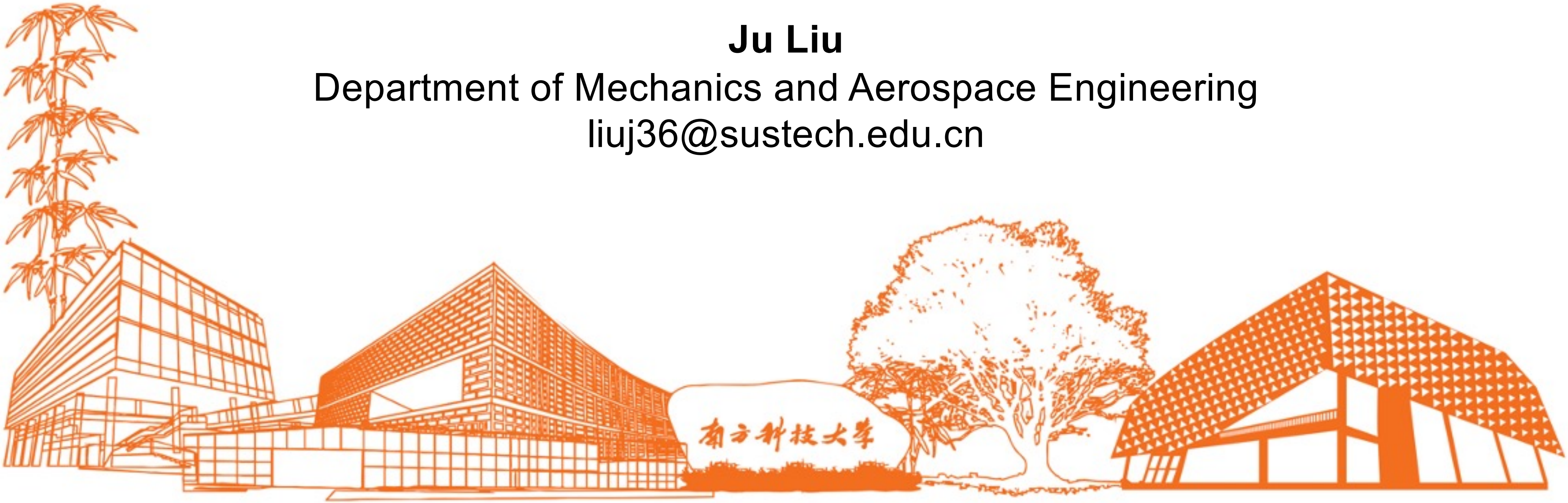
MAE 5032 High Performance Computing: Methods and Applications

Lab 9: PETSc

Ju Liu
Department of Mechanics and Aerospace Engineering
liuj36@sustech.edu.cn



Objective

- understand the basic function usage of PETSc
- link an application code to PETSc on Taiyi cluster
- tune the code with different solver options through command line arguments

Physical background

- The Poisson equation $-\Delta u = f$ characterizes the state of temperature distribution or electrical potential or pressure in fluid flow, etc.

- We consider it in 1D and, in particular, in the unit inverval (0, 1)

$$-\frac{d^2u}{dx^2} = f(x)$$

- We consider homogeneous boundary conditions:

$$u(0) = u(1) = 0$$

Physical background

- Suppose the domain is discretized into uniform grids with gird size $h = 1/N$, we may approximate the differential equation as

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f_i$$

- After re-organization, we have a matrix problem with the left-hand size matrix in the tri-diagonal format

$$A = \begin{bmatrix} 2 & -1 & 0 & & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & 2 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2 \end{bmatrix}.$$

Code

- The matrix is setup with function MatSetUp, which will automatically allocate the memory layout for the matrix using default options.

```
// Create matrix.
// We pass in nlocal as the "local" size of the matrix to force it
// to have the same parallel layout as the vector created above.
MatCreate(PETSC_COMM_WORLD,&A);
MatSetSizes(A, nlocal, nlocal, n, n);
MatSetFromOptions(A);
MatSetUp(A);
```

- Use –info to see the effect of matrix allocation

<mat:seqaij> MatAssemblyEnd_SeqAIJ(): Number of mallocs during MatSetValues() is 0

means there is no extra memory allocation.

Code

- The matrix is setup with function MatSetUp, which will automatically allocate the memory layout for the matrix using default options.

- Use `MatMPIAIJSetPreallocation` to manually preallocate the matrix

```
MatCreate(PETSC_COMM_WORLD,&A);
MatSetSizes(A, nlocal, nlocal, n, n);
MatSetType(A, MATMPIAIJ);
MatMPIAIJSetPreallocation(A, 2, PETSC_NULLPTR, 0, PETSC_NULLPTR);
MatSetOption(A, MAT_NEW_NONZERO_ALLOCATION_ERR, PETSC_FALSE);
```

Code

- The matrix is assembled locally in each individual process

```
if (rstart == 0)
{
  rstart = 1;
  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);
}

if (rend == n)
{
  rend = n-1;
  i    = n-1; col[0] = n-2; col[1] = n-1; value[0] = -1.0; value[1] = 2.0;
  MatSetValues(A,1,&i,2,col,value,INSERT_VALUES);
}

// Set entries corresponding to the mesh interior
value[0] = -1.0; value[1] = 2.0; value[2] = -1.0;
for (i=rstart; i<rend; i++)
{
  col[0] = i-1; col[1] = i; col[2] = i+1;
  MatSetValues(A,1,&i,3,col,value,INSERT_VALUES);
}

// Assemble the matrix
MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);
MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);
```

Code

- In the ex4.c code, the right-hand side vector b is generated by a manufactured solution (all-1 vector).

- The correctness of the ksp solver is assessed by comparing the solution against the manufactured solution

- We print the norm of the error and the number of iterations on screen.

Code

number of processes

```
#!/bin/bash

~/lib/petsc-3.22.2-debug/bin/mpiexec -np 5 ./ex4.out \
   -n 10000 \
   -ksp_type gmres \
   -ksp_gmres_restart 30 \
   -ksp_rtol 1.0e-10 \
   -ksp_atol 1.0e-50 \
   -ksp_max_it 1500 \
   -pc_type none \
   -ksp_monitor_short \
   -ksp_converged_reason \
   -ksp_view
```

grid size

ksp method

ksp stopping criteria

preconditioner type

print extra info on screen

Code

- Test the code either on your own machine or on Taiyi

- set -n 10000

- run in serial with
  - –ksp_type preonly –pc_type lu
  - –ksp_type preonly –pc_type cholesky
  - –ksp_type richardson –pc_type jacobi
  - –ksp_type gmres –pc_type none
  - –ksp_type gmres –pc_type asm
  - –ksp_type cg –pc_type asm
  - –ksp_type cg –pc_type hypre

Code

- PETSc default direct solver does not support parallel solve

- run in parallel using two processes with
  - –ksp_type preonly –pc_type lu
  - –ksp_type preonly –pc_type cholesky
  - –ksp_type richardson –pc_type jacobi
  - –ksp_type gmres –pc_type none
  - –ksp_type gmres –pc_type asm
  - –ksp_type cg –pc_type asm
  - –ksp_type cg –pc_type hypre

Code

- In the ex5.c code, the right-hand side vector b is generated by a manufactured solution $u = \sin(\pi x)$.

- The solution vector *x* is an approximation of the exact solution *u*.

- We monitor the error $e_i = x_i - u_i$ in two different norms

$$e_2 := \left(h \sum_i e_i^2\right)^{1/2}$$

$$e_\infty := \max(e_i)$$