Eigen ライブラリ
インストール手順書
及び
利用マニュアル

（Ver1.0）

目次

１．概要

Eigen ライブラリは、実対称行列およびエルミート行列の固有値と固有ベクトルを求めるライブラリです。通信空間を 2 次元に分割しており、高並列でもスケールするように工夫されています。

実対称行列の固有値固有ベクトルを求める場合は、3 重対角化あるいは 5 重対角化を選択することができます。

２．必要ソフト

Eigen ライブラリでは、BLAS および Scalapack を使用しています。よって、リンク時（ロードモジュール作成時）に、BLAS および Scalapack をリンクするためのオプションが必要になります。これらのオプションは、インストール時の環境変数で指定します。詳細は「3-1.インストール方法」を参照してください。

また、Eigen ライブラリは、MPI によるプロセス並列化と OpenMP によるスレッド並列化の両方が実施されています。よって、MPI ライブラリが必要です。さらに、OpenMP に対応したコンパイラも必要になります。

必要ライブラリ

BLAS
ScaLapack
MPI

３．インストール

3.1. インストール方法

各ルーチンの Makefile のコンパイルオプション等を利用する計算機の環境に合わせて修正し、make してください。

3.2. 時間計測機能

Eigen ライブラリには、実行時間を計測する機能があります。この機能を有効にするためには、コンパイルオプションに「-Dtimer」を指定する必要があります。コンパイルオプションの指定方法については、「3-1.インストール方法」内の「コンパイルオプションの指定」を参照してください。ただし、当機能を有効にすると、実行時間が若干長くなります。これは、演算時間と通信時間を分離するために、barrier 同期をとるためです。

当機能を有効にすると、以下の処理単位で実行時間が表示されます。Eigen ライブラリを call する度に、標準出力に出力されます。

実行時間表示機能の計測範囲

対角化処理の演算時間(秒)
対角化処理の通信時間(秒)
分割統治法の時間(秒)
逆変換処理の演算時間(秒)
逆変換処理の転送時間(秒)

４．使用法

Eigen ライブラリでは、以下のルーチンを利用することができます。
Fortran から呼び出す場合は、ルーチン一覧に記載のルーチン名をそのまま使用します。
C 言語および C++から呼び出す場合は、注意事項があります。「4.3 Eigen ライブラリのリンク方法」を参考にしてください。

4.1. ルーチン一覧

Eigen ライブラリのルーチン群

	ルーチン名	機能
実対称行列	matrix_adjust	Eigen ライブラリでは､2 次元のサイクリック分割を行っている。行列のデータ配置をサイクリック分割に対応させるための変換ルーチンである。当ライブラリを使用する直前に呼び出す必要がある。
	eigen_real_symmetric	実対称行列の固有値固有ベクトルを求める。対角化・固有値・固有ベクトルの一連の動作を実行する。 行列の大きさに応じて、HouseHolder-3 重対角化あるいは NarrowBandReduction-5 重対角化のどちらかが動作する。
	eigen_s	実対称行列を HouseHolder-3 重対角化して固有値固有ベクトルを求める。3 重対角化・固有値・固有ベクトルの一連の動作を実行する。
	eigen_sx	実対称行列を NarrowBandReduction-5 重対角化して固有値固有ベクトルを求める。5 重対角化・固有値・固有ベクトルの一連の動作を実行する。
	eigen_trd	実対称行列を HouseHolder-3 重対角化により 3 重対角化する処理。
	eigen_prd	実対称行列を NarrowBandReduction-5 重対角化により 5 重対角化する処理。
	eigen_dc	実対称行列の HouseHolder-3 重対角化により、3 重対角化された行列を入力とし、分割統治法で固有値を求める処理。
	eigen_dcx	実対称行列の NarrowBandReduction-5 重対角化により、5 重対角化された行列を入力とし、分割統治法で固有値を求める処理。
	eigen_tbk	実対称行列の HouseHolder-3 重対角化に対応する固有ベクトルを求める処理。
	eigen_pbk	実対称行列の NarrowBandReduction-5 重対角化に対応する固有ベクトルを求める処理。
エルミート行列	eigen_hermite	エルミート行列の固有値固有ベクトルを求める。対角化・固有値・固有ベクトルの一連の動作を実行する。
	eigen_hrd	エルミート行列を対角化する処理。
	eigen_dch	エルミート行列について、対角化された行列を入力とし、分割統治法で固有値を求める処理。
	eigen_hbk	エルミート行列に対応した固有ベクトルを求める処理。

4.2. 各ルーチンの引数解説

Eigen ライブラリの各ルーチンの引数について解説します。

	ルーチン名	引数
実対称行列	matrix_adjust	matrix_adjust_s(n, a, b, nm) n　：行列の次元（integer）　（入力） a　：行列を格納した配列（real(8)）　（入力） b　：行列を格納した配列（real(8)）　（入力） nm　：配列 b のサイズ（integer）　（入力）
	eigen_real_symmetric	eigen_real_symmetric(n,a,lda,w,z,ldz,ifl,m) n　：行列の次元（integer）　（入力） a　：行列を格納した配列（real(8)）　（入力） lda：配列 a のサイズ（integer）　（入力） w　：固有値を格納する配列（real(8)）　（出力） z　：固有ベクトルを格納する配列（real(8)）（出力） ldz：配列 z のサイズ（integer）　（入力） ifl：固有ベクトルの要否フラグ（integer）　（入力） 「0」を指定すると、固有値と固有ベクトルを求める。 「1」を指定した場合は、固有値のみを求める。 m　：ブロック化係数（integer）　（入力） （指定した値で処理をブロック化する。32～128 程度を指定すると良い。）
	eigen_s	eigen_s(n,a,lda,w,z,ldz,ifl,m) n　：行列の次元（integer）　（入力） a　：行列を格納した配列（real(8)）　（入力） lda：配列 a のサイズ（integer）　（入力） w　：固有値を格納する配列（real(8)）　（出力） z　：固有ベクトルを格納する配列（real(8)）（出力） ldz：配列 z のサイズ（integer）　（入力） ifl：固有ベクトルの要否フラグ（integer）　（入力） 「0」を指定すると、固有値と固有ベクトルを求める。 「1」を指定した場合は、固有値のみを求める。 m　：ブロック化係数（integer）　（入力） （指定した値で処理をブロック化する。32～128 程度を指定すると良い。）

実対称行列	eigen_sx	eigen_sx(n,a,lda,w,z,ldz,ifl,m) n　：行列の次元（integer）　（入力） a　：行列を格納した配列（real(8)）　（入力） lda：配列 a のサイズ（integer）　（入力） w　：固有値を格納する配列（real(8)）　（出力） z　：固有ベクトルを格納する配列（real(8)）（出力） ldz：配列 z のサイズ（integer）　（入力） ifl：固有ベクトルの要否フラグ（integer）　（入力） 「0」を指定すると、固有値と固有ベクトルを求める。 「1」を指定した場合は、固有値のみを求める。 m　：ブロック化係数（integer）　（入力） （指定した値で処理をブロック化する。32～128 程度を指定すると良い。）
	eigen_trd	eigen_trd(n,a,lda,d,e,m) n　：行列の次元（integer）　（入力） a　：行列を格納した配列（real(8)）　（入力） lda：配列 a のサイズ（integer）　（入力） d　：対角要素を格納する配列（real(8)）　（出力） e　：副対角要素を格納する配列（real(8)）（出力） m　：ブロック化係数（integer）　（入力） （指定した値で処理をブロック化する。32～128 程度を指定すると良い。）
	eigen_prd	eigen_prd(n,a,lda,d,e,lde,m) n　：行列の次元（integer）　（入力） a　：行列を格納した配列（real(8)）　（入力） lda：配列 a のサイズ（integer）　（入力） d　：対角要素を格納する配列（real(8)）　（出力） e　：副対角要素を格納する配列（real(8)）（出力） lde：配列 e のサイズ（integer）　（入力） m　：ブロック化係数（integer）　（入力） （指定した値で処理をブロック化する。32～128 程度を指定すると良い。）
	eigen_dc	eigen_dc(n,w,e,z,ldz,info) n　：行列の次元（integer）　（入力） w　：固有値を格納する配列（real(8)）　（出力） e　：副対角要素を格納する配列（real(8)）　（入力） z　：固有ベクトルを格納する配列（real(8)）（出力） ldz　：配列 z のサイズ（integer）　（入力） info：エラー番号（integer）　（出力）

実対称行列	eigen_dcx	eigen_dcx(n,w,e,lde,z,ldz,info) n　：行列の次元（integer）　（入力） w　：固有値を格納する配列（real(8)）　（出力） e　：副対角要素を格納する配列（real(8)）　（入力） lde　：並列 e のサイズ（integer）　（入力） z　：固有ベクトルを格納する配列（real(8)）（出力） ldz　：配列 z のサイズ（integer）　（入力） info：エラー番号（integer）　（出力）
	eigen_tbk	eigen_tbk(n,a,lda,z,ldz,e,m) n　：行列の次元（integer）　（入力） a　：行列を格納した配列（real(8)）　（入力） lda：配列 a のサイズ（integer）　（入力） z　：固有ベクトルを格納する配列（real(8)）（出力） ldz　：配列 z のサイズ（integer）　（入力） e　：副対角要素を格納する配列（real(8)）　（入力） m　：ブロック化係数（integer）　（入力） （指定した値で処理をブロック化する。128 程度を指定すると良い。）
	eigen_pbk	eigen_pbk(n,a,lda,z,ldz,e,m) n　：行列の次元（integer）　（入力） a　：行列を格納した配列（real(8)）　（入力） lda：配列 a のサイズ（integer）　（入力） z　：固有ベクトルを格納する配列（real(8)）（出力） ldz　：配列 z のサイズ（integer）　（入力） e　：副対角要素を格納する配列（real(8)）　（入力） m　：ブロック化係数（integer）　（入力） （指定した値で処理をブロック化する。128 程度を指定すると良い。）
エルミート行列	eigen_hermite	eigen_hermite(ar,ai,w,n,lda) ar　：行列(実部)を格納した配列（real(8)）　（入出力） 出力時は固有ベクトル(実部)が格納される。 ai　：行列(虚部)を格納した配列（real(8)）　（入出力） 出力時は固有ベクトル(虚部)が格納される。 w　：固有値を格納する配列（real(8)）　（出力） n　：行列の次元（integer）　（入力） lda：配列 a のサイズ（integer）　（入力）

エルミート行列	eigen_hrd	eigen_hrd(n,ar,ai,lda,d,e,e2,tau,m,zr,zi) n　：行列の次元（integer）　（入力） ar　：行列(実部)を格納した配列（real(8)）　（入力） ai　：行列(虚部)を格納した配列（real(8)）　（入力） lda：配列 a のサイズ（integer）　（入力） d　：対角要素を格納する配列（real(8)）　（出力） e　：副対角要素を格納する配列（real(8)）　（出力） e2　：作業用配列（real(8)）　（入力） tau：作業用配列（real(8)）　（入力） m　：ブロック化係数（integer）　（入力） （指定した値で処理をブロック化する。128 程度を指定すると良い。） zr　：固有ベクトル(実部)を格納した配列（real(8)）（入力） zi　：固有ベクトル(虚部)を格納した配列（real(8)）（入力）
	eigen_dch	eigen_dch(n,w,e2,zr,ldz,info) n　：行列の次元（integer）　（入力） w　：固有値を格納する配列（real(8)）　（出力） e2　：作業用配列（real(8)）　（入力） zr　：固有ベクトル(実部)を格納した配列（real(8)）（入力） ldz　：配列 z のサイズ（integer）　（入力） info：エラー番号（integer）　（出力）
	eigen_hbk	eigen_hbk(ar,ai,lda,zr,zi,ldz,tau,n) ar　：行列(実部)を格納した配列（real(8)）　（入力） ai　：行列(虚部)を格納した配列（real(8)）　（入力） lda　：配列 a のサイズ（integer）　（入力） zr　：固有ベクトル(実部)を格納した配列（real(8)）（出力） zi　：固有ベクトル(虚部)を格納した配列（real(8)）（出力） ldz　：配列 z のサイズ（integer）　（入力） tau　：作業用配列（real(8)）　（入力） n　：行列の次元（integer）　（入力）

4.3. Eigen ライブラリのリンク方法

Eigen の各ライブラリは、「libEigen_s.a」(3 重対角化版),「libEigen_sx.a」(5 重対角化版)、「libEigen_h.a」(エルミート版) というファイル名でインストールされます。よって、リンク時に、「-lEigen_s」、「-lEigen_sx」、または、「-lEigen_s」というオプションを指定することでリンクすることが出来ます。また、必要に応じてライブラリをインストールしたディレクトリパスも指定する必要があります。

4.3.1. fortran プログラムで使用する場合(3 重対角化版)

リンクオプションでライブラリへのパス(例：/home/lib)を指定する場合

```
mpifrt -o prog prog.f -L/home/lib -lEigen_s
```

環境変数でライブラリへのパス(例：/home/lib)を指定する場合

```
LD_LIBRARY_PATH=/home/lib:$LD_LIBRARY_PATH

mpifrt -o prog prog.f -lEigen_s
```

4.3.2. C 言語プログラムで使用する場合

Eigen ライブラリは Fortran で作成されています。よって、C 言語から呼び出す場合は、C 言語と Fortran を結合する必要があります。以下の手順でリンクしてください。なお、GNU, Intel, Fujitsu の各コンパイラは以下の手順で対応できます。

1）C 言語のメイン(main)を変更します。
main を大文字にし、アンダースコアを 2 つ挿入します。
例： main() ——> MAIN__()

2）全ての C 言語のオブジェクトを生成します。
例：fcc –c *.c

3）Fortran コンパイラで C 言語のオブジェクトと Eigen ライブラリをリンクします。
例：frt *.o -lEigen

以上