2016年度 第22回 公開ソフト講習会
「KMATHLIB」

理化学研究所 計算科学研究機構
研究部門 大規模並列数値計算技術研究チーム

今村 俊幸（チームリーダー）
大井 祥栄（特別研究員）
廣田 悠輔（特別研究員）
椋木 大地（特別研究員）

本講習会では一般社団法人 高度情報科学技術研究機構
が所有するスーパーコンピュータを利用しています.

講習会プログラム

- 前半: 座学(1時間程度を予定)
- 後半: ハンズオン(2時間程度を予定)

- 質問は随時受け付けます
- スライド資料とハンズオン資料は端末に保存済み
  ※2次配布等はご遠慮ください

講習会プログラム

- はじめに
- KMATHLIB API
  - 概要・特徴
  - データ構造
- 利用方法
  - インターフェース
  - プラグイン
- ハンズオン
- Q&A

講習会プログラム

- はじめに
- KMATHLIB API
  - 概要・特徴
  - データ構造
- 利用方法
  - インターフェース
  - プラグイン
- ハンズオン
- Q&A

はじめに

- 昨今の計算科学ソフトウェア(アプリケーション)開発では数値計算ライブラリの利用が必要不可欠.
- ライブラリ毎に使用方法やデータ構造が異なるため, これらの差異を理解し必要に応じてデータ構造変換機能等の実装が必要.

- プログラムの煩雑化, アプリケーション開発・保守の負担増
(プログラムが大規模な程顕著).

- アプリケーション開発・保守の負担軽減が必須.

はじめに

KMATHLIB API

- 共通のインターフェースを介して数値計算ライブラリを統一的に利用するためのソフトウェア(フレームワーク).
- 様々なライブラリのアプリケーションへの導入を支援し，長期的な開発・保守の負担軽減が可能.

プログラム例 (Fortran90)

```
<パラメタ設定等のプリプロセス>
<行列A, Bの設定>

call eigen_s(n, n, a, nm, w, z, nm, m_forward = 32, m_backward = 128)
                                                    EigenExaの呼び出し
ictxt = eigen_get_blacs_context()
call descinit(descb, n, n, mb, nb, 0, 0, ictxt, lrow, ierror)
call descinit(descxy, n, n, mb, nb, 0, 0, ictxt, lrow, ierror)
call descinit(descw, n, n, 1, 1, 0, 0, ictxt, nm, ierror)
call pdlacpy('A', n, n, xy, 1, 1, descxy, w, 1, 1, descw)
                                                    データ
                                                    構造変換
call pdgesv(n, n, b, 1, 1, descb, ipiv, xy, 1, 1, descxy, ierror)
                                                    ScaLAPACKの呼び出し
<計算結果を用いたポストプロセス>
```

KMATHLIB APIを利用しない場合
(ライブラリを直接呼び出す場合)

```
<パラメタ設定等のプリプロセス>
<行列A,Bの設定>

call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DEIG_EigenExa, ierr)
call KMATH_Set_Parameter(h, 'A', A, ierr)
call KMATH_Set_Parameter(h, 'X', D, ierr)
call KMATH_Set_Parameter(h, 'Z', Y, ierr)
call KMATH_Solve(h, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLSM_ScaLAPACK, ierr)
call KMATH_Set_Parameter(h, 'A', B, ierr)
call KMATH_Set_Parameter(h, 'B', Y, ierr)
call KMATH_Set_Parameter(h, 'X', X, ierr)
call KMATH_Solve(h, ierr)

call KMATH_Destroy(h, ierr)

<計算結果を用いたポストプロセス>
```

KMATHLIB APIを利用した場合

- 実対称行列A，正則行列B，対角行列D，正則行列XとするABX=BXDに対して，(1)固有値問題AY=YDをEigenExaを用いて解き，(2)その結果を用いた密行列連立一次方程式BX=YをScaLAPACKを用いて解くプログラム．
- EigenExaの利用，データ構造変換，ScaLAPACKの利用部分を抜粋．

プログラム例 (Fortran90)

```
<パラメタ設定等のプリプロセス>
<行列A, Bの設定>

call eigen_s(n, n, a, nm, w, z, nm, m_forward = 32, m_backward = 128)
                                                    EigenExaの呼び出し
ictxt = eigen_get_blacs_context()
call descinit(descb, n, n, mb, nb, 0, 0, ictxt, lrow, ierror)
call descinit(descxy, n, n, mb, nb, 0, 0, ictxt, lrow, ierror)
call descinit(descw, n, n, 1, 1, 0, 0, ictxt, nm, ierror)
call pdlacpy('A', n, n, xy, 1, 1, descxy, w, 1, 1, descw)
                                                    データ
                                                    構造変換
call pdgesv(n, n, b, 1, 1, descb, ipiv, xy, 1, 1, descxy, ierror)
                                                    ScaLAPACKの呼び出し
<計算結果を用いたポストプロセス>
```

KMATHLIB APIを利用しない場合
(ライブラリを直接呼び出す場合)

```
<パラメタ設定等のプリプロセス>
<行列A,Bの設定>

call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DEIG_EigenExa, ierr)
call KMATH_Set_Parameter(h, 'A', A, ierr)
call KMATH_Set_Parameter(h, 'X', D, ierr)
call KMATH_Set_Parameter(h, 'Z', Y, ierr)
call KMATH_Solve(h, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLSM_ScaLAPACK, ierr)
call KMATH_Set_Parameter(h, 'A', B, ierr)
call KMATH_Set_Parameter(h, 'B', Y, ierr)
call KMATH_Set_Parameter(h, 'X', X, ierr)
call KMATH_Solve(h, ierr)

call KMATH_Destroy(h, ierr)

<計算結果を用いたポストプロセス>
```

KMATHLIB APIを利用した場合

- KMATHLIB APIを利用した場合，データ構造変換の明示的な記述が不要．
- 書式が統一化されており，処理の理解が容易．
- プログラムの行数が増える場合もある．

講習会の目的

- KMATHLIB APIの特徴・機能・使い方の理解
- 自身のソフトウェアに応用するスキルの習得

講習会プログラム

- はじめに
- KMATHLIB API
  - 概要・特徴
  - データ構造
- 利用方法
  - インターフェース
  - プラグイン
- ハンズオン
- Q&A

KMATHLIB API

- 開発および保守: 大規模並列数値計算技術研究チーム
  - 今村 俊幸 (チームリーダー)
  - 大井 祥栄 (特別研究員)
  - 廣田 悠輔 (特別研究員)
  - 椋木 大地 (特別研究員)
  - Tan Yiyu (研究員)
- 問い合わせ, バグの報告等は以下のアドレスへお願いします.

kmathlib[at]riken.jp ([at]を@に変更してください)

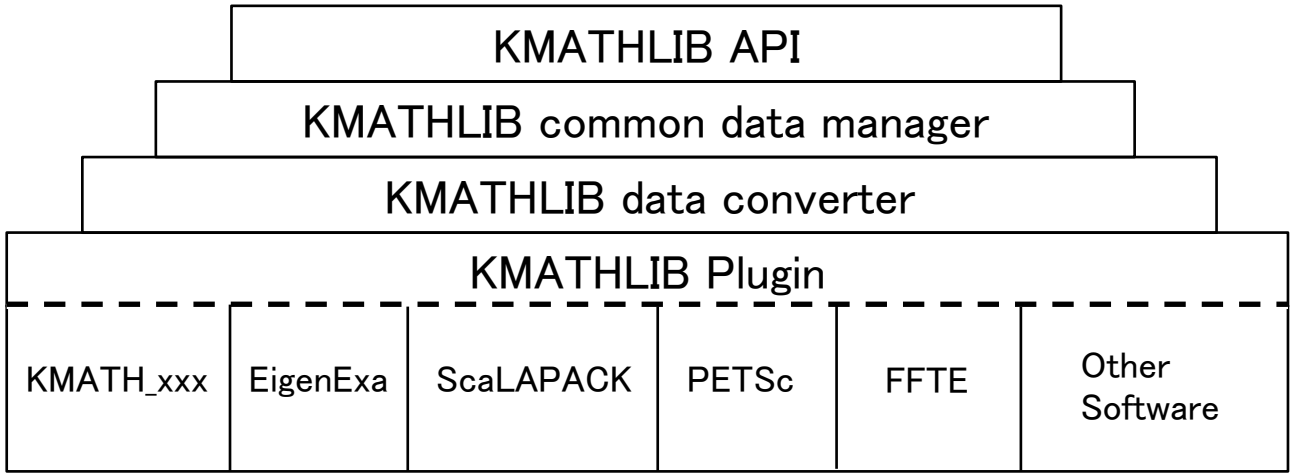
類似ソフトウェア

- PETSc(Portable Extensible Toolkit for Scientific Computation)
  - ANL(Argonne National Laboratory)で開発されているオープンソースのライブラリ.
  - 様々なデータ形式や(主に疎行列の)連立一次方程式解法を豊富に含む.
  - C, C++, Fortran 77/90/95で利用可能.
  - 実数版と複素数版に分かれており, 目的に応じて利用.
  - 様々なシステムで利用可能.
    - ✓京コンピュータ(ver. 3.4.3)
    - ✓本講習会ではRIST FX10にインストールしたver. 3.6.3を利用.

PETScとKMATLIB API

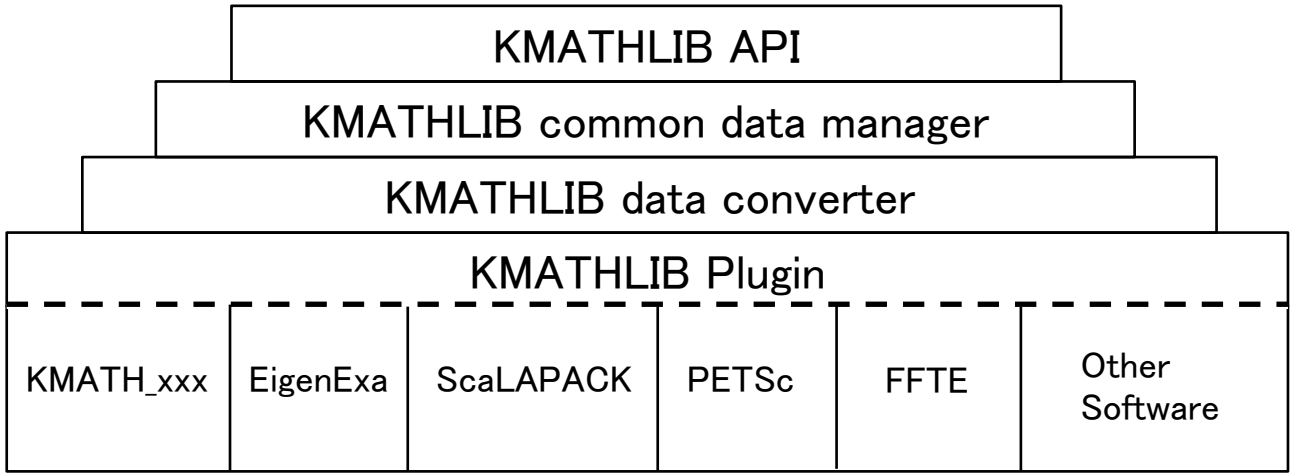
- KMATHLIB APIとPETScはアプリケーション開発を支援する点では類似.
- KMATHLIB APIはPETScを含むより幅広いライブラリを利用可能.
- KMATHLIB APIではPETScの一部の機能を利用.
  - データ構造.
  - 行列・ベクトルの定義.
  - 値の入力.
- PETScの使用方法もある程度知る必要がある.

KMATHLIB APIの基本構成



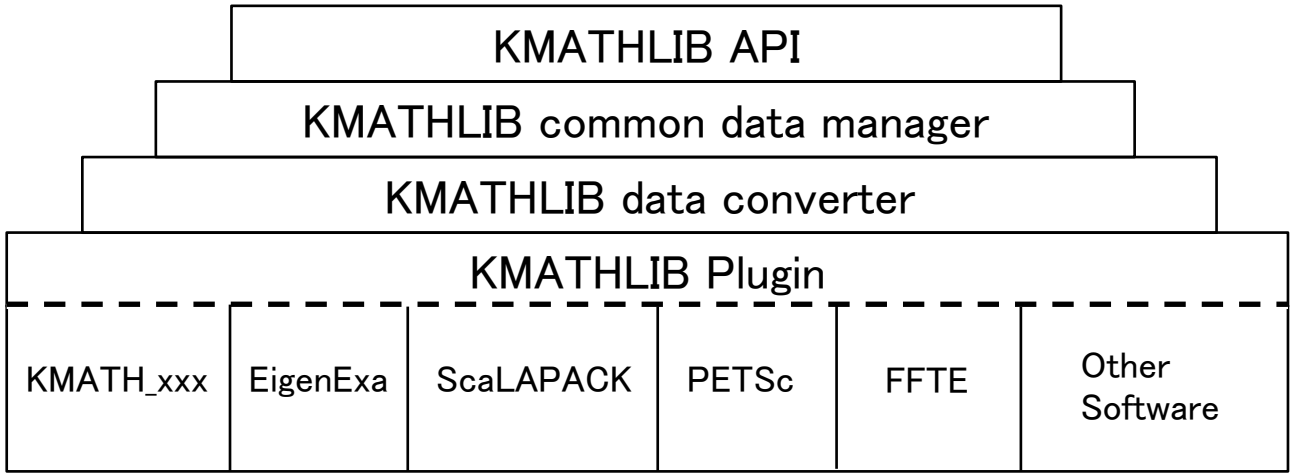
- 階層構造
  - ライブラリ利用の為のプラグイン部とそれ以外の補助機能をまとめたコア部から成る
  - ライブラリは最下層に位置し，プラグインを介して呼び出される．
  - ユーザはフロントエンドのAPIを利用．
  - KMATHLIB API自体は数値計算機能をもたず，ライブラリを用いることで計算が可能．

KMATHLIB APIのメリット



- ライブラリ毎のデータ構造や使用方法の差異は中間層に位置する補助機能により吸収される(利用者が考慮不要).
- 高度に抽象化されたAPIにより, 統一的なプログラムの記述で様々なライブラリが利用可能.

KMATHLIB APIのメリット



■ 任意の数値計算ライブラリを組み込み可能.

➢高い拡張性を保持.

インストール・公開状況

- チームWebサイト[1]にて公開中.
- 京コンピュータにインストール済.
- 本講習会ではRIST[2] FX10でビルド・インストールしたバイナリを利用

[1] http://www.aics.riken.jp/labs/lpnctrt/KMATHLIB_API.html

[2] 一般社団法人 高度情報科学技術研究機構 (RIST)

動作概要

- 京コンピュータ，富士通 FX10，FX100，x86系のクラスタシステム上での動作を確認．
  - アーキテクチャ非依存で様々なシステムで動作する汎用性をもつ．
  - 但し，対象とする数値計算ライブラリの正常動作が必要．
- 対応言語はFortran 90/95．

講習会プログラム

- はじめに
- KMATHLIB API
  - 概要・特徴
  - データ構造
- 利用方法
  - インターフェース
  - プラグイン
- ハンズオン
- Q&A

データ構造変換

- KMATHLIB APIではデータ構造の異なるライブラリを統一的に利用するため，標準のデータ構造(標準形式)をもつ．
  - KMATHLIB APIでは標準形式として，PETScで用いられる形式を採用している．
  - プラグインを介してライブラリを呼び出す際，ライブラリで用いられる形式(内部形式)に自動的にデータ構造変換を行う．計算終了後，再び標準形式に変換される．
  - データ構造変換の必要性を判定し，不要な変換を一部回避．

PETScのデータ構造

- PETScは独自のデータ構造を採用.
  - PETScがデータを制御しているため，ユーザからは実態の確認不可.
  - 内部フォーマットはいくつか存在.
  - 行列・ベクトル等はハンドラ変数を使ってアクセス.
    - ✓行列: Mat A
    - ✓ベクトル: Vec b

PETScのデータ構造

- PETScがサポートする入出力データ構造
  - AIJ(CSR形式)
  - Block AIJ
  - Symmetric Brock AIJ
  - MatrixMatket (*.mtx)
  - Dense

講習会プログラム

- はじめに
- KMATHLIB API
  - 概要・特徴
  - データ構造
- **利用方法**
  - **インターフェース**
  - プラグイン
- ハンズオン
- Q&A

KMATHLIB API利用方法

```
program test

use kmath_lib_mod
#include <finclude/petscsys.h>
…

Mat :: A
Vec :: B, X
…

call PetscInitialize(PETSC_NULL_CHARACTER, ierr)
call MPI_Comm_rank(PETSC_COMM_WORLD,rank,ierr)

call MatCreateDense(PETSC_COMM_WORLD, PETSC_DECIDE, &
PETSC_DECIDE, m, n, PETSC_NULL_SCALAR, A, ierr)
…

if (rank == 0) then
 do i = 0, m - 1
…

call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLS_ScaLAPACK, ierr)
call KMATH_Set_Parameter(h, 'A', A, ierr)
…
call KMATH_Solve(h, ierr)

call KMATH_Destroy(h, ierr)

write*

call PetscFinalize(ierr)

end program test
```

モジュールとヘッダファイルの読み込み

変数等の定義

PETScとMPIの初期化

行列・ベクトル等の生成値の代入

KMATHLIB APIを用いた処理

計算結果の出力等

PETScの終了

モジュールとヘッダの読み込み

```
program test

use kmath_lib_mod
use kmath_plugin_dls_scalapack_mod

implicit none

#include <finclude/petscsys.h>
#include <finclude/petscvec.h>
#include <finclude/petscmat.h>
```

KMATHLIB APIメインモジュールの読み込み

プラグインモジュールの読み込み
kmath_plugin_dls_scalapack_modは
DLS(Dense Linear System)を
ScaLAPACKを用いて解くプラグイン
モジュール

PETScヘッダファイルの読み込み
petscsys.h: PETScのメインファイル
petscvec.h: ベクトル
petscmat.h: 行列
行列やベクトルを扱う場合，それぞれに対応するヘッダファイルを読み込む必要がある
implicit none(暗黙の型宣言の無効化)より後に記述

変数等の定義

```
Mat :: A
Vec :: B, X, Y

PetscInt :: ierr
PetscReal :: norm, r
PetscScalar :: v(1), r1
integer :: sn, rank
integer :: m, n, i, j, iar(1), jar(1)
integer, allocatable :: seed(:)
type(s_context) :: h
```

標準(PETSc)形式の行列とベクトルの定義

標準形式の変数定義
PetscInt: 整数型
PetscReal: 実数型
PetscScalar: 実数または複素数型
(実数版と複素数版で異なる)

KMATHLIB APIのコンテキストハンドル(KMATHLIB APIインターフェース)

- type(s_context) :: context
  - KMATHLIB API コンテキストの構造体

メンバ	型	説明
c	integer(8)	コンテキストの実体. Cによってアロケートされたメモリ領域へのポインタ値が格納される.

PETScとMPIの初期化

```
call random_seed(size = sn)
allocate(seed(sn))
seed = 123;
call random_seed(put = seed)
deallocate(seed)

call PetscInitialize(PETSC_NULL_CHARACTER, ierr)

call MPI_Comm_rank(PETSC_COMM_WORLD, rank, ierr)
```

乱数の種生成

PETScの初期化

MPIの初期化

行列の生成

```
m = 2000
n = 2000

call MatCreateDense(PETSC_COMM_WORLD,
PETSC_DECIDE, PETSC_DECIDE, m, n,
PETSC_NULL_SCALAR, A, ierr)
```

行列のサイズ定義

行列Aを2000×2000の密行列として生成

ベクトルの生成

```
call VecCreate(PETSC_COMM_WORLD, B, ierr)
call VecSetType(B, VECMPI, ierr)
call VecSetSizes(B, PETSC_DECIDE, n, ierr)

call VecCreate(PETSC_COMM_WORLD, X, ierr)
call VecSetType(X, VECMPI, ierr)
call VecSetSizes(X, PETSC_DECIDE, n, ierr)

call VecDuplicate(X, Y, ierr)
```

ベクトルBを2000次元ベクトルとして生成

ベクトルXを2000次元ベクトルとして生成

ベクトルYにベクトルXをコピー

行列への値の代入

```
if (rank == 0) then
  do i = 0, m - 1
    iar(1) = i
    do j = 0, n - 1
      jar(1) = j
      call random_number(r)
      r1 = r
      v(1) = r1
      call MatSetValues(A, 1, iar, 1, jar, v, INSERT_VALUES, ierr)
    end do
  end do
end if

call MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY, ierr)
call MatAssemblyEnd  (A, MAT_FINAL_ASSEMBLY, ierr)
```

乱数の生成

行列Aのiar行jar列に乱数vを代入

行列の組立

ベクトルへの値の代入

```
if (rank == 0) then
  do i = 0, n - 1
    iar(1) = i
    call random_number(r)
    r1 = r
    v(1) = r1
    call VecSetValues(B, 1, iar, v, INSERT_VALUES, ierr)
  end do
end if

call VecAssemblyBegin(B, ierr)
call VecAssemblyEnd  (B, ierr)
```

乱数の生成

ベクトルBのiar次元目に乱数vを代入

ベクトルの組立

KMATHLIB APIを用いた処理

```
call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLS_ScaLAPACK, ierr)

call KMATH_Set_Parameter(h, 'A', A, ierr)
call KMATH_Set_Parameter(h, 'B', B, ierr)
call KMATH_Set_Parameter(h, 'X', X, ierr)

call KMATH_Solve(h, ierr)

call KMATH_Destroy(h, ierr)
```

KMATHLIB APIの初期化

プラグイン指定

パラメータ指定

計算の実行

KMATHLIB APIの終了

インターフェース (KMATH_*)

- KMATH_Create(handle, comm, ierr)
  - MPIコミュニケータを指定してKMATHLIB APIを初期化.
  - メモリを確保し, コンテキストハンドルを返す.
  - KMATHLIB APIのインターフェースの中で最初に呼ばれる.

引数	型	I/O	説明
handle	type(s_context)	inout	対象のコンテキストハンドル
comm	integer	in	MPIコミュニケータ
ierr	integer	out	エラー値 “0”で正常終了

インターフェース (KMATH_*)

- KMATH_Set_Plugin(handle, plugin_setup, ierr)
  - プラグインエントリポイントを指定してプラグインを登録.
  - 以前のプラグインが設定されており, 計算結果が内部に保留されていた場合, PETSc出力変数にその内容が書き出される.

引数	型	I/O	説明
handle	type(s_context)	inout	対象のコンテキストハンドル
plugin_setup	interface	in	プラグインエントリポイント
ierr	integer	out	エラー値 “0”で正常終了

インターフェース (KMATH_*)

- KMATH_Set_Parameter (handle, key, val, ierr)
  - 指定したパラメータの値を設定.
  - 入力データおよび出力データの行列やベクトルの変数(PETSc形式)もこのインターフェースを介してKMATHLIB APIに設定.

引数	型	I/O	説明
handle	type(s_context)	inout	対象のコンテキストハンドル
key	character(*)	in	パラメータ名
val	integer double precision character(*) PetscFortranAddr	in	設定する値
ierr	intger	out	エラー値 “0”で正常終了

インターフェース (KMATH_*)

- KMATH_Set_Parameters (handle, key, val, ierr)
  - 指定したパラメータの値を設定.
  - 入力データおよび出力データの行列やベクトルの変数(PETSc形式)もこのインターフェースを介してKMATHLIB APIに設定.

引数	型	I/O	説明
handle	type(s_context)	inout	対象のコンテキストハンドル
key	character(*)(:)	in	パラメータ名
val	integer(:) double precision(:) character(*) (:) PetscFortranAddr(:)	in	設定する値
ierr	intger	out	エラー値 “0”で正常終了

インターフェース (KMATH_*)

- KMATH_Solve(handle, ierr)
  - 計算の実行.
  - 必要に応じて標準形式(PETSc形式)のが内部形式に変換.
  - 呼び出し終了時点では, 計算結果は内部に保留.
  - エラー値として”-1”が返された場合は計算失敗.

引数	型	I/O	説明
handle	type(s_context)	inout	対象のコンテキストハンドル
ierr	integer	out	エラー値 “0”で正常終了

インターフェース (KMATH_*)

- KMATH_Flush (handle, ierr)
  - 内部状態をフラッシュ.
  - 計算結果が内部に保留されていた場合, この呼び出しにより PETSc出力変数に内容が書き出される.

引数	型	I/O	説明
handle	type(s_context)	inout	対象のコンテキストハンドル
ierr	integer	out	エラー値 “0”で正常終了

インターフェース (KMATH_*)

- KMATH_Destroy(handle, ierr)
  - KMATHLIB APIを終了.
  - ハンドルに紐付けられたコンテキストを解放.
  - 計算結果が内部に保留されていた場合, PETSc出力変数に内容が書き出される.
  - KMATHLIB APIのインターフェースで最後に呼ばれる.

引数	型	I/O	説明
handle	type(s_context)	inout	対象のコンテキストハンドル
ierr	integer	out	エラー値 “0”で正常終了

KMATHLIB APIを用いた処理

```
call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLS_ScaLAPACK, ierr)

call KMATH_Set_Parameter(h, 'A', A, ierr)
call KMATH_Set_Parameter(h, 'B', B, ierr)
call KMATH_Set_Parameter(h, 'X', X, ierr)

call KMATH_Solve(h, ierr)

call KMATH_Destroy(h, ierr)
```

KMATHLIB APIの初期化

プラグイン指定

パラメータ指定

計算の実行

KMATHLIB APIの終了

計算結果の出力等

```
call MatMult(A, X, Y, ierr)

call VecAXPY(Y, -1.0d0, B, ierr)

call VecNorm(Y, NORM_2, norm, ierr)

if (rank == 0) then
  print*,'diff(2NORM) = ',norm
end if
```

AXを計算し，Yへ代入

Y-Bを計算し，Yへ代入

YのL2ノルムを計算し，normへ代入

標準出力へnormを表示

PETScの終了

```
call VecDestroy(Y, ierr)
call VecDestroy(X, ierr)
call VecDestroy(B, ierr)
call MatDestroy(A, ierr)

call PetscFinalize(ierr)

end program test
```

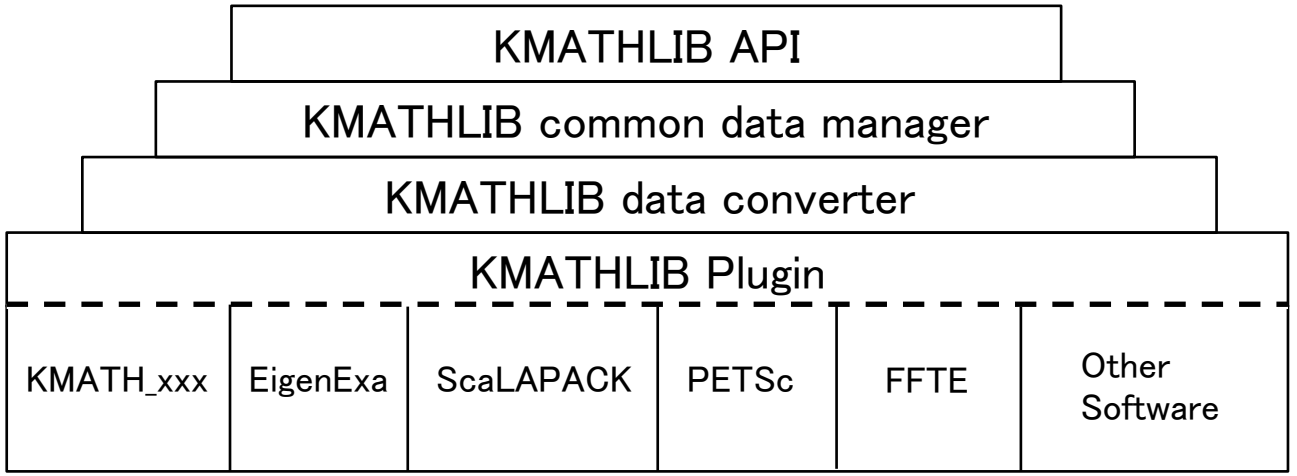
行列・ベクトルの解放

PETScの終了

講習会プログラム

- はじめに
- KMATHLIB API
  - 概要・特徴
  - データ構造
- **利用方法**
  - インターフェース
  - **プラグイン**
- ハンズオン
- Q&A

プラグイン



- ライブラリはプラグインを介して呼び出される.
- プラグインはデータ構造変換ルーチンや計算ライブラリを呼び出すルーチンから構成.
- いくつかの標準プラグインが同梱.

標準プラグイン

- 密行列固有値解法
- 密行列連立一次方程式解法
- 疎行列連立一次方程式解法
- フーリエ変換法
- 密行列特異値分解法
- 乱数生成法

プラグイン利用方法

```
call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLS_ScaLAPACK, ierr)

call KMATH_Set_Parameter(h, 'A', A, ierr)
call KMATH_Set_Parameter(h, 'B', B, ierr)
call KMATH_Set_Parameter(h, 'X', X, ierr)

call KMATH_Solve(h, ierr)

call KMATH_Destroy(h, ierr)
```

① ②

③

① 利用するプラグインの指定.

② ライブラリで扱うパラメータの指定.

③ 計算の実行.

①プラグインの指定

```
call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLS_ScaLAPACK, ierr)
                                                                  ①
call KMATH_Set_Parameter(h, 'A', A, ierr)
call KMATH_Set_Parameter(h, 'B', B, ierr)
call KMATH_Set_Parameter(h, 'X', X, ierr)

call KMATH_Solve(h, ierr)

call KMATH_Destroy(h, ierr)
```

- 全てのプラグインに対してライブラリ名，プラグインモジュール名，エントリポイント名が定義されており，適切に指定．

① プラグインの指定

- プラグイン名: 利用するプラグインの指定に利用(Makefile等に記述)
- プラグインモジュール名: 読み込むプラグインモジュールの指定に利用(プログラムの冒頭に記述)
- エントリポイント名: 利用するプラグインの指定に利用(プログラムのプラグインを利用する箇所)

②パラメータの指定

```
call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLS_ScaLAPACK, ierr)

call KMATH_Set_Parameter(h, 'A', A, ierr)
call KMATH_Set_Parameter(h, 'B', B, ierr)
call KMATH_Set_Parameter(h, 'X', X, ierr)

call KMATH_Solve(h, ierr)

call KMATH_Destroy(h, ierr)
```

②

■ プラグイン毎に扱うパラメータが定義されており，適切に指定する必要がある．

➢ パラメータが複数ある場合は，それぞれ指定する．

③計算の実行

```
call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLS_ScaLAPACK, ierr)

call KMATH_Set_Parameter(h, 'A', A, ierr)
call KMATH_Set_Parameter(h, 'B', B, ierr)
call KMATH_Set_Parameter(h, 'X', X, ierr)

call KMATH_Solve(h, ierr)    ③

call KMATH_Destroy(h, ierr)
```

■ 指定したプラグインとパラメータを用いて計算を実行.

プラグイン利用の注意事項

```
call KMATH_Create(h, PETSC_COMM_WORLD, ierr)

call KMATH_Set_Plugin(h, KMATH_Plugin_Setup_DLS_ScaLAPACK, ierr)

call KMATH_Set_Parameter(h, 'A', A, ierr)
call KMATH_Set_Parameter(h, 'B', B, ierr)
call KMATH_Set_Parameter(h, 'X', X, ierr)

call KMATH_Solve(h, ierr)
```

③

```
call KMATH_Destroy(h, ierr)
```

- ③計算の実行が終了しても計算結果は出力変数に書き出されない
  - KMATH_Set_Plugin, KMATH_Flush, KMATH_Destroyが呼ばれた場合，計算結果が出力変数に書き出される．
- KMATH_Destroy, KMATH_Flushが呼ばれるまでプラグインやパラメータの情報は残る．

標準プラグイン

- 密行列の固有値問題を EigenExa を用いて解く
  - kmp_deig_eigenexa (ライブラリ名)
  - kmath_plugin_deig_eigenexa_mod (プラグインモジュール名)
  - KMATH_Plugin_Setup_DEIG_EigenExa (エントリポイント名)

パラメータ	型	I/O	説明
'A'	Mat (PETSc形式)	入力	固有対を求める行列
'X'	Vec (PETSc形式)	出力	固有値を並べたベクトル
'Z'	Mat	出力	各列に固有ベクトルを並べた行列

標準プラグイン

■ 密行列固有値問題を LAPACK を用いて解く

- kmp_deig_lapack
- kmath_plugin_deig_lapack_mod
- KMATH_Plugin_Setup_DEIG_LAPACK

パラメータ	型	I/O	説明
'A'	Mat	入力	固有対を求める行列
'X'	Vec	出力	固有値を並べたベクトル
'Z'	Mat	出力	各列に固有ベクトルを並べた行列

標準プラグイン

■ 密行列固有値問題を ScaLAPACK を用いて解く

- kmp_deig_scalapack
- kmath_plugin_deig_scalapack_mod
- KMATH_Plugin_Setup_DEIG_ScaLAPACK

パラメータ	型	I/O	説明
'A'	Mat	入力	固有対を求める行列
'X'	Vec	出力	固有値を並べたベクトル
'Z'	Mat	出力	各列に固有ベクトルを並べた行列

標準プラグイン

■ 密行列固有値問題を SSL II を用いて解く

- kmp_deig_ssl2
- kmath_plugin_deig_ssl2_mod
- KMATH_Plugin_Setup_DEIG_SSL2

パラメータ	型	I/O	説明
'A'	Mat	入力	固有対を求める行列
'X'	Vec	出力	固有値を並べたベクトル
'Z'	Mat	出力	各列に固有ベクトルを並べた行列

標準プラグイン

■ 密行列連立一次方程式を SSL II を用いて解く

- kmp_dls_ssl2
- kmath_plugin_dls_ssl2_mod
- KMATH_Plugin_Setup_DLS_SSL2

パラメータ	型	I/O	説明
‘A’	Mat	入力	係数行列
‘B’	Vec	入力	右辺ベクトル
‘matrix_type’	character	入力	係数行列のタイプ 一般行列: ”general” 実対称行列: ”symmetric”
‘X’	Vec	出力	解ベクトル

標準プラグイン

- 密行列連立一次方程式を LAPACK を用いて解く
  - kmp_dls_lapack
  - kmath_plugin_dls_lapack_mod
  - KMATH_Plugin_Setup_DLS_LAPACK

パラメータ	型	I/O	説明
‘A’	Mat	入力	係数行列
‘B’	Vec	入力	右辺ベクトル
‘matrix_type’	character	入力	係数行列のタイプ 一般行列: ”general” 実対称行列: ”symmetric”
‘X’	Vec	出力	解ベクトル

標準プラグイン

- 密行列連立一次方程式を ScaLAPACK を用いて解く
  - kmp_dls_scalapack
  - kmath_plugin_dls_scalapack_mod
  - KMATH_Plugin_Setup_DLS_ScaLAPACK

パラメータ	型	I/O	説明
‘A’	Mat	入力	係数行列
‘B’	Vec	入力	右辺ベクトル
‘matrix_type’	character	入力	係数行列のタイプ 一般行列: ”general” 実対称行列: ”symmetric”
‘X’	Vec	出力	解ベクトル

標準プラグイン

- 複数右辺密行列連立一次方程式を ScaLAPACK を用いて解く
  - kmp_dlsm_scalapack
  - kmath_plugin_dlsm_scalapack_mod
  - KMATH_Plugin_Setup_DLSM_ScaLAPACK

パラメータ	型	I/O	説明
‘A’	Mat	入力	係数行列
‘B’	Mat	入力	各列に右辺ベクトルを並べた行列
‘matrix_type’	character	入力	係数行列のタイプ 一般行列: ”general” 実対称行列: ”symmetric”
‘X’	Mat	出力	各列に解ベクトルを並べた行列

標準プラグイン

■ 疎行列連立一次方程式を PETSc を用いて解く

- kmp_sls_petsc
- kmath_plugin_sls_petsc_mod
- KMATH_Plugin_Setup_SLS_PE

パラメータ	型	I/O	説明
‘A’	Mat	入力	係数行列
‘B’	Vec	入力	右辺ベクトル
‘matrix_type’	character	入力	係数行列のタイプ 一般行列: ”general” 実対称行列: ”symmetric”
‘X’	Vec	出力	解ベクトル

標準プラグイン

■ フーリエ変換を FFTW を用いて実行

- kmp_fft_fftw
- kmath_plugin_fft_fftw_mod
- KMATH_Plugin_Setup_FFT_FFTW

パラメータ	型	I/O	説明
‘dimension’	integer	入力	入力データの次元 1次元(n): n行 1列の行列を作成 2次元(n,m): n行 m列の行列を作成 3次元(n,m,o): n x m行 o列の行列を作成
‘A’	Mat	入力	入力データとなる行列
‘direction’	character	入力	順変換: ‘forward’ 逆変換: ‘backward’
‘size_x’ ‘size_y’ ‘size_z’	integer	入力	入力データの各方向のサイズ
‘X’	Mat	出力	出力データとなる行列

標準プラグイン

■ 3次元フーリエ変換を KMATH_FFT3D を用いて実行

- kmp_fft_kmath_fft3d
- kmath_plugin_fft_kmath_fft3d_mod
- KMATH_Plugin_Setup_FFT_KMATH_FFT3D

パラメータ	型	I/O	説明
'A'	Mat	入力	入力データとなる行列
'direction'	character	入力	順変換: 'forward' 逆変換: 'backward'
'size_x' 'size_y' 'size_z'	integer	入力	入力データの各方向のサイズ
'X'	Mat	出力	出力データとなる行列

標準プラグイン

■ 密行列特異値分解を ScaLAPACK を用いて解く

- kmp_dsvd_scalapack
- kmath_plugin_dsvd_scalapack_mod
- KMATH_Plugin_Setup_DSVD_ScaLAPACK

パラメータ	型	I/O	説明
'A'	Mat	入力	入力データとなる行列
'S'	Vec	出力	特異値を並べたベクトル
'U'	Mat	出力	各列に左特異値ベクトルを並べた行列
'X'	Mat	出力	各列に右特異値を並べた行列の転置行列

標準プラグイン

- [1, 2) に一様に分布する疑似乱数生成を KMATH_Random を用いて実行
  - kmp_rnd_kmath_random
  - kmath_plugin_rnd_kmath_random_mod
  - KMATH_Plugin_Setup_Rnd_KMATH_Random

パラメータ	型	I/O	説明
'X'	Vec	出力	生成された疑似乱数列

プラグイン開発

- プラグインは任意に追加可能であり，標準プラグインと追加プラグインは同様に扱われる．
- 標準形式と内部形式のデータ構造変換をプラグインに記述できれば，任意のライブラリをプラグインに追加可能．
- プラグイン開発のためのテンプレートおよびフレームワークを提供．