] H. C. Andersen, J. Chem. Phys. 72, 2384–2393 (1980), doi:10.1063/1.439486.

[17] M. Parrinello and A. Rahman, Phys. Rev. Lett. 45, 1196–1199 (1980), doi:10.1103/PhysRevLett.45.1196; J. Appl. Phys. 52, 7182–7190 (1981), doi:10.1063/1.328693; J. Chem. Phys. 76, 2662–2666 (1982), doi:10.1063/1.443248.

[18] P. Pulay and G. Fogarasi, Chem. Phys. Lett. 386, 272–278 (2004), doi:10.1016/j.cplett.2004.01.069.

[19] J. Kolafa, J. Comput. Chem. 25, 335–342 (2004), doi:10.1002/jcc.10385.

[20] J. C. Tully, J. Chem. Phys. 93, 1061–1071 (1990), doi:10.1063/1.459170.

[21] D. Marx and J. Hutter, Ab Initio Molecular Dynamics, Basic Theory and Advanced Methods, Cambridge, 2009.

[22] M. Tuckerman, Statistical Mechanics: Theory and Molecular Simulation, Oxford, 2010.

[23] M. E. Tuckerman, B. J. Berne, G. J. Martyna and M. L. Klein, J. Chem. Phys. 99, 2796–2808 (1993), doi:10.1063/1.465188.

[24] I. R. Craig and D. E. Manolopoulos, J. Chem. Phys. 121, 3368–3373 (2004), doi:10.1063/1.1777575; J. Chem. Phys. 122, 084106 (2005), doi:10.1063/1.1850093; J. Chem. Phys. 123, 034102 (2005), doi:10.1063/1.1954769.

[25] B. J. Braams and D. E. Manolopoulos, J. Chem. Phys. 125, 124105 (2006), doi:10.1063/1.2357599.

[26] J. Cao and G. A. Voth, J. Chem. Phys. 99, 10070–10073 (1993), doi:10.1063/1.465512; J. Chem. Phys. 100, 5093–5105 (1994), doi:10.1063/1.467175; J. Chem. Phys. 100, 5106–5117 (1994), doi:10.1063/1.467176; J. Chem. Phys. 101, 6157–6167 (1994), doi:10.1063/1.468503; J. Chem. Phys. 101, 6168–6183 (1994), doi:10.1063/1.468399; J. Chem. Phys. 101, 6184–6192 (1994), doi:10.1063/1.468400.

[27] S. Jang and G. A. Voth, J. Chem. Phys. 111, 2357–2370 (1999), doi:10.1063/1.479514; J. Chem. Phys. 111, 2371–2384 (1999), doi:10.1063/1.479515.

[28] M. Shiga, M. Tachikawa and S. Miura, Chem. Phys. Lett. 332, 396–402 (2000), doi:10.1016/S0009-2614(00)01269-0; J. Chem. Phys. 115, 9149–9159 (2001), doi:10.1063/1.1407289.

[29] M. Shiga and A. Nakayama, Chem. Phys. Lett. 451, 175–181 (2008), doi:10.1016/j.cplett.2007.11.091.

[30] A. Kaczmarek, M. Shiga and D. Marx, J. Phys. Chem. A 113, 1985–1994 (2009), doi:10.1021/jp8081936.

[31] A. Witt, S. D. Ivanov, M. Shiga, H. Forbert, and D. Marx, J. Chem. Phys. 130, 194510 (2009), doi:10.1063/1.3125009.

[32] M. Shiga and H. Fujisaki, J. Chem. Phys. 136, 184103 (2012), doi:10.1063/1.4709723.

[33] M. Shiga, J. Comput. Chem. 43, 1864–1879 (2022), doi:10.1002/jcc.26989.

[34] M. Rossi, M. Ceriotti, and D. E. Manolopoulos, J. Chem. Phys. 140, 234116 (2014), doi:10.1063/1.4883861.

[35] K. Suzuki, M. Tachikawa, and M. Shiga, J. Chem. Phys. 132, 144108 (2010), doi:10.1063/1.3367724.

[36] A. Laio and M. Parrinello, Proc. Natl. Acad. Sci. USA 99, 12562–12566 (2002), doi:10.1073/pnas.202427399.

[37] P. Raiteri, A. Laio, F. L. Gervasio, C. Micheletti, and M. Parrinello, J. Phys. Chem. B 110, 3533–3539 (2006), doi:10.1021/jp054359r.

[38] B. Ensing, A. Laio, M. Parrinello, and M. L. Klein, J. Phys. Chem. B 109, 6676–6687 (2005), doi:10.1021/jp045571i.

[39] W. E, W. Ren, and E. Vanden-Eijnden, Phys. Rev. B 66, 052301 (2002), doi:10.1103/PhysRevB.66.052301; J. Chem. Phys. 126, 164103 (2007), doi:10.1063/1.2720838.

[40] J. Ribas-Ariño, M. Shiga and D. Marx, Angew. Chem. Int. Ed. 48, 4190–4193 (2009), doi:10.1002/anie.200900673; J. Am. Chem. Soc. 132, 10609–10614 (2010), doi:10.1021/ja104958e.

[41] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey and M. L. Klein, J. Chem. Phys. 79, 926–935 (1983), doi:10.1063/1.445869.

[42] E. Neria, S. Fischer and M. Karplus, J. Chem. Phys. 105, 1902–1921 (1996), doi:10.1063/1.472061.

[43] S. Habershon, T. E. Markland and D. E. Manolopoulos, J. Chem. Phys. 131, 024501 (2009), doi:10.1063/1.3167790.

[44] J. Lobaugh and G. A. Voth, J. Chem. Phys. 106, 2400–2410 (1997), doi:10.1063/1.473151.

[45] J. R. Reimers, R. O. Watts and M. L. Klein, Chem. Phys. 64, 95–114 (1982), doi:10.1016/0301-0104(82)85006-4.

[46] M. S. Daw and M. I. Baskes, Phys. Rev. Lett. 50, 1285–1288 (1983), doi:10.1103/PhysRevLett.50.1285; Phys. Rev. B 29, 6443–6453 (1984), doi:10.1103/PhysRevB.29.6443.

[47] S. M. Foiles, M. I. Baskes and M. S. Daw, Phys. Rev. B 33, 7983–7991 (1986), doi:10.1103/PhysRevB.33.7983.

[48] J. E. Angelo, N. R. Moody, and M. I. Baskes, Modelling Simul. Mater. Sci. Eng. 3, 289–307 (1995), doi:10.1088/0965-0393/3/3/001; M. I. Baskes, X. W. Sha, J. E. Angelo, and N. R. Moody, Modelling Simul. Mater. Sci. Eng. 5, 651–652 (1997), doi:10.1088/0965-0393/5/6/007.

[49] J. Tersoff, Phys. Rev. Lett. 61, 2879–2882 (1988), doi:10.1103/PhysRevLett.61.2879; J. Tersoff, Phys. Rev. B 37, 6991–7000 (1988), doi:10.1103/PhysRevB.37.6991; J. Tersoff, Phys. Rev. B 39, 5566–5568 (1989), doi:10.1103/PhysRevB.39.5566; J. Tersoff, Phys. Rev. B 41, 3248 (1990), doi:10.1103/PhysRevB.41.3248.2.

[50] P. Clabaut, M. Beisert, C. Michel, S. N. Steinmann, J. Chem. Phys. 157, 194705 (2022), doi:10.1063/5.0130368.

[51] Y. Mishin, M. J. Mehl, D. A. Papaconstantopoulos, Acta Mater. 53, 4029–4041 (2005), doi:10.1016/j.actamat.2005.05.001.

[52] R. T. Cygan, J.-J. Liang, A. G. Kalinichev, J. Phys. Chem. B 108, 1255–1266 (2004), doi:10.1021/jp0363287.

[53] M. Shiga and M. Masia, J. Chem. Phys. 139, 044120 (2013), doi:10.1063/1.4816629; J. Chem. Phys. 139, 119901 (2013), doi:10.1063/1.4821990; J. Chem. Phys. 139, 144103 (2013), doi:10.1063/1.4823729.

[54] J. J. P. Stewart, “MOPAC: A General Molecular Orbital Package.” Quant. Chem. Prog. Exch., 10, 86 (1990); MOPAC2009, MOPAC2012, James J. P. Stewart, Stewart Computational Chemistry, Colorado Springs, CO, USA.

Available from website http://openmopac.net.

[55] Density Functional based Tight Binding (DFTB): D. Porezag, T. Frauenheim, T. Köhler, G. Seifert, and R. Kaschner, “Construction of tight-binding-like potentials on the basis of density-functional theory: Application to carbon”, Phys. Rev. B 51, 12947–12957 (1995), doi:10.1103/PhysRevB.51.12947; G. Seifert, D. Porezag, and T. Frauenheim, “Calculations of molecules, clusters, and solids with a simplified LCAO-DFT-LDA scheme”, Int. J. Quantum Chem. 58, 185–192 (1996), doi:10.1002/(SICI)1097-461X(1996)58:2<185::AID-QUA7>3.0.CO;2-U; M. Elstner, D. Porezag, G. Jungnickel, J. Elsner, M. Haugk, T. Frauenheim, S. Suhai, and G. Seifert, “Self-consistent-charge density-functional tight-binding method for simulations of complex materials properties”, Phys. Rev. B 58, 7260–7268 (1998), doi:10.1103/PhysRevB.58.7260.

Available from website http://www.dftb.org/home.

[56] The CP2K developers group, 2020. CP2K is freely available from website https://www.cp2k.org.

Please see the end of your PIMD output for the list of citations recommended for your particular type of CP2K calculation.

J. Hutter, M. Iannuzzi, F. Schiffmann, J. VandeVondele, “CP2K: Atomistic simulations of condensed matter systems”, Wiley Interdisciplinary Reviews: Computational Molecular Science, 4, 15–25 (2014), doi:10.1002/wcms.1159.

T. D. Kühne, M. Iannuzzi, M. Del Ben, V. V. Rybkin, P. Seewald, F. Stein, T. Laino, R. Z. Khaliullin, O. Schütt, F. Schiffmann, D. Golze, J. Wilhelm, S. Chulkov, M. H. Bani-Hashemian, V. Weber, U. Borštnik, M. Taillefumier, A. S. Jakobovits, A. Lazzaro, H. Pabst, T. Müller, R. Schade, M. Guidon, S. Andermatt, N. Holmberg, G. K. Schenter, A. Hehn, A. Bussy, F. Belleflamme, G. Tabacchi, A. Glöß, M. Lass, I. Bethune, C. J. Mundy, C. Plessl, M. Watkins, J. VandeVondele, M. Krack, and J. Hutter, “CP2K: An electronic structure and molecular dynamics software package-Quickstep: Efficient and accurate electronic structure calculations”, J. Chem. Phys. 152, 194103 (2020), doi:10.1063/5.0007045.

[57] OpenMX, version 4.0 with patch 4.0.1, OpenMX website, `https://www.openmx-square.org/` (2026).

T. Ozaki, “Variationally optimized atomic orbitals for large-scale electronic structures”, Phys. Rev. B 67, 155108 (2003), doi:10.1103/PhysRevB.67.155108.

[58] T. Yamasaki, A. Kuroda, T. Kato, J. Nara, J. Koga, T. Uda, K. Minami, and T. Ohno, “Multi-axis Decomposition of Density Functional Program for Strong Scaling up to 82,944 Nodes on the K Computer: Compactly Folded 3D-FFT Communicators in the 6D Torus Network”, Comput. Phys. Commun. 244, 264–269 (2019), doi:10.1016/j.cpc.2019.04.008.

T. Hamada and T. Ohno, “First-Principles Broadband Dielectric Spectroscopy”, J. Aust. Ceram. Soc., 47, 61 (2011).

PHASE/0 is available from website https://azuma.nims.go.jp.

[59] GAUSSIAN 98, Revision A.6, M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, V. G. Zakrzewski, J. A. Montgomery, Jr., R. E. Stratmann, J. C. Burant, S. Dapprich, J. M. Millam, A. D. Daniels, K. N. Kudin, M. C. Strain, Ö. Farkas, J. Tomasi, V. Barone, M. Cossi, R. Cammi, B. Mennucci, C. Pomelli, C. Adamo, S. Clifford, J. Ochterski, G. A. Petersson, P. Y. Ayala, Q. Cui, K. Morokuma, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. Cioslowski, J. V. Ortiz, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. Gomperts, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, C. Gonzalez, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, J. L. Andres, C. Gonzalez, M. Head-Gordon, E. S. Replogle, and J. A. Pople, GAUSSIAN, Inc., Pittsburgh PA, 1998.

available from website http://www.gaussian.com/index.htm.

[60] GAUSSIAN 03, Revision C.02, M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, Ö. Farkas, D. K. Malick, A. D. Rabuck, K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, and J. A. Pople, GAUSSIAN, Inc., Wallingford CT, 2004.

Available from website http://www.gaussian.com/index.htm.

[61] GAUSSIAN 09, Revision D.01, M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Ö. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, and D. J. Fox, Gaussian, Inc., Wallingford CT, 2009.

Available from website http://www.gaussian.com/index.htm.

[62] GAUSSIAN 16, Revision B.01, M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox, Gaussian, Inc., Wallingford CT, 2016.

Available from website http://www.gaussian.com/index.htm.

[63] GAMESS: General atomic and molecular electronic structure system, M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. J. Su, T. L. Windus, M. Dupuis, J. A. Montgomery, J. Comput. Chem. **14**, 1347–1363 (1993), doi:10.1002/jcc.540141112; "Advances in electronic structure theory: GAMESS a decade later", M. S. Gordon, M. W. Schmidt pp. 1167-1189, in "Theory and Applications of Computational Chemistry: the first forty years", C. E. Dykstra, G. Frenking, K. S. Kim, G. E. Scuseria (editors), Elsevier, Amsterdam, 2005.

Available from website http://www.msg.chem.iastate.edu/gamess.

[64] TURBOMOLE V6.2 2010, a development of the University of Karlsruhe and Forschungszentrum Karlsruhe GmbH, 1989–2007, TURBOMOLE GmbH, since 2007; R. Ahlrichs, M. Bär, M. Häser, H. Horn, C. Kölmel, "Electronic structure calculations on workstation computers: The program system TURBOMOLE", Chem. Phys. Lett. 162, 165–169 (1989), doi:10.1016/0009-2614(89)85118-8.

Available from website http://www.turbomole.com.

[65] MOLPRO quantum chemistry package: H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, M. Schütz, "Molpro: a general purpose quantum chemistry program package", WIREs Comput. Mol. Sci. 2, 242–253 (2012), doi:10.1002/wcms.82; MOLPRO, version 2012.1, H.-J. Werner, P. J. Knowles, G. Knizia, F. R. Manby, M. Schütz, P. Celani, T. Korona, R. Lindh, A. Mitrushenkov, G. Rauhut, K. R. Shamasundar, T. B. Adler, R. D. Amos, A. Bernhardsson, A. Berning, D. L. Cooper, M. J. O. Deegan, A. J. Dobbyn, F. Eckert, E. Goll, C. Hampel, A. Hesselmann, G. Hetzer, T. Hrenar, G. Jansen, C. Köppl, Y. Liu, A. W. Lloyd, R. A. Mata, A. J. May, S. J. McNicholas, W. Meyer, M. E. Mura, A. Nicklass, D. P. O'Neill, P. Palmieri, D. Peng, K. Pflüger, R. Pitzer, M. Reiher, T. Shiozaki, H. Stoll, A. J. Stone, R. Tarroni, T. Thorsteinsson, M. Wang.

Available from website http://www.molpro.net.

[66] NTChem: T. Nakajima, M. Katouda, M. Kamiya, and Y. Nakatsuka, Int. J. Quantum Chem. 115, 349–359 (2015), doi:10.1002/qua.24860.

Available from website http://labs.aics.riken.jp/nakajimat_top/ntchem_j.html.

[67] ORCA - An ab initio, DFT and semiempirical SCF-MO package - version 3.0. For the general ORCA program, see F. Neese, WIREs Comput. Mol. Sci. 2, 73–78 (2012), doi:10.1002/wcms.81. The version-3.0 distribution includes contributions by F. Neese, F. Wennmohs, U. Becker, D. Bykov, D. Ganyushin, A. Hansen, R. Izsák, D. G. Liakos, C. Kollmar, S. Kossmann, D. A. Pantazis, T. Petrenko, C. Reimann, C. Riplinger, M. Roemelt, B. Sandhoefer, I. Schapiro, K. Sivalingam, B. Wezisla, M. Kállay, S. Grimme, E. Valeev and G. Chan.

Available from website https://orcaforum.cec.mpg.de.

[68] Quantum Espresso: P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. Fabris, G. Fratesi, S. de Gironcoli, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, R. M. Wentzcovitch, J. Phys.: Condens. Matter 21, 395502 (2009), doi:10.1088/0953-8984/21/39/395502. https://iopscience.iop.org/article/10.1088/0953-8984/21/39/395502.

Available from website https://www.quantum-espresso.org.

[69] The Vienna Ab initio Simulation Package (VASP): G. Kresse and J. Hafner, "Ab initio molecular dynamics for liquid metals", Phys. Rev. B 47, 558–561 (1993), doi:10.1103/PhysRevB.47.558; G. Kresse and J. Hafner, "Ab initio molecular-dynamics simulation of the liquid-metal–amorphous-semiconductor transition in germanium", Phys. Rev. B 49, 14251–14269 (1994), doi:10.1103/PhysRevB.49.14251; G. Kresse and J. Furthmüller, "Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set", Comput. Mater. Sci. 6, 15–50 (1996), doi:10.1016/0927-0256(96)00008-0; G. Kresse and J. Furthmüller, "Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set", Phys. Rev. B 54, 11169–11186 (1996), doi:10.1103/PhysRevB.54.11169.

Available from website http://www.vasp.at.

[70] Scalable Molecular Analysis Solver for High-performance computing systems (SMASH), K. Ishimura.

Available from website http://smash-qc.sourceforge.net.

[71] ABINIT-MP.

Available from website http://moldb.nihs.go.jp/abinitmp.

[72] J. W. Ponder, TINKER: Software tools for molecular design, Washington University School of Medicine, Saint Louis, MO.

Available from website http://dasher.wustl.edu/tinker.

[73] W. L. Jorgensen, “BOSS - Biochemical and Organic Simulation System”, The Encyclopedia of Computational Chemistry, P. v. R. Schleyer (editor-in-chief), John Wiley & Sons Ltd, Athens, USA, 5, 3281-3285 (1998); W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives, J. Am. Chem. Soc. 118, 11225–11236 (1996), doi:10.1021/ja9621760; W. Damm, A. Frontera, J. Tirado-Rives, and W. L. Jorgensen, J. Comput. Chem. 18, 1955-1970 (1997); W. L. Jorgensen and N. A. McDonald, Theochem 424, 145-155 (1998); W. L. Jorgensen and N. A. McDonald, J. Phys. Chem. B 102, 8049-8059 (1998); R. C. Rizzo and W. L. Jorgensen, J. Am. Chem. Soc. 121, 4827-4836 (1999); E. K. Watkins and W. L. Jorgensen, J. Phys. Chem. A 105, 4118-4125 (2001); M. L. P. Price, D. Ostrovsky, and W. L. Jorgensen, J. Comput. Chem., 22, 1340-1352 (2001); W. L. Jorgensen, J. P. Ulmschneider, J. Tirado-Rives, J. Phys. Chem. B 108, 16264-70 (2004); W. L. Jorgensen, J. Tirado-Rives, J. Comput. Chem. 26, 1689-1700 (2005); K. P. Jensen and W. L. Jorgensen, J. Chem. Theory Comput. 2, 1499-1509 (2006).

[74] D. A. Pearlman, D. A. Case, J. W. Caldwell, W. S. Ross, T. E. Cheatham III, S. DeBolt, D. Ferguson, G. Seibel and P. Kollman, AMBER, a Package of Computer Programs for Applying Molecular Mechanics, Normal Mode Analysis, Molecular Dynamics and Free Energy Calculations to Simulate the Structural and Energetic Properties of Molecules, Comput. Phys. Commun. 91, 1–41 (1995), doi:10.1016/0010-4655(95)00041-D; J. Wang, P. Cieplak and P. A. Kollman, J. Comput. Chem., 21, 1049-1074 (2000) W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, Jr., D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell and P. A. Kollman, J. Am. Chem. Soc., 117, 5179-5197 (1995); G. Moyna, H. J. Williams, R. J. Nachman and A. I. Scott, Biopolymers, 49, 403-413 (1999); W. S. Ross and C. C. Hardin, J. Am. Chem. Soc., 116, 6070-6080 (1994); J. Aqvist, J. Phys. Chem., 94, 8021-8024 (1990).

[75] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan and M. Karplus, CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, J. Comput. Chem. 4, 187–217 (1983), doi:10.1002/jcc.540040211. W. E. Reiher III, “Theoretical Studies of Hydrogen Bonding”, Ph.D. Thesis, Department of Chemistry, Harvard University, Cambridge, MA, 1985. L. Nilsson and M. Karplus, “Empirical Energy Functions for Energy Minimizations and Dynamics of Nucleic Acids”, J. Comput. Chem., 7, 591-616 (1986). E. Neria, S. Fischer and M. Karplus, “Simulation of Activation Free Energies in Molecular Systems”, J. Chem. Phys., 105, 1902-1921 (1996). A. D. MacKerell, Jr., et al., “All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins”, J. Phys. Chem. B, 102, 3586-3616 (1998); N. Foloppe and A. D. MacKerell, Jr., “All-Atom Empirical Force Field for Nucleic Acids: I. Parameter Optimization Based on Small Molecule and Condensed Phase Macromolecular Target Data”, J. Comput. Chem., 21, 86-104 (2000); Current parameter values are available from the CHARMM website, http://mackerell.umaryland.edu/CHARMM_ff_params.html.

[76] T. Morishita, S. G. Itoh, H. Okumura, M. Mikami, Phys. Rev. E 85, 066702 (2012), doi:10.1103/PhysRevE.85.066702; J. Comput. Chem. 34, 1375–1384 (2013), doi:10.1002/jcc.23267.

[77] T. Morishita, Y. Yonezawa, A. M. Ito, J. Chem. Theory Comput. 13, 3106–3119 (2017), doi:10.1021/acs.jctc.7b00252.

[78] L. Rosso, P. Minárý, Z. Zhu, M. E. Tuckerman, J. Chem. Phys. 116, 4389–4402 (2002), doi:10.1063/1.1448491.

[79] J. B. Abrams and M. E. Tuckerman, J. Phys. Chem. B 112, 15742–15757 (2008), doi:10.1021/jp805039u.

[80] L. Maragliano and E. Vanden-Eijnden, Chem. Phys. Lett. 426, 168–175 (2006), doi:10.1016/j.cplett.2006.05.062.

[81] W. E and X. Zhou, Nonlinearity 24, 1831–1842 (2011), doi:10.1088/0951-7715/24/6/008; A. Samanta and W. E, J. Chem. Phys. 136, 124104 (2012), doi:10.1063/1.3692803.

[82] M. Chen, T.-Q. Yu and M. E. Tuckerman, Proc. Natl. Acad. Sci. USA 112, 3235–3240 (2015), doi:10.1073/pnas.1418241112.

[83] S. Awasthi, V. Kapil and N. N. Nair, J. Comput. Chem. 37, 1413–1424 (2016), doi:10.1002/jcc.24349.

[84] S. Awasthi and N. N. Nair, J. Chem. Phys. 146, 094108 (2017), doi:10.1063/1.4977704.

[85] FFTPACK: P. N. Swarztrauber, Vectorizing the FFT's, in Parallel Computations, edited by G. Rodrigue, Academic Press, 1982, pages 51-83.

[86] Y. Nagai, "AccelNet: Exact backward-compatible acceleration of polynomial angular descriptors through Cartesian moment factorization", arXiv:2608.03280 [cond-mat.mtrl-sci] (2026), preprint. `https://doi.org/10.48550/arXiv.2608.03280`. Source code: `https://github.com/cometscome/AccelNet`.

[87] N. Artrith, The Atomic Energy Neural Network (AENET) version 2.0.0: N. Artrith and A. Urban, Comput. Mater. Sci. 114, 135–150 (2016), doi:10.1016/j.commatsci.2015.11.047; N. Artrith, A. Urban, and G. Ceder, Phys. Rev. B 96, 014112 (2017), doi:10.1103/PhysRevB.96.014112; available from AENET website. `http://ann.atomistic.net/`.

[88] J. Sala, E. Guárdia, M. Masia, J. Chem. Phys. 133, 234101 (2010), doi:10.1063/1.3511713.

[89] L. Ojamäe, I. Shavitt and S. J. Singer, J. Chem. Phys. 109, 5547–5564 (1998), doi:10.1063/1.477173.

[90] I. S. Novikov, K. Gubaev, E. V. Podryabinkin, and A. V. Shapeev, Machine Learning: Science and Technology 2, 025002 (2021), doi:10.1088/2632-2153/abc9fe.

[91] A. V. Shapeev, Multiscale Modeling & Simulation 14, 1153–1173 (2016), doi:10.1137/15M1054183.

[92] I. Batatia, D. P. Kovács, G. Simm, C. Ortner and G. Csányi, "MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields", arXiv:2206.07697 (2022).

[93] M. Riera, C. Knight, E. F. Bull-Vulpe, X. Zhu, H. Agnew, D. G. A. Smith, A. C. Simmonett, and F. Paesani, "MBX: A many-body energy and force calculator for data-driven many-body simulations", J. Chem. Phys. 159, 054802 (2023), doi:10.1063/5.0156036.

[94] S. Gupta, E. F. Bull-Vulpe, H. Agnew, S. Iyer, X. Zhu, R. Zhou, C. Knight, and F. Paesani, "MBX v1.2: Accelerating data-driven many-body molecular dynamics simulations", J. Chem. Theory Comput. 21, 1838 (2025), doi:10.1021/acs.jctc.4c01333.

[95] n2p2 - A neural network potential package: A. Singraber, J. Behler, and C. Dellago, J. Chem. Theory Comput. 15, 1827–1840 (2019), doi:10.1021/acs.jctc.8b00770. The software DOI is doi:10.5281/zenodo.1344446. Available from website https://github.com/CompPhysVienna/n2p2.

[96] PFP - PreFerred Potential. Website https://matlantis.com/en/product/about-pfp/.