Computer simulations create the future

PETSc for Parallel FEM

RIKEN CCS HPC Summer School
Toshiyuki Imamura

Sample files

Please access /work/gt36/share/1D-PETSc

```
$ cd /work/gt36/<YID>/
$ cp -r /work/gt36/share/1D-PETSc .
$ cd 1D-PETSc
$ ls
1d2-petsc.c    ex1.c    input.dat  run_1d2.sh
1d2-petscf.F90 ex1f.F90 Makefile   run_ex1.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

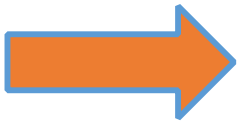
Numerical Library

- **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;
  }
```



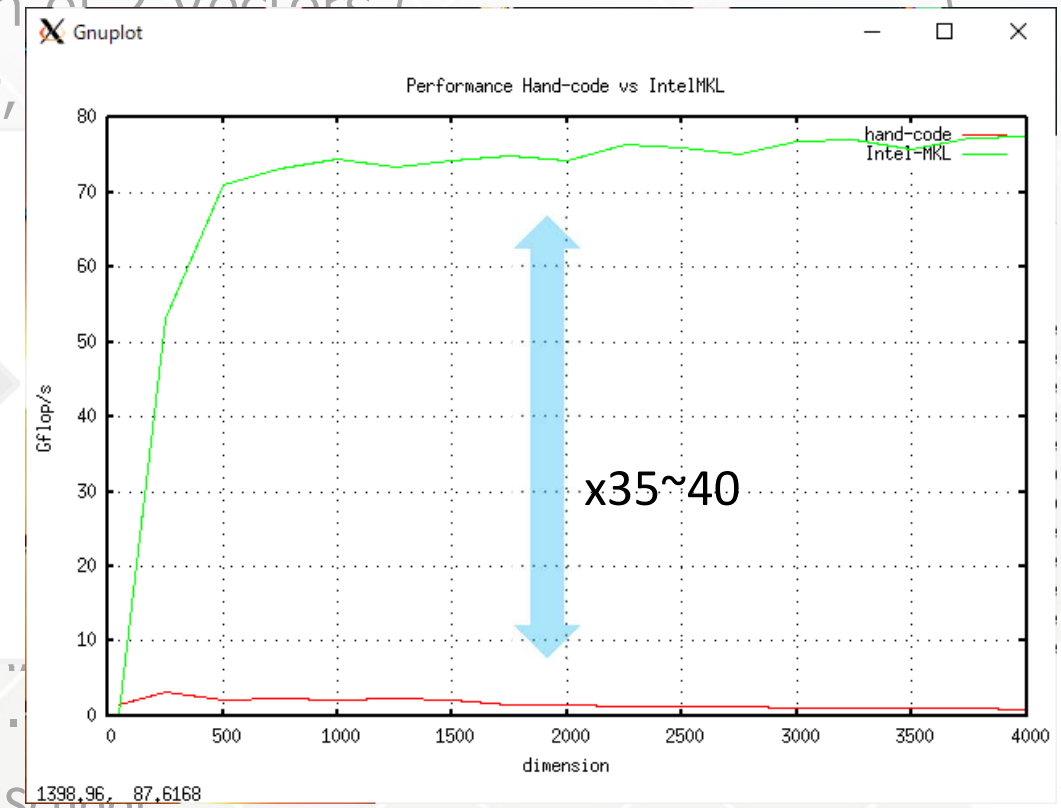
```
cblas_dgemm( CblasRowMajor,
             CblasNoTrans, CblasNoTrans,
             N, N, N,
             1.0, a, N, b, N, 0.0, c, N);
```

    - 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/)

Dense, General

- Elemental (http://libelemental.org/)
- 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/)

FFT

- 2decomp&FFT (http://www.2decomp.org/)
- 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

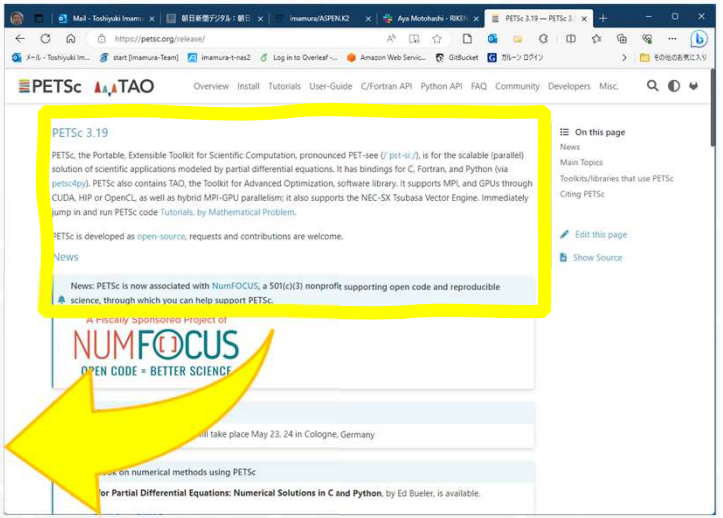
General use of PETSc

PETSc/TAO

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

https://petsc.org/

PETSc, the Portable, Extensible Toolkit for Scientific Computation, pronounced PET-see (/ˈpɛt-siː/), is for the scalable (parallel) solution of scientific applications modeled by partial differential equations. It has bindings for C, Fortran, and Python (via petsc4py). PETSc also contains TAO, the Toolkit for Advanced Optimization, software library. It supports MPI, and GPUs through CUDA, HIP or OpenCL, as well as hybrid MPI-GPU parallelism; it also supports the NEC-SX Tsubasa Vector Engine. Immediately jump in and run PETSc code.



(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 $Ax = b$.
- PETSc handles the internal data format and interface data; due to PETSc's management mechanism, the user never sees the actual state in memory. Matrix A is handled by handler variables, and matrix elements are accessed by the query API.

Preparation

- **Module load is necessary to access Odyssey and use some software packages**
- **Here, at least basic packages for [odyssey] is significant.**

```
% module purge
% module list
No Modulefiles Currently Loaded.
% module load odyssey
Loading odyssey
  Loading requirement: fjmpi/1.2.37 fj/1.2.37
% module list
Currently Loaded Modulefiles:
 1) fjmpi/1.2.37   2) fj/1.2.37(default)   3) odyssey
```

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 /work/gt36/share/1D-PETSc

```
$ cd /work/gt36/<YID>/
$ cp -r /work/gt36/share/1D-PETSc .
$ cd 1D-PETSc
$ ls
1d2-petsc.c    ex1.c    input.dat  run_1d2.sh
1d2-petscf.F90 ex1f.F90 Makefile   run_ex1.sh
```

Compile and link

Computer simulations create the future

Makefile

```
#
# The minimal example of Makefile for compiling PETSc executables
# By Toshiyuki Imamura (imamura.toshiyuki@riken.jp)
#

PETSC_DIR = /work/gt36/share/petsc/3.15.0
PETSC_LIB = $(PETSC_DIR)/lib
PETSC_INC = $(PETSC_DIR)/include
HYPRE_DIR = /work/gt36/share/hypre/2.20.0
HYPRE_LIB = $(HYPRE_DIR)/lib
HYPRE_INC = $(HYPRE_DIR)/include

CC       = mpifccpx -Nclang -Kopenmp
FC       = mpifrtpx -Nclang -Kopenmp
INCFLAGS   = -I$(PETSC_INC) -I$(HYPRE_INC)
LDFLAGS    = -Wl,-rpath,$(PETSC_LIB) -L$(PETSC_LIB) -lpetsc
LDFLAGS   := $(LDFLAGS) -Wl,-rpath,$(HYPRE_LIB) -L$(HYPRE_LIB) -lHYPRE
LDFLAGS   := $(LDFLAGS) -lfjprofmpif -lfjprofmpi -lfjcrt -SCALAPACK -SSL2BLAMP

programs=1d2-petsc ex1 1d2-petscf ex1f

all: $(programs)

.SUFFIXES: .o .c .F90
.c.o:
	$(CC) -c  $<  -I./ -I$(PETSC_INC)
.F90.o:
	$(FC) -c  $<  -cpp -I./ -I$(PETSC_INC)

1d2-petsc: 1d2-petsc.o
	$(CC) -o 1d2-petsc 1d2-petsc.o $(INCFLAGS) $(LDFLAGS)
1d2-petscf: 1d2-petscf.o
	$(FC) -o 1d2-petscf 1d2-petscf.o $(INCFLAGS) $(LDFLAGS)

ex1: ex1.o
	$(CC) -o ex1 ex1.o $(INCFLAGS) $(LDFLAGS)
ex1f: ex1f.o
	$(FC) -o ex1f ex1f.o $(INCFLAGS) $(LDFLAGS)

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

Job submission

```
#!/bin/bash

#PJM -L node=1
#PJM -L rscgrp=lecture-o
#PJM -L elapse=00:10:00
#PJM -g gt36
#PJM --mpi proc=8

module load odyssey

# C
mpiexec -np 1 ./1d2-petsc
#mpiexec -np 8 ./1d2-petsc -ksp_type cg -pc_type none
#mpiexec -np 8 ./1d2-petsc -ksp_type cg -pc_type bjacobi
#mpiexec -np 8 ./1d2-petsc -ksp_type cg -pc_type asm
#mpiexec -np 8 ./1d2-petsc -ksp_type gmres -pc_type none
#mpiexec -np 8 ./1d2-petsc -ksp_type gmres -pc_type bjacobi
#mpiexec -np 8 ./1d2-petsc -ksp_type gmres -pc_type asm

# Fortran
#mpiexec -np 1 ./1d2-petscf
```

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)

- **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

- **Change the KSP or PC types**

mpirun –np 8 ./1d2-petsc -ksp_type cg -pc_type none

Converged Reason = DIVERGED_ITS
2000 iters, RESID= 4.599971e+04
5.002301e-02 9.699515e-01 sec.

mpirun –np 8 ./1d2-petsc -ksp_type cg -pc_type bjacobi

Converged Reason = CONVERGED_ATOL
164 iters, RESID= 1.759163e-09
2.389900e-02 2.276256e-01 sec.

mpirun –np 8 ./1d2-petsc -ksp_type cg -pc_type asm

Converged Reason = CONVERGED_ATOL
21 iters, RESID= 1.420679e-09
2.384981e-02 4.237503e-02 sec.

---

mpirun –np 8 ./1d2-petsc -ksp_type gmres -pc_type none

Converged Reason = DIVERGED_ITS
2000 iters, RESID= 2.184571e+02
2.386021e-02 1.957389e+00 sec.

mpirun –np 8 ./1d2-petsc -ksp_type gmres -pc_type bjacobi

Converged Reason = DIVERGED_BREAKDOWN
30 iters, RESID= 1.985900e+02
2.430236e-02 6.757502e-02 sec.

mpirun –np 8 ./1d2-petsc -ksp_type gmres -pc_type asm

Converged Reason = CONVERGED_ATOL
137 iters, RESID= 2.002330e-10
2.392390e-02 2.671833e-01 sec.

Sample files

Please access /work/gt36/share/1D-PETSc

```
$ cd /work/gt36/<YID>/
$ cp -r /work/gt36/share/1D-PETSc .
$ cd 1D-PETSc
$ ls
1d2-petsc.c    ex1.c    input.dat  run_1d2.sh
1d2-petscf.F90 ex1f.F90 Makefile   run_ex1.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 types**

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.**