IT4Innovations national supercomputing center

PRACE AUTUMN SCHOOL 2016
September 27-30, 2016
Hagenberg, Austria

PRACE Autumn School 2016
PETSc tutorial

Part VI: Linear System Solvers

Václav Hapla
IT4Innovations
VSB– Technical University of Ostrava
Ostrava, Czech Republic

Interlude

- implicit matrices
- `-help`
- `-log_summary`
- look at `makefile`

Linear system solvers (KSP)

```
KSP ksp;
KSPType type = KSPCG;
MPI_Comm comm = PETSC_COMM_WORLD;
```

- **create**: `KSPCreate(comm, &ksp);`
- **type**: `KSPSetType(ksp, type);  // default=KSPGMRES`
- **options:** `KSPSetFromOptions(ksp);`
- **dealloc**: `KSPDestroy(&ksp);`

Solving linear system

```
Mat A;  Vec b,x;
// initialize A,b,x
KSPSetOperators(ksp, A, A);
KSPSolve(ksp,b,x);
```

- The first matrix **defines** the linear **system**
- The second matrix is used in **constructing** the **preconditioner**

Convergence tolerances

```
KSPSetTolerances(ksp, rtol, atol, dtol, maxit);
```

- `rtol` = the **relative** convergence tolerance
  stop if `residual < rtol * norm(RHS)`
- `atol` = **absolute** convergence tolerance
  stop if `resisual < atol`
- `dtol` = **divergence** tolerance
  stop if `residual > dtol * norm(RHS)`
- `maxit` = **maximum** number of **iterations**
- all can be set to `PETSC_DEFAULT`
- **options database**: `-ksp_rtol 1e-6 -ksp_divtol 1e5 -ksp_atol 1e-12 -ksp_max_it 2000`

Preconditioners

- many built-in and interfaced preconditioners
- ILU, block Jacobi, sparse aproximate inverse ...
- can be composed (`PCCOMPOSITE`)

```
KSP ksp;
PC pc;
PCType pctype=PCILU;
...
KSPGetPC(ksp,&pc);
PCSetType(pc,pctype);
```

Try various linear system solvers

```
KSPType ksptype = KSPCG;
KSP ksp;
...
KSPSetType(ksp,ksptype);
```

- `type = {KSPRICHARDSON, KSPCG, KSPGCR, KSPGMRES, ...}`
- command-line:
  `-ksp_type {richardson,cg,gcr,gmres,...}`
- see manual pages of `KSPSetType`, `KSPType`, and concrete types

Try various preconditioners

```
PCType pctype = PCILU;
PC pc;
KSP ksp;
...
KSPGetPC(ksp,&pc);
PCSetType(pc,pctype);
```

- `type={PCJACOBI,PCBJACOBI,PCILU,PCICC...}`
- command-line:
  `-pc_type {richardson,cg,gcr,gmres,...}`
- see manual pages of `KSPSetType`, `KSPType`, and concrete types

Direct solvers

- direct solvers = special case of preconditioned iterative solvers
- just one iteration with application of „ultimate preconditioner“, i.e. forward & backward substitution of a complete factor

```
KSPSetType(ksp, KSPPREONLY);
PCSetType(pc, PCLU);     // or PCCHOLESKY
```

Iterative/preconditioned/direct solvers

method	PCType	KSPType
pure iterative	none	cg, gmres, gcr, richardson,...
preconditioned iterative	ilu, icc, jacobi, sor, ...	cg, gmres, gcr, richardson,...
direct	lu, cholesky	preonly

MatSolverPackage

```
PC pc;
MatSolverPackage pkg = MATSOLVERMUMPS;
PCFactorSetMatSolverPackage(pc,pkg);
```

- `pkg={MATSOLVERMUMPS,MATSOLVERUMFPACK,...}`
- command-line:
  `-pc_factor_mat_solver_package {mumps,umfpack,...}`
- see manual pages of `PCFactorSetMatSolverPackage`, `MatSolverPackage`, and concrete packages

Low-level access to factorization

```
IS isr, isc; MatFactorInfo info;
MatGetOrdering(A,MATORDERING_NATURAL,&isr,&isc);
MatLUFactor(A,isr,isc,&info);
  // = MatLUFactorSymbolic + MatLUFactorNumeric
  // LU, Cholesky

MatSolve(A,b,x);
  // Solves A x = b
MatSolves(Mat A,Vecs bs,Vecs xs)
  // Solves A x = b for a collection of vectors
MatMatSolve(Mat A,Mat B,Mat X)
  // Solves A X = B, given a factored matrix
```

Hands-on: ex5

`make ex5; mpirun -n 3 ./ex5`

1. monitor convergence: `-ksp_monitor`
2. view solver details: `-ksp_view`
3. view reason of iteration termination: `-ksp_converged_reason`
   - system matrix is singular
   - iterative solver diverges
   - direct solver detects a zero pivot
4. notice `KSPSetInitialGuessNonzero`
5. enforce Dirichlet boundary conditions using `MatZeroRowsColumns`
6. avoid coefficient jumps when enforcing Dirichlet boundary conditions - see next slide
7. view solution using `-ksp_view_solution`
8. the string is fixed on one side – modify code to get it fixed on both sides (just one number!)
9. try various solvers and preconditioners (`-ksp_type, -pc_type`), e.g. `gmres + jacobi`

See manual pages of `KSPSetFromOptions`, `KSPSolve` and `KSPSetType` for available options.

Dirichlet Boundary Conditions

- enforce Dirichlet boundary conditions for DOFs with indices in the index set `dbcidx`

```
diag = 1.0
for i = dbcidx
  x(i) = 0.0     // or other value of prescribed displacement
  A(i,:) = 0;  A(:,i) = 0;
  A(i,i) = diag;  b(i) = diag*x(i);
end
```

- can be improved in the following way to avoid coefficient jumps:

```
diag = max(abs(diag(A)));
```

- PETSc: use `MatGetDiagonal`, `VecAbs`, `VecMax` to get `diag`,

```
MatZeroRowsColumnsIS(A, dbcidx, diag, x, b);
```

DMDA + hands-on ex6

- Manages data for a structured grid in 1, 2, or 3 dimensions.
- In the global representation of the vector each process stores a non-overlapping rectangular (or slab in 3d) portion of the grid points.
- In the local representation these rectangular regions (slabs) are extended in all directions by a stencil width.
- See Jed Brown's presentation from slide 42.
- Compare `ex6.c` with `ex5.c` using your favourite diff tool.
  (Is it vimdiff? ☺)
- Run both.

Hands-on: ex7

- doing the same as ex6
- but using different API: `KSPSetComputeOperators, KSPSetComputeRHS`
- DMDA takes care of matrix and vector allocation completely
- good pre-stage for SNES
- compare code and run with ex6

Thanks for your attention.