Computer simulations create the future

PETSc for Parallel FEM

RIKEN CCS HPC Summer School
Takeshi Terao, Thosiyuki Imamura

Sample files

Please access /vol0001/ra020019/1D-PETSc

```
$ cd /vol0400/data/ra020019/<YID>_data/
$ cp /col0001/ra020019/1D-PETSc .
$ cd 1D-PETSc/C
$ ls
1d2-petsc.c  ex1.c   Makefile
input.dat    job.sh  job_ex.sh
```

Numerical Library
What is it? For what?

Numerical Library

- **Numerical Library is one of building blocks for ENSURING your advanced programming.**
- **It supports an API for very complex mathematical features, algorithm, schemes, also data handling…**
  - Solving sys. Eqs, FFT, Eigenvalue calculation, SVD, minimization, statistics, etc…
- **There are reference codes.**
  - They might be examples of good (bad) programming.

```
*
*       Form  C := alpha*A*B + beta*C.
*
        DO 90 j = 1,n
           IF (beta.EQ.zero) THEN
              DO 50 i = 1,m
                 c(i,j) = zero
 50           CONTINUE
           ELSE IF (beta.NE.one) THEN
              DO 60 i = 1,m
                 c(i,j) = beta*c(i,j)
 60           CONTINUE
           END IF
           DO 80 l = 1,k
              temp = alpha*b(l,j)
              DO 70 i = 1,m
                 c(i,j) = c(i,j) + temp*a(i,l)
 70           CONTINUE
 80        CONTINUE
 90     CONTINUE
```

http://www.netlib.org/lapack/explore-html/d1/d54/group__double__blas__level3_gaeda3cbd99c8fb834a60a6412878226e1.html

Numerical Library

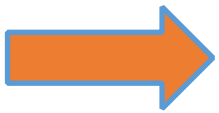
- **Numerical Library is one of building blocks for ENSURING your advanced programming.**
- **It supports an API for very complex mathematical features, algorithm, schemes, also data handling…**
  - Solving sys. Eqs, FFT, Eigenvalue calculation, SVD, minimization, statistics, etc…
- **There are reference codes.**
  - They might be examples of good (bad) programming.
- **It provides us with better performance and finer accuracy than what you made.**
  - Commercial library: faster and more accurate but expensive
  - Open Source library: fast and free (sometimes faster than commercial library)
  - You must check them before you run your application codes.

Example

- **When you HOPE to SPEED UP your code bottlenecked in matmul, use (or link) an appropriate BLAS (Basic Linear Algebra Subprograms) library!**

  - Standard API for linear algebra kernels (case of (C)BLAS).
    - GEMM : Matrix-matrix multiplication ($C := \alpha AB + \beta C$)
    - AXPY: linear combination of 2 Vectors ( )
    - NRM2: Norm of a vector, etc. ( )

```
for(i=0; i<N; i++)
 for(j=0; j<N; j++) {
  t = 0.0;
  for(k=0; k<N; k++)
     t += a[i][k]*b[k][j];
  c[i][j] = t;
 }
```



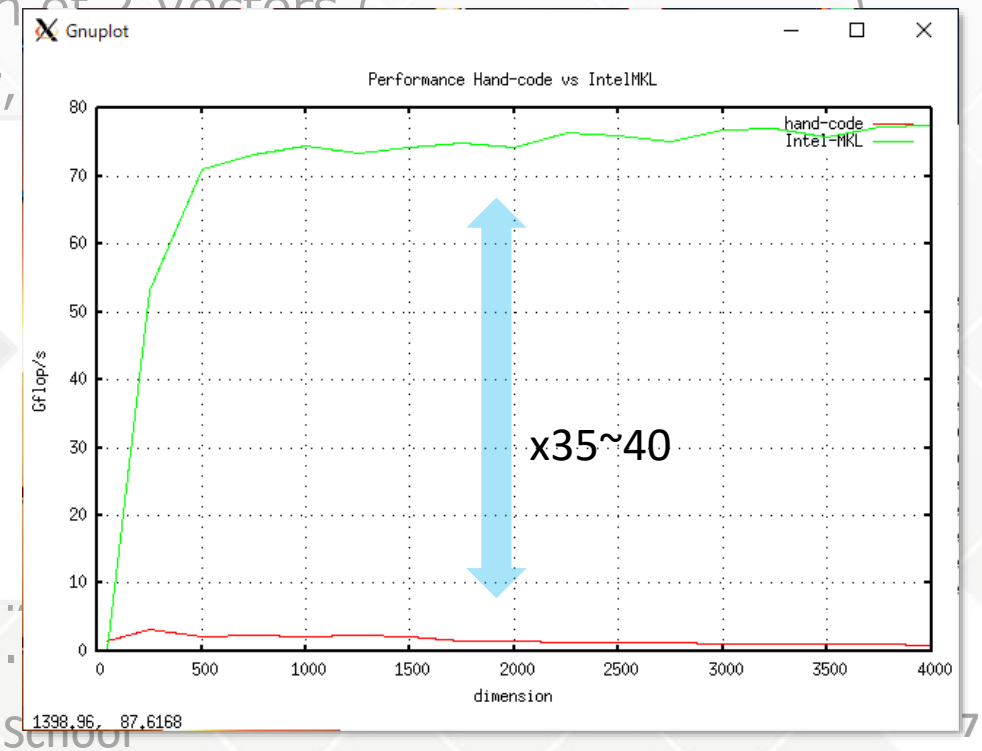
```
cblas_dgemm( CblasRowMajor,
             CblasNoTrans, CblasNoTrans,
             N, N, N,
             1.0, a, N, b, N, 0.0, c, N);
```

  - NVIDIA CUBLAS, AMD clMATH, MAGMABLAS(@UTK), KBLAS(@KAUST), ASPEN.K2(@RIKEN) : for GPGPU

Example

- **When you HOPE to SPEED UP your code bottlenecked in matmul, use (or link) an appropriate BLAS (Basic Linear Algebra Subprograms) library!**
  - Standard API for linear algebra kernels (case of (C)BLAS).
    - GEMM : Matrix-matrix multiplication ($C := \alpha AB + \beta C$)
    - AXPY: linear combination of 2 Vectors ( )
    - NRM2: Norm of a vector,

```
for(i=0; i<N; i++)
 for(j=0; j<N; j++) {
  t = 0.0;
  for(k=0; k<N; k++)
     t += a[i][k]*b[k][j];
  c[i][j] = t;
 }
```



- KBLAS(@KAUST), ASPEN.

Example

- **When you HOPE to SPEED UP your code bottlenecked in matmul, use (or link) an appropriate BLAS (Basic Linear Algebra Subprograms) library!**

  - Standard API for linear algebra kernels.
    - GEMM : Matrix-matrix multiplication ($C := \alpha AB + \beta C$)
    - AXPY: linear combination of 2 Vectors ($y := \alpha x + y$)
    - NRM2: Norm of a vector, etc. ($a = \|x\|_2$)

    Reference codes are available from netlib@UTK.

    http://www.netlib.org/BLAS/

  - Commercial: Intel MKL, AMD ACML (free)
  - Open Source: ATLAS(@UTK), GotoBLAS(@TACC), OpenBLAS for general purposed microprocessors
  - nVIDIA CUBLAS, AMD clMATH, MAGMABLAS(@UTK), KBLAS(@KAUST), ASPEN.K2(@RIKEN) : for GPGPU

Other cases

Computer simulations create the future

Suggestion: for more complex problems, use followings;

- LAPACK (http://www.netlib.org/lapack/)
- ScaLAPACK (http://www.netlib.org/scalapack/)
- Elemental (http://libelemental.org/)

Dense, General

- EigenExa (http://www.aics.riken.jp/labs/lpnctrt/EigenExa_e.html)
- ELPA (http://elpa.rzg.mpg.de/)

Dense Eigenvalue

- PETSc (http://www.mcs.anl.gov/petsc/)
- Trillions (https://trilinos.org/)

Sparse, General

- ARPACK (http://www.caam.rice.edu/software/ARPACK/)

Sparse, Eigenvalue

- FFTW (http://www.fftw.org/)
- FFTE (http://www.ffte.jp/)
- 2decomp&FFT (http://www.2decomp.org/)

FFT

- MT, MTGP, dSFMT (http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index.html)

Random number

- GMP big number librart(https://gmplib.org/)
- QD pack, MPACK, and so on

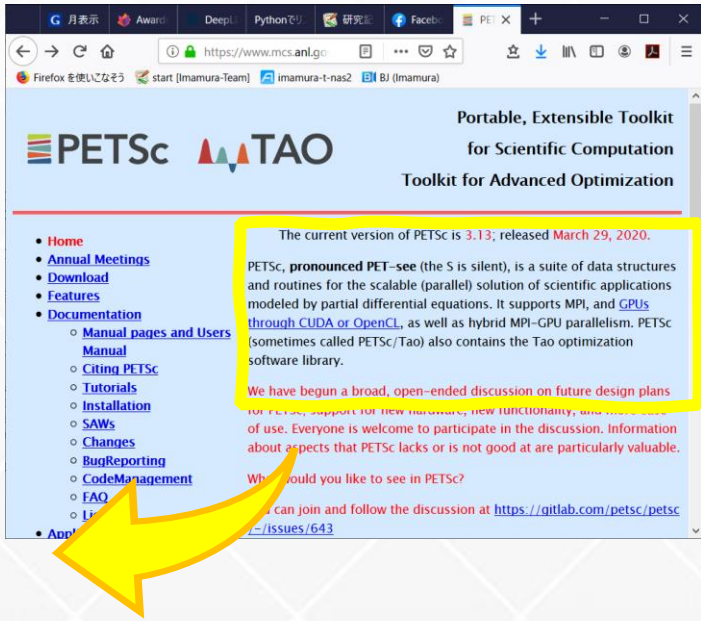
Multi-precision number

General use of PETSc(SLEPC)

PETSc/TAO

- **Developed by Argonne National Lab. USA**
  - Portable, Extensible Toolkit for Scientific Computation
  - Toolkit for Advanced Optimization

https://www.mcs.anl.gov/petsc/



PETSc, pronounced PET-see (the S is silent), is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It supports MPI, and GPUs through CUDA or OpenCL, as well as hybrid MPI-GPU parallelism. PETSc (sometimes called PETSc/Tao) also contains the Tao optimization software library.

(cf. PETSc/TAO homepage)

Simple example (ex1.c)

$$A = \begin{pmatrix} -2 & 1 & & \\ 1 & -2 & & \\ & & \ddots & 1 \\ & & 1 & -2 \end{pmatrix}, b = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

- **Solve the linear system $Ax = b$.**

```
$ make ex1
$ pjsub job_ex.sh
```

Simple example (C/ex1.c)

```
Vec         x, b, u;     /* approx solution, RHS, exact solution */
Mat          A;          /* linear system matrix */
KSP         ksp;         /* linear solver context */
PC          pc;          /* preconditioner context */
PetscReal     norm;      /* norm of solution error */

VecCreate(PETSC_COMM_WORLD,&x);       //Define x
VecSetSizes(x,PETSC_DECIDE,n); // Set size of x
VecDuplicate(x,&b);
VecDuplicate(x,&u);

MatCreate(PETSC_COMM_WORLD,&A); // Define A
MatSetSizes(A,PETSC_DECIDE,PETSC_DECIDE,n,n); // Specify size of A
MatSetFromOptions(A);  // Reflects -mat_type option (default is AIJ format)
MatSetUp(A);
```

Simple example (C/ex1.c)

```
value[0] = -1.0; value[1] = 2.0; value[2] = -1.0;
for (i=1; i<n-1; i++) {
  col[0] = i-1; col[1] = i; col[2] = i+1;
  MatSetValues(A,1,&i,3,col,value,INSERT_VALUES);
}
i    = n - 1; col[0] = n - 2; col[1] = n - 1;
MatSetValues(A,1,&i,2,col,value,INSERT_VALUES);
i    = 0; col[0] = 0; col[1] = 1; value[0] = 2.0; value[1] = -1.0;
MatSetValues(A,1,&i,2,col,value,INSERT_VALUES);
MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);
MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);
```

Simple example (F/ex1f.F90)

```
   call MatCreate(PETSC_COMM_WORLD,A,ierr)
   call MatSetSizes(A,PETSC_DECIDE,PETSC_DECIDE,n,n,ierr)
   call MatSetFromOptions(A,ierr)
   call MatSetUp(A,ierr)

   value(1) = -1.0
   value(2) = 2.0
   value(3) = -1.0
   do 50 i=1,n-2
     col(1) = i-1
     col(2) = i
     col(3) = i+1
     call MatSetValues(A,i1,i,i3,col,value,INSERT_VALUES,ierr)
50  continue
   i = n - 1
   col(1) = n - 2
   col(2) = n - 1
   call MatSetValues(A,i1,i,i2,col,value,INSERT_VALUES,ierr)
   i = 0
   col(1) = 0
   col(2) = 1
   value(1) = 2.0
   value(2) = -1.0
   call MatSetValues(A,i1,i,i2,col,value,INSERT_VALUES,ierr)
   call MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY,ierr)
   call MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY,ierr)
```

Conputed result

```
KSP Object: 1 MPI processes
  type: gmres
    restart=30, using Classical (unmodified) Gram-Schmidt Orthogonalization with no
iterative refinement
    happy breakdown tolerance 1e-30
  maximum iterations=10000, initial guess is zero
  tolerances:  relative=1e-05, absolute=1e-50, divergence=10000.
  left preconditioning
  using PRECONDITIONED norm type for convergence test
PC Object: 1 MPI processes
  type: jacobi
  type DIAGONAL
  linear system matrix = precond matrix:
  Mat Object: 1 MPI processes
    type: seqaij
    rows=100, cols=100
    total: nonzeros=298, allocated nonzeros=500
    total number of mallocs used during MatSetValues calls=0
      not using I-node routines
Norm of error 0.0114852, Iterations 318
```

Conputed result

```
mpiexec ./ex1 -n 100 -ksp_type cg -pc_type none
```

```
KSP Object: 1 MPI processes
  type: cg
  maximum iterations=10000, initial guess is zero
  tolerances:  relative=1e-05, absolute=1e-50, divergence=10000.
  left preconditioning
  using PRECONDITIONED norm type for convergence test
PC Object: 1 MPI processes
  type: none
  linear system matrix = precond matrix:
  Mat Object: 1 MPI processes
    type: seqaij
    rows=100, cols=100
    total: nonzeros=298, allocated nonzeros=500
    total number of mallocs used during MatSetValues calls=0
      not using I-node routines
Norm of error 1.85656e-14, Iterations 50
```

Hands-on time

Please access /vol0001/ra020019/1D-PETSc

```
$ cd /vol0400/data/ra020019/<YID>_data/
$ cp /col0001/ra020019/1D-PETSc .
$ cd 1D-PETSc
$ ls
1d2-petsc.c  1d2-petsc.f90  Makefile  input.dat  job.sh
```

Compile and link

Makefile

```
PETSC_LIB  = -L/vol0001/ra020019/PETSc/petsc-3.17.2/lib -lpetsc
PETSC_LIB += -L/vol0001/ra020019/PETSc/lib -lHYPRE
PETSC_INC  = -I/vol0001/ra020019/PETSc/petsc-3.17.2/include
PETSC_INC += -I/vol0001/ra020019/PETSc/petsc-3.17.2/include/petsc
PETSC_DIR = /vol0001/ra020019/PETSc/petsc-3.17.2

programs=1d2-petsc
all: $(programs)

.c.o:
        $(CC) -c  $<  -I./ $(PETSC_INC)

1d2-petsc: 1d2-petsc.o
        $(CC) -o 1d2-petsc 1d2-petsc.o $(PETSC_LIB) -lpetsc -lm -lblas -llapack

clean:
        rm -rf *.o *~ $(programs) *bak
```

Use **make** command

→ → →

eps_exec is generated.

Job submission

```
#!/bin/bash

#PJM -L "node=1“
#PJM -L "rscgrp=small"
#PJM -L "elapse=00:10:00"
#PJM -g ra020019
#PJM --mpi "max-proc-per-node=4"

export PETSC_DIR=/vol0001/ra020019/PETSc/
export
LD_LIBRARY_PATH=$PETSC_DIR/petsc3.17.2/lib:$PETSC_DIR/lib:$LD_LIBRARY_
PATH

setenv MP_STDINMODE all
mpiexec ./a.out < input.dat
```

How to setup a matrix? (in C)

- **PETSc handles internal data format and interface data flexibly. Because of PETSc management mechanism, user does not to see actual state on memory . Matrix A is dealt with a handler variable, and matrix elements are accessed via a query API.**

```
Vec Rhs_PETSC;
VecCreate(PETSC_COMM_WORLD, &Rhs_PETSC);
VecSetSizes(Rhs_PETSC, N, PETSC_DECIDE);
VecSetFromOptions(Rhs_PETSC);
Vec x_PETSC;

Mat AMat_PETSC;
MatCreate(PETSC_COMM_WORLD, &AMat_PETSC);
MatSetType(AMat_PETSC,MATMPIAIJ);
MatSetSizes(AMat_PETSC, PETSC_DECIDE, N, Ng, PETSC_DECIDE);
MatSetUp(AMat_PETSC);
```

Create a vector handler

Create a matrix handler

How to setup a matrix? (in C)

- **PETSc handles internal data format and interface data flexibly. Because of PETSc management mechanism, user does not to see actual state on memory . Matrix A is dealt with a handler variable, and matrix elements are accessed via a query API.**

```
Vec Rhs_PETSC;
VecCreate(PETSC_COMM_WORLD, &Rhs_PETSC);
VecSetSizes(Rhs_PETSC, N, PETSC_DECIDE);
VecSetFromOptions(Rhs_PETSC);
Vec x_PETSC;

Mat AMat_PETSC;
MatCreate(PETSC_COMM_WORLD, &AMat_PETSC);
MatSetType(AMat_PETSC,MATMPIAIJ);
MatSetSizes(AMat_PETSC, PETSC_DECIDE, N, Ng, PETSC_DECIDE);
MatSetUp(AMat_PETSC);
```

Define the matrix type

How to setup a matrix? (in C)

Computer simulations create the future

- **PETSc handles internal data format and interface data flexibly. Because of PETSc management mechanism, user does not to see actual state on memory . Matrix A is dealt with a handler variable, and matrix elements are accessed via a query API.**

```
Vec Rhs_PETSC;
VecCreate(PETSC_COMM_WORLD, &Rhs_PETSC);
VecSetSizes(Rhs_PETSC, N, PETSC_DECIDE);
VecSetFromOptions(Rhs_PETSC);
Vec x_PETSC;

Mat AMat_PETSC;
MatCreate(PETSC_COMM_WORLD, &AMat_PETSC);
MatSetType(AMat_PETSC,MATMPIAIJ);
MatSetSizes(AMat_PETSC, PETSC_DECIDE, N, Ng, PETSC_DECIDE);
MatSetUp(AMat_PETSC);
```

Vector size

Matrix size

How to setup a matrix? (in C)

- **PETSc handles internal data format and interface data flexibly. Because of PETSc management mechanism, user does not to see actual state on memory . Matrix A is dealt with a handler variable, and matrix elements are accessed via a query API.**

```
for(i=0;i<N;i++){
  int Row = Base + i;
  MatSetValue(AMat_PETSC, Row, Row, Diag[i], INSERT_VALUES);
  for(j=Index[i];j<Index[i+1];j++){
    int Col = Base + Item[j];
    if(Item[j] >= N){
      Col = Base + N;
      if(MyRank>0 && Item[j] == N){ Col = Base - 1; }}
  MatSetValue(AMat_PETSC, Row, Col, AMat[j], INSERT_VALUES);
}}
```

Set matrix value to PETSc

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

Assemble the matrix data set on a distributed manner

Play with PETSc/SLEPC

- **Change the KSP or PC types**

mpiexec ./1d2-petsc -ksp_type cg -pc_type none

Converged Reason = DIVERGED_ITS
2000 iters, RESID= 4.799971e+04
1.863066e-02 6.551349e-01 sec.

mpiexec ./1d2-petsc -ksp_type cg -pc_type bjacobi

Converged Reason = CONVERGED_ATOL
10 iters, RESID= 7.278355e-09
1.149673e-02 1.180010e-02 sec.

mpiexec ./1d2-petsc -ksp_type cg -pc_type asm

Converged Reason = CONVERGED_ATOL
10 iters, RESID= 7.278355e-09
1.149673e-02 1.180010e-02 sec.

---

mpiexec ./1d2-petsc -ksp_type gmres -pc_type none

Converged Reason = DIVERGED_ITS
2000 iters, RESID= 2.229877e+02
1.618363e-02 1.751933e+00 sec.

mpiexec ./1d2-petsc -ksp_type gmres -pc_type bjacobi

Converged Reason = CONVERGED_ATOL
68 iters, RESID= 4.784548e-15
1.152908e-02 6.925694e-02 sec.

mpiexec ./1d2-petsc -ksp_type gmres -pc_type asm

Converged Reason = CONVERGED_ATOL
40 iters, RESID= 8.775967e-09
1.234917e-02 8.183712e-02 sec.

Hands-on time

Please access /vol0001/ra020019/1D-PETSc

```
$ cd /vol0400/data/ra020019/<YID>_data/
$ cp /col0001/ra020019/1D-PETSc .
$ cd 1D-PETSc/C
$ ls
1d2-petsc.c  ex1.c   Makefile
input.dat    job.sh  job_ex.sh
```

- **KSP tpes**

Solver	KSPType	Options Database Name
Richardson	KSPRICHARDSON	richardson
Chebychev	KSPCHEBYCHEV	chebychev
Conjugate Gradient	KSPCG	cg
BiConjugate Gradient	KSPBICG	bicg
Generalized Minimal Residual	KSPGMRES	gmres
BiCGSTAB	KSPBCGS	bcgs
Conjugate Gradient Squared	KSPCGS	cgs

- **There are other types.**
- **See user manual for details**

- **PC tpes**

SPrecondition	PCType	Options Database Name
Jacobi	PCJACOBI	jacobi
Block Jacob	PCBJACOBI	bjacobi
SOR (and SSOR)	PCSOR	sor
Incomplete Cholesky	PCICC	icc
Incomplete LU	PCILU	ilu
Additive Schwarz	PCASM	asm
No preconditioning	PCNONE	none

- **There are other types.**
- **See user manual for details**
- **Some combinations cannot be calculated.**