Using PETSc Solvers in PyLith

Matthew Knepley, Brad Aagaard, and Charles Williams

Computation Institute
University of Chicago

Department of Molecular Biology and Physiology
Rush University Medical Center

CDM 2012
Golden, CO June 18–22, 2012

Main Point

We want to enable users to,

assess solver performance,

and optimize solvers
for particular problems.

Main Point

We want to enable users to,

assess solver performance,

and optimize solvers
for particular problems.

Main Point

We want to enable users to,

assess solver performance,

and optimize solvers
for particular problems.

Outline

Controlling PETSc

All of PETSc can be controlled by options

```
-ksp_type gmres
-start_in_debugger
```

All objects can have a prefix, `-velocity_pc_type jacobi`

All PETSc options can be given to PyLith

```
--petsc.ksp_type=gmres
--petsc.start_in_debugger
```

Controlling PETSc

All of PETSc can be controlled by options

`-ksp_type gmres`

`-start_in_debugger`

All objects can have a prefix, `-velocity_pc_type jacobi`

All PETSc options can be given to PyLith

`--petsc.ksp_type=gmres`

`--petsc.start_in_debugger`

Examples

We will illustrate options using

PETSc SNES ex5, located at
`$PETSC_DIR/src/snes/examples/tutorials/ex5.c`

and

PyLith Example hex8, located at
`$PYLITH_DIR/examples/3d/hex8/`

Outline

Nonlinear Systems

I am not going to discuss nonlinear systems today,

however if Newton is failing,

contact

petsc-maint@mcs.anl.gov

Nonlinear Systems

I am not going to discuss nonlinear systems today,

however if Newton is failing,

contact

petsc-maint@mcs.anl.gov

What is a Krylov solver?

A Krylov solver builds a small model of a linear operator A, using a subspace defined by

$$\mathcal{K}(A, r) = \mathrm{span}\{r, Ar, A^2 r, A^3 r, \ldots\}$$

where r is the initial residual.

The small system is sovled directly, and the solution is projected back to the original space.

What is a Krylov solver?

A Krylov solver builds a small model of a linear operator A, using a subspace defined by

$$\mathcal{K}(A, r) = \mathrm{span}\{r, Ar, A^2 r, A^3 r, \ldots\}$$

where r is the initial residual.

The small system is sovled directly, and the solution is projected back to the original space.

What Should I Know About Krylov Solvers?

- They can handle *low-mode* errors
- They need preconditioners
- They do a lot of inner products

What is a Preconditioner?

A preconditioner M changes a linear system,

$$M^{-1}Ax = M^{-1}b$$

so that the effective operator is $M^{-1}A$, which is hopefully better for Krylov methods.

- Preconditioner should be inexpensive
- Preconditioner should accelerate convergence

What is a Preconditioner?

A preconditioner M changes a linear system,

$$M^{-1}Ax = M^{-1}b$$

so that the effective operator is $M^{-1}A$, which is hopefully better for Krylov methods.

- Preconditioner should be inexpensive
- Preconditioner should accelerate convergence

What is a Preconditioner?

A preconditioner M changes a linear system,

$$M^{-1}Ax = M^{-1}b$$

so that the effective operator is $M^{-1}A$, which is hopefully better for Krylov methods.

- Preconditioner should be inexpensive
- Preconditioner should accelerate convergence

Always start with LU

Always, always start with LU:

- No iterative tolerance
- (Almost) no condition number dependence
- Check for accidental singularity

In parallel, you need a 3rd party package

- MUMPS (`--download-mumps`)
- SuperLU (`--download-superlu_dist`)

Always start with LU

Always, always start with LU:

- No iterative tolerance
- (Almost) no condition number dependence
- Check for accidental singularity

In parallel, you need a 3rd party package

- MUMPS (`--download-mumps`)
- SuperLU (`--download-superlu_dist`)

What if LU fails?

LU will fail for

- Singular problems
- Saddle-point problems

For saddles use `PC_FIELDSPLIT`

- Separately solves each field
- Decomposition is automatic in PyLith
- Autodetect with `-pc_fieldsplit_detect_saddle_point`
- Exact with full Schur complement solve

What if LU fails?

LU will fail for

- Singular problems
- Saddle-point problems

For saddles use `PC_FIELDSPLIT`

- Separately solves each field
- Decomposition is automatic in PyLith
- Autodetect with `-pc_fieldsplit_detect_saddle_point`
- Exact with full Schur complement solve

Outline

Outline

3. How do I improve?
   - Look at what you have
   - Back off in steps

What solver did you use?

Use `-snes_view` or `-ksp_view` to output a description of the solver:

```
KSP Object:          (fieldsplit_0_)          1 MPI processes
  type: fgmres
    GMRES: restart=100, using Classical (unmodified) Gram-
      Schmidt Orthogonalization with no iterative refinemen
    GMRES: happy breakdown tolerance 1e-30
  maximum iterations=1, initial guess is zero
  tolerances:  relative=1e-09, absolute=1e-50,
    divergence=10000
  right preconditioning
  has attached null space
  using UNPRECONDITIONED norm type for convergence test
```

What did the convergence look like?

Use `-snes_monitor` and `-ksp_monitor`, or `-log_summary`:

What did the convergence look like?

Use `-snes_monitor` and `-ksp_monitor`, or `-log_summary`:

```
0 SNES Function norm 0.207564
1 SNES Function norm 0.0148968
2 SNES Function norm 0.000113968
3 SNES Function norm 6.9256e-09
4 SNES Function norm < 1.e-11
```

What did the convergence look like?

Use `-snes_monitor` and `-ksp_monitor`, or `-log_summary`:

```
0 KSP Residual norm 1.61409
    Residual norms for mg_levels_1_ solve.
    0 KSP Residual norm 0.213376
    1 KSP Residual norm 0.0192085
  Residual norms for mg_levels_2_ solve.
  0 KSP Residual norm 0.223226
  1 KSP Residual norm 0.0219992
    Residual norms for mg_levels_1_ solve.
    0 KSP Residual norm 0.0248252
    1 KSP Residual norm 0.0153432
  Residual norms for mg_levels_2_ solve.
  0 KSP Residual norm 0.0124024
  1 KSP Residual norm 0.0018736
1 KSP Residual norm 0.02282
```

What did the convergence look like?

Use `-snes_monitor` and `-ksp_monitor`, or `-log_summary`:

```
Event          Count        Time (sec)      Flops       Total
            Max Ratio  Max       Ratio    Max   Ratio Mflop/s
KSPSetUp       12 1.0 3.6259e-03 1.0 0.00e+00 0.0      0
KSPSolve        3 1.0 4.8937e-01 1.0 8.93e+05 1.0      2
SNESSolve       1 1.0 4.9477e-01 1.0 9.22e+05 1.0      2
```

Look at timing

Use `-log_summary`:

Event	Time (sec) Max	Ratio	Flops Max	Ratio	--- Global --- %T	%f	%M	%L	%R	Total MF/s
VecMDot	1.8904e-03	1.0	2.49e+04	1.0	0	3	0	0	0	13
MatMult	2.1026e-03	1.0	2.65e+05	1.0	0	29	0	0	0	126
PCApply	4.6001e-01	1.0	7.78e+05	1.0	58	84	0	0	64	2
KSPSetUp	3.6259e-03	1.0	0.00e+00	0.0	0	0	0	0	4	0
KSPSolve	4.8937e-01	1.0	8.93e+05	1.0	61	97	0	0	90	2
SNESSolve	4.9477e-01	1.0	9.22e+05	1.0	62	100	0	0	92	2

Use `-log_summary_python` to get this information as a Python module

Look at timing

Use `-log_summary`:

Event	Time (sec)		Flops		--- Global ---					Total
	Max	Ratio	Max	Ratio	%T	%f	%M	%L	%R	MF/s
VecMDot	1.8904e-03	1.0	2.49e+04	1.0	0	3	0	0	0	13
MatMult	2.1026e-03	1.0	2.65e+05	1.0	0	29	0	0	0	126
PCApply	4.6001e-01	1.0	7.78e+05	1.0	58	84	0	0	64	2
KSPSetUp	3.6259e-03	1.0	0.00e+00	0.0	0	0	0	0	4	0
KSPSolve	4.8937e-01	1.0	8.93e+05	1.0	61	97	0	0	90	2
SNESSolve	4.9477e-01	1.0	9.22e+05	1.0	62	100	0	0	92	2

Use `-log_summary_python` to get this information as a Python module

Outline

3. How do I improve?
   - Look at what you have
   - Back off in steps

Weaken the KSP

GMRES $\Longrightarrow$ BiCGStab

- `-ksp_type bcgs`
- Less storage
- Fewer dot products (less work)
- Variants `-ksp_type bcgsl` and `-ksp_type ibcgs`

Complete Table of Solvers and Preconditioners

Weaken the PC

LU $\Longrightarrow$ ILU

- `-pc_type ilu`
- Less storage and work

In parallel,

- Hypre `-pc_type hypre -pc_hypre_type euclid`
- Block-Jacobi `-pc_type bjacobi -sub_pc_type ilu`
- Additive Schwarz `-pc_type asm -sub_pc_type ilu`

Default for MG smoother is Chebychev/SOR(2)

Weaken the PC

LU $\Longrightarrow$ ILU

- `-pc_type ilu`
- Less storage and work

In parallel,

- Hypre `-pc_type hypre -pc_hypre_type euclid`
- Block-Jacobi `-pc_type bjacobi -sub_pc_type ilu`
- Additive Schwarz `-pc_type asm -sub_pc_type ilu`

Default for MG smoother is Chebychev/SOR(2)

Weaken the PC

LU $\Longrightarrow$ ILU

- `-pc_type ilu`
- Less storage and work

In parallel,

- Hypre `-pc_type hypre -pc_hypre_type euclid`
- Block-Jacobi `-pc_type bjacobi -sub_pc_type ilu`
- Additive Schwarz `-pc_type asm -sub_pc_type ilu`

Default for MG smoother is Chebychev/SOR(2)

Algebraic Multigrid (AMG)

- Can solve elliptic problems
  - Laplace, elasticity, Stokes
- Works for unstructured meshes
- `-pc_type gamg`, `-pc_type ml`, `-pc_type hypre -pc_hypre_type boomeramg`
- CRUCIAL to have an accurate near-null space
  - MatSetNearNullSpace()
  - PyLith provides this automatically
- Use `-pc_mg_log` to put timing in its own log stage

PC_FieldSplit

- Separate solves for block operators
  - Physical insight for subsystems
  - Have optimial PCs for simpler equations
  - Suboptions `fs_fieldsplit_0_*`
- Flexibly combine subsolves
  - Jacobi: `fs_pc_fieldsplit_type = additive`
  - Gauss-Siedel: `fs_pc_fieldsplit_type = multiplicative`
  - Schur complement: `fs_pc_fieldsplit_type = schur`

Stokes example

The common block preconditioners for Stokes require only options:

The Stokes System

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix}$$

```
-pc_type fieldsplit
-pc_field_split_type

-fieldsplit_0_ksp_type preonly
```

Stokes example

The common block preconditioners for Stokes require only options:

```
-pc_type fieldsplit
-pc_field_split_type additive
-fieldsplit_0_pc_type ml
-fieldsplit_0_ksp_type preonly
-fieldsplit_1_pc_type jacobi
-fieldsplit_1_ksp_type preonly
```

PC

$$\begin{pmatrix} \hat{A} & 0 \\ 0 & I \end{pmatrix}$$

Cohouet and Chabard, Some fast 3D finite element solvers for the generalized Stokes problem, 1988.

Stokes example

The common block preconditioners for Stokes require only options:

```
-pc_type fieldsplit
-pc_field_split_type
multiplicative
-fieldsplit_0_pc_type hypre
-fieldsplit_0_ksp_type preonly
-fieldsplit_1_pc_type jacobi
-fieldsplit_1_ksp_type preonly
```

PC

$$\begin{pmatrix} \hat{A} & B \\ 0 & I \end{pmatrix}$$

Elman, Multigrid and Krylov subspace methods for the discrete Stokes equations, 1994.

Stokes example

The common block preconditioners for Stokes require only options:

```
-pc_type fieldsplit
-pc_field_split_type schur
-fieldsplit_0_pc_type gamg
-fieldsplit_0_ksp_type preonly
-fieldsplit_1_pc_type none
-fieldsplit_1_ksp_type minres
```

PC

$$\begin{pmatrix} \hat{A} & 0 \\ 0 & -\hat{S} \end{pmatrix}$$

```
-pc_fieldsplit_schur_factorization_type diag
```

May and Moresi, Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics, 2007.

Olshanskii, Peters, and Reusken, Uniform preconditioners for a parameter dependent saddle point problem with application to generalized Stokes interface equations, 2006.

Stokes example

The common block preconditioners for Stokes require only options:

```
-pc_type fieldsplit
-pc_field_split_type schur

-fieldsplit_0_pc_type gamg
-fieldsplit_0_ksp_type preonly

-fieldsplit_1_pc_type none
-fieldsplit_1_ksp_type minres

-pc_fieldsplit_schur_factorization_type lower
```

PC

$$\begin{pmatrix} \hat{A} & 0 \\ B^T & \hat{S} \end{pmatrix}$$

May and Moresi, Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics, 2007.

Stokes example

The common block preconditioners for Stokes require only options:

```
-pc_type fieldsplit
-pc_field_split_type schur
-fieldsplit_0_pc_type gamg
-fieldsplit_0_ksp_type preonly
-fieldsplit_1_pc_type none
-fieldsplit_1_ksp_type minres
-pc_fieldsplit_schur_factorization_type upper
```

PC

$$\begin{pmatrix} \hat{A} & B \\ 0 & \hat{S} \end{pmatrix}$$

May and Moresi, Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics, 2007.

Stokes example

The common block preconditioners for Stokes require only options:

```
-pc_type fieldsplit
-pc_field_split_type schur

-fieldsplit_0_pc_type gamg
-fieldsplit_0_ksp_type preonly

-fieldsplit_1_pc_type lsc
-fieldsplit_1_ksp_type minres

-pc_fieldsplit_schur_factorization_type upper
```

PC

$$\begin{pmatrix} \hat{A} & B \\ 0 & \hat{S}_{\mathrm{LSC}} \end{pmatrix}$$

May and Moresi, Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics, 2007.

Kay, Loghin and Wathen, A Preconditioner for the Steady-State N-S Equations, 2002.
Elman, Howle, Shadid, Shuttleworth, and Tuminaro, Block preconditioners based on approximate commutators, 2006.

Stokes example

The common block preconditioners for Stokes require only options:

```
-pc_type fieldsplit
-pc_field_split_type schur
-pc_fieldsplit_schur_factorization_type full
```

PC

$$\begin{pmatrix} I & 0 \\ B^T A^{-1} & I \end{pmatrix} \begin{pmatrix} \hat{A} & 0 \\ 0 & \hat{S} \end{pmatrix} \begin{pmatrix} I & A^{-1}B \\ 0 & I \end{pmatrix}$$

Stokes example

All block preconditioners can be *embedded* in MG using only options:

```
-pc_type mg -pc_mg_levels 5 -pc_mg_galerkin
-mg_levels_pc_type fieldsplit
-mg_levels_pc_field_split_type
```

System on each Coarse Level

$$R\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} P$$

Stokes example

All block preconditioners can be *embedded* in MG using only options:

```
-pc_type mg -pc_mg_levels 5 -pc_mg_galerkin
-mg_levels_pc_type fieldsplit
-mg_levels_pc_field_split_type additive

-mg_levels_fieldsplit_0_pc_type gamg
-mg_levels_fieldsplit_0_ksp_type preonly

-mg_levels_fieldsplit_1_pc_type jacobi
-mg_levels_fieldsplit_1_ksp_type preonly
```

Smoother PC

$$\begin{pmatrix} \hat{A} & 0 \\ 0 & I \end{pmatrix}$$

Stokes example

All block preconditioners can be *embedded* in MG using only options:

```
-pc_type mg -pc_mg_levels 5 -pc_mg_galerkin
-mg_levels_pc_type fieldsplit
-mg_levels_pc_field_split_type
multiplicative

-mg_levels_fieldsplit_0_pc_type gamg
-mg_levels_fieldsplit_0_ksp_type preonly

-mg_levels_fieldsplit_1_pc_type jacobi
-mg_levels_fieldsplit_1_ksp_type preonly
```

Smoother

PC

$$\begin{pmatrix} \hat{A} & B \\ 0 & I \end{pmatrix}$$

Stokes example

All block preconditioners can be *embedded* in MG using only options:

```
-pc_type mg -pc_mg_levels 5 -pc_mg_galerkin
-mg_levels_pc_type fieldsplit
-mg_levels_pc_field_split_type schur

-mg_levels_fieldsplit_0_pc_type gamg
-mg_levels_fieldsplit_0_ksp_type preonly

-mg_levels_fieldsplit_1_pc_type none
-mg_levels_fieldsplit_1_ksp_type minres
```

Smoother PC

$$\begin{pmatrix} \hat{A} & 0 \\ 0 & -\hat{S} \end{pmatrix}$$

```
-mg_levels_pc_fieldsplit_schur_factorization_type diag
```

Stokes example

All block preconditioners can be *embedded* in MG using only options:

```
-pc_type mg -pc_mg_levels 5 -pc_mg_galerkin
-mg_levels_pc_type fieldsplit
-mg_levels_pc_field_split_type schur

-mg_levels_fieldsplit_0_pc_type gamg
-mg_levels_fieldsplit_0_ksp_type preonly

-mg_levels_fieldsplit_1_pc_type none
-mg_levels_fieldsplit_1_ksp_type minres
```

Smoother PC

$$\begin{pmatrix} \hat{A} & 0 \\ B^T & \hat{S} \end{pmatrix}$$

```
-mg_levels_pc_fieldsplit_schur_factorization_type lower
```

Stokes example

All block preconditioners can be *embedded* in MG using only options:

```
-pc_type mg -pc_mg_levels 5 -pc_mg_galerkin
-mg_levels_pc_type fieldsplit
-mg_levels_pc_field_split_type schur

-mg_levels_fieldsplit_0_pc_type gamg
-mg_levels_fieldsplit_0_ksp_type preonly

-mg_levels_fieldsplit_1_pc_type none
-mg_levels_fieldsplit_1_ksp_type minres
```

Smoother PC

$$\begin{pmatrix} \hat{A} & B \\ 0 & \hat{S} \end{pmatrix}$$

```
-mg_levels_pc_fieldsplit_schur_factorization_type upper
```

Stokes example

All block preconditioners can be *embedded* in MG using only options:

```
-pc_type mg -pc_mg_levels 5 -pc_mg_galerkin
-mg_levels_pc_type fieldsplit
-mg_levels_pc_field_split_type schur

-mg_levels_fieldsplit_0_pc_type gamg
-mg_levels_fieldsplit_0_ksp_type preonly

-mg_levels_fieldsplit_1_pc_type lsc
-mg_levels_fieldsplit_1_ksp_type minres
```

Smoother PC

$$\begin{pmatrix} \hat{A} & B \\ 0 & \hat{S}_{\mathrm{LSC}} \end{pmatrix}$$

```
-mg_levels_pc_fieldsplit_schur_factorization_type upper
```

Why us FGMRES?

Flexible GMRES (FGMRES) allows a different preconditioner at each step:

- Takes twice the memory
- Needed for iterative PCs
- Avoided sometimes with a careful PC choice

Outline

Example 3D Hex8 `step10.cfg`

Okay, Computer Boy,
Can you do this for
a real PyLith Example?

Example 3D Hex8 `step10.cfg`

First, we try LU on the whole problem (`solver01.cfg`)

```
[pylithapp.petsc]
snes_view = true
pc_type = lu
```

FAIL

This is due to the saddle point introduced to handle the fault.

Example 3D Hex8 `step10.cfg`

First, we try LU on the whole problem (`solver01.cfg`)

```
[pylithapp.petsc]
snes_view = true
pc_type = lu
```

FAIL

This is due to the saddle point introduced to handle the fault.

Example 3D Hex8 `step10.cfg`

First, we try LU on the whole problem (`solver01.cfg`)

```
[pylithapp.petsc]
snes_view = true
pc_type = lu
```

FAIL

This is due to the saddle point introduced to handle the fault.

Example 3D Hex8 `step10.cfg`

Next, we split fields using `PC_FIELDSPLIT` (`solver02.cfg`)

```
[pylithapp.timedependent.formulation]
split_fields = True
matrix_type = aij
[pylithapp.petsc]
snes_view = true
ksp_monitor_true_residual = true
fs_pc_type = fieldsplit
fs_pc_fieldsplit_real_diagonal = true
fs_pc_fieldsplit_type = additive
fs_fieldsplit_0_ksp_type = preonly
fs_fieldsplit_0_pc_type = lu
fs_fieldsplit_1_ksp_type = gmres
fs_fieldsplit_1_ksp_rtol = 1.0e-11
fs_fieldsplit_1_pc_type = jacobi
```

Does not converge because preconditioner is not strong enough

Example 3D Hex8 `step10.cfg`

Next, we split fields using `PC_FIELDSPLIT` (`solver02.cfg`)

```
[pylithapp.timedependent.formulation]
split_fields = True
matrix_type = aij
[pylithapp.petsc]
snes_view = true
ksp_monitor_true_residual = true
fs_pc_type = fieldsplit
fs_pc_fieldsplit_real_diagonal = true
fs_pc_fieldsplit_type = additive
fs_fieldsplit_0_ksp_type = preonly
fs_fieldsplit_0_pc_type = lu
fs_fieldsplit_1_ksp_type = gmres
fs_fieldsplit_1_ksp_rtol = 1.0e-11
fs_fieldsplit_1_pc_type = jacobi
```

Does not converge because preconditioner is not strong enough

Example 3D Hex8 `step10.cfg`

We need to use a full Schur factorization (`solver03.cfg`)

```
fs_pc_type = fieldsplit
fs_pc_fieldsplit_real_diagonal = true
fs_pc_fieldsplit_type = schur
fs_pc_fieldsplit_schur_factorization_type = full
fs_fieldsplit_0_ksp_type = preonly
fs_fieldsplit_0_pc_type = lu
fs_fieldsplit_1_ksp_type = gmres
fs_fieldsplit_1_ksp_rtol = 1.0e-11
fs_fieldsplit_1_pc_type = jacobi
```

Works in one iterate! This is good for checking the physics.

Example 3D Hex8 `step10.cfg`

We need to use a full Schur factorization (`solver03.cfg`)

```
fs_pc_type = fieldsplit
fs_pc_fieldsplit_real_diagonal = true
fs_pc_fieldsplit_type = schur
fs_pc_fieldsplit_schur_factorization_type = full
fs_fieldsplit_0_ksp_type = preonly
fs_fieldsplit_0_pc_type = lu
fs_fieldsplit_1_ksp_type = gmres
fs_fieldsplit_1_ksp_rtol = 1.0e-11
fs_fieldsplit_1_pc_type = jacobi
```

Works in one iterate! This is good for checking the physics.

Example 3D Hex8 `step10.cfg`

We can add a user defined preconditioner for the Schur complement (`solver04.cfg`)

```
[pylithapp.timedependent.formulation]
use_custom_constraint_pc = True
[pylithapp.petsc]
fs_pc_fieldsplit_schur_precondition = user
```

Example 3D Hex8 `step10.cfg`

We can add a user defined preconditioner for the Schur complement (`solver04.cfg`)

```
[pylithapp.timedependent.formulation]
use_custom_constraint_pc = True
[pylithapp.petsc]
fs_pc_fieldsplit_schur_precondition = user
```

The initial convergence

```
0 SNES Function norm 1.547533880440e-02
  Linear solve converged due to CONVERGED_RTOL iterations 30
  0 KSP Residual norm 1.158385264202e-02
  Linear solve converged due to CONVERGED_RTOL iterations 30
  1 KSP Residual norm 2.198105129707e-13
Linear solve converged due to CONVERGED_RTOL iterations 1
1 SNES Function norm 1.146157083300e-13
```

Example 3D Hex8 `step10.cfg`

We can add a user defined preconditioner for the Schur complement (`solver04.cfg`)

```
[pylithapp.timedependent.formulation]
use_custom_constraint_pc = True
[pylithapp.petsc]
fs_pc_fieldsplit_schur_precondition = user
```

improves to

```
0 SNES Function norm 1.547533880440e-02
  Linear solve converged due to CONVERGED_RTOL iterations 24
  0 KSP Residual norm 1.158385264203e-02
  Linear solve converged due to CONVERGED_RTOL iterations 25
  1 KSP Residual norm 5.404218646700e-14
Linear solve converged due to CONVERGED_RTOL iterations 1
1 SNES Function norm 2.200824144647e-14
```

and gets much better for larger problems.

Example 3D Hex8 `step10.cfg`

You can back off the Schur complement tolerance (`solver05.cfg`)

```
fs_fieldsplit_1_ksp_rtol = 1.0e-05
```

at the cost of more iterates

```
0 SNES Function norm 1.547533880440e-02
  Linear solve converged due to CONVERGED_RTOL iterations 10
  0 KSP Residual norm 1.158385275006e-02
  Linear solve converged due to CONVERGED_RTOL iterations 10
  1 KSP Residual norm 1.743099082900e-07
  Linear solve converged due to CONVERGED_RTOL iterations 15
  2 KSP Residual norm 9.111244675 71e-13
Linear solve converged due to CONVERGED_RTOL iterations 2
1 SNES Function norm 2.316774353785e-11
```

Example 3D Hex8 `step10.cfg`

You can back off the primal LU solver (`solver06.cfg`)

```
fs_fieldsplit_0_ksp_type = preonly
fs_fieldsplit_0_pc_type  = ml
```

at the cost of many more iterates

```
  0 SNES Function norm 1.547533880440e-02
  ...
 34 SNES Function norm 1.094751648499e-09
    0 KSP Residual norm 1.044862482330e-09
    1 KSP Residual norm 1.026476859438e-11
    2 KSP Residual norm 2.352619621602e-13
    3 KSP Residual norm 4.901870841230e-15
    4 KSP Residual norm 1.028487933615e-16
    5 KSP Residual norm 2.250903096143e-18
    6 KSP Residual norm 5.050245895484e-20
  35 SNES Function norm 4.074830594018e-10
Nonlinear solve converged due to CONVERGED_FNORM_ABS iterations 35
```

Example 3D Hex8 `step10.cfg`

You can restore the behavior with a lower tolerance (`solver07.cfg`)

```
fs_fieldsplit_0_ksp_type = gmres
fs_fieldsplit_0_ksp_rtol = 5.0e-10
```

but it is quite sensitive to the tolerance.

```
0 SNES Function norm 1.547533880440e-02
  Linear solve converged due to CONVERGED_RTOL iterations 10
  0 KSP Residual norm 1.158385274961e-02
  Linear solve converged due to CONVERGED_RTOL iterations 10
  1 KSP Residual norm 1.744541880226e-07
  Linear solve converged due to CONVERGED_RTOL iterations 15
  2 KSP Residual norm 1.585882433753e-12
  Linear solve converged due to CONVERGED_RTOL iterations 16
  3 KSP Residual norm 1.222018988543e-17
Linear solve converged due to CONVERGED_RTOL iterations 3
1 SNES Function norm 5.034307203820e-11
```