Supplement: Pragmatic Solvers for 3D Stokes and Elasticity Problems with Heterogeneous Coefficients: Evaluating Modern Incomplete LDL^T Preconditioners

We strive to make the work presented in the accompanying paper reproducible, interpretable, and extensible. We do so by providing our experimental application code, open-source, and similarly providing wrappers for the solver libraries employed in the experiments, in particular PARDISO[1] and ILUPACK.

1 Software, configuration, and settings

We provided additional specifics, sufficient to re-run our experiments, using the following software versions

- PETSc 3.12.4
- SuiteSparse 5.7.1
- ILUPACK 2.4 (05102016)
- MC21 1.0.0
- MC64 1.6.0
- PARDISO 6.0.0

Note that some of the experiments in the paper were run using older versions of this software, namely

- PETSc 3.7.7
- SuiteSparse 4.4.3
- PARDISO 5.0.0

but no performance differences have been observed with the upgraded versions.

1.1 Application Code

The main application code used to generate all results in the paper is available, open source under a BSD-2 license, at `https://bitbucket.org/psanan/exsaddle`. All description below refers to version 0.1, as defined by the Git tag `v0.1`.

[1]Note that we use recent versions of PARDISO, which represent continued development past the 2006 version of PARDISO available in Intel's MKL

1.1.1 PETSc

One should use PETSc 3.12 or later, configured with SuiteSparse via PETSc's `configure` script, but *without* METIS, as this can cause linking issues with ILUPACK. For full information on obtaining and configuring PETSc, see `https://mcs.anl.gov/petsc/installation.html`. An example configuration command is

```
./configure --with-cc-gcc --with-cxx=g++ --with-fc-gfortran --with-debugging=0
    --download-mpich --download-suitesparse --download-fblaslapack
```

If wanting to use the custom PARDISO wrapper as in Section 1.5, configure with additional options, e.g.

```
./configure --with-fc=mpif90 --with-cc=mpicc --with-cxx=mpiCC --with-valgrind=0
    --download-suitesparse --with-debugging=no --FOPTFLAGS=\"-g -O3\" --COPTFLAGS=\"-g
    -O3\" --CXXOPTFLAGS=\"-g -O3\" --known-mpi-shared-libraries=1
    --with-blaslapack-lib=-lopenblas --with-shared-libraries=0
    PETSC_ARCH=arch-leonard-simple --with-openmp --with-pardiso
    --with-pardiso-lib=$HOME/code/pardiso/libpardiso600-GNU720-X86-64.so
```

And ensure that `~/pardiso.lic` exists and is valid, corresponding to your username (`whoami`). To obtain a license, visit https://www.pardiso-project.org/.

1.1.2 ILUPACK

Download ILUPACK from `http://www.icm.tu-bs.de/~bolle/ilupack/` and place the unpacked `ilupack` directory into a location of your choosing. Obtain the `MC21` and `MC64` packages from `http://www.hsl.rl.ac.uk/`, and follow the instructions in `ilupack/notdistributed/README` to build the object files in 'ilupack/notdistributed' as instructed, e.g. place `mc21-1.0.0.tar.gz` and `mc64-1.6.0.tar.gz` in the same directory as `ilupack` and

```
tar xzvf mc21-1.0.0.tar.gz
tar xzvf mc64-1.6.0.tar.gz
cp mc64-1.6.0/src/mc64*.f ilupack/notdistributed/
cp mc21-1.0.0/src/mc21*.f ilupack/notdistributed/
cd ilupack/notdistributed
gfortran -O3 -fPIC -c -o MC21D.o mc21d.f
gfortran -O3 -fPIC -c -o MC64D.o mc64d.f
gfortran -O3 -fPIC -c -o MC21S.o mc21s.f
gfortran -O3 -fPIC -c -o MC64S.o mc64s.f
```

You must choose the appropriate "platform" value, here "GNU64", appropriate for 64-bit Linux systems, using GCC compilers.

To test, build a simple example, e.g.

```
cd ilupack/simple_examples
make -f MC64/makefile.GNU64 MAIN=dmainsym
./dmainsym.out
```

Expecting output like

```
 Output:
    factorization successful with 2 levels completed
    final elbow space factor=    2.01
    approximate solution after one preconditioning step
     -4.2725543538362838e-02
     -1.5897898097154955e-02
```

```
      1.1053236900891336e-01
     -1.2041337668369712e-01
      2.6074727770036638e-02
     -1.2240764087025115e-01
      2.0476595964287550e-01
      2.1187527186133781e-01

    iteration successful completed after 5 steps
    approximate solution after completing the iterative process
     -4.2725543538362859e-02
     -1.5897898097154934e-02
      1.1053236900891339e-01
     -1.2041337668369716e-01
      2.6074727770036631e-02
     -1.2240764087025119e-01
      2.0476595964287558e-01
      2.1187527186133778e-01

    residual norm      0.0
```

1.1.3 Building and Testing exSaddle

Obtain the exSaddle code, e.g.

```
git clone https://bitbucket.org/psanan/exsaddle
cd exsaddle
```

Ensure that you have `PETSC_DIR` and `PETSC_ARCH` properly defined in your environment, and run the following, replacing `$ILUPACK_PLATFORM` with the value you determined above (for instance, `GNU64`), and `$ILUPACK_DIR` with the location of your `ilupack` directory.

```
make EXSADDLE_WITH_PCILDL=1 ILUPACK_PLATFORM=$ILUPACK_PLATFORM ILUPACK_DIR=$ILUPACK_DIR
    -j
```

to produce the `exSaddle2d`,`exSaddle2d_lame`,`exSaddle3d`, and `exSaddle3d_lame` executables.

These are PETSc-based applications around a `KSP` (linear solver) object named `saddle`. This allows all solver parameters to be specified from the command line, as in the examples below.

Run any of these with the `-help` command-line option to see available options[2].

Run `make test` to run a suite of tests. No output indicates that the tests passed. These are very simple diff-based tests, so small perturbations in residual histories are not usually a cause for concern.

1.2 ILDL-preconditioned Solves

The ILDL preconditioner used to produce the results in this paper is available by supplying the command line option `-saddle_pc_type ildl`. Drop tolerance is controlled with the `-saddle_pc_ildl_droptol` argument.

```
$ ./exSaddle3d -saddle_ksp_type gmres -saddle_ksp_pc_side right -saddle_ksp_monitor
    -saddle_ksp_view -saddle_ksp_gmres_restart 60 -saddle_pc_type ildl
    -saddle_pc_ildl_droptol 1e-3
Boundary Conditions: SolCx
ModelType: StokesXSinker
```

[2]It may be helpful to only view the first part of the output, e.g. with `./exSaddle3d -help | less` or `./exSaddle3d -help intro`)

```
  params: eta0 1.0000e+00
  params: eta1 1.0000e+00
  params: num sinkers 3
  params: sinker radius 5.0000e-02
relative fill ILDL/A:  1.2e+00 (wrt 191755 nz)
  Residual norms for saddle_ solve.
  0 KSP Residual norm 4.951778290662e-02
  1 KSP Residual norm 8.555234687001e-05
  2 KSP Residual norm 8.738329705063e-07
  3 KSP Residual norm 4.127323187726e-08
KSP Object:(saddle_) 1 MPI processes
  type: gmres
    GMRES: restart=60, using Classical (unmodified) Gram-Schmidt Orthogonalization with
        no iterative refinement
    GMRES: happy breakdown tolerance 1e-30
  maximum iterations=10000, initial guess is zero
  tolerances:  relative=1e-05, absolute=1e-50, divergence=10000.
  right preconditioning
  using UNPRECONDITIONED norm type for convergence test
PC Object:(saddle_) 1 MPI processes
  type: ildl
    ILDL: matching : 1
    ILDL: droptol : 0.001
    ILDL: ordering : metisn
  linear system matrix = precond matrix:
  Mat Object:  Asaddle   1 MPI processes
    type: seqaij
    rows=2312, cols=2312
    total: nonzeros=381196, allocated nonzeros=542628
    total number of mallocs used during MatSetValues calls =0
      using I-node routines: found 854 nodes, limit used is 5
```

Note: our custom ILUPACK wrapper for PETSc is now also available as a stand-alone plugin, at `http://www.bitbucket.org/psanan/pcilupack`.

1.3 ILDL as a sub-preconditioner

ILDL sub-preconditioners within a block Jacobi preconditioner is specified at the command line, similar to the simple ILDL preconditioning, e.g.

```
$ mpiexec -np 2 ./exSaddle3d -saddle_ksp_type gmres -saddle_ksp_pc_side right
    -saddle_ksp_monitor -saddle_ksp_view -saddle_ksp_gmres_restart 60 -saddle_pc_type
    bjacobi -saddle_sub_pc_type ildl -saddle_sub_pc_ildl_droptol 1e-3 -mx 8
Boundary Conditions: SolCx
ModelType: StokesOneSinker
  params: eta0 1.0000e+00
  params: eta1 1.0000e+00
  params: x 5.0000e-01
  params: y 5.0000e-01
  params: z 5.0000e-01
  params: rad 2.5000e-01
relative fill ILDL/A:  2.0e+00 (wrt 635901 nz)
```

```
relative fill ILDL/A:  1.8e+00 (wrt 725376 nz)
  Residual norms for saddle_ solve.
  0 KSP Residual norm 1.802527444022e-02
  1 KSP Residual norm 1.653982450825e-02
  2 KSP Residual norm 1.545695555321e-02
  3 KSP Residual norm 5.760379268671e-03
  [ ... ]
 31 KSP Residual norm 3.274050646028e-07
 32 KSP Residual norm 2.491411899377e-07
 33 KSP Residual norm 1.398221660044e-07
KSP Object: (saddle_) 2 MPI processes
  type: gmres
    restart=60, 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.
  right preconditioning
  using UNPRECONDITIONED norm type for convergence test
PC Object: (saddle_) 2 MPI processes
  type: bjacobi
    number of blocks = 2
    Local solve is same for all blocks, in the following KSP and PC objects:
  KSP Object: (saddle_sub_) 1 MPI processes
    type: preonly
    maximum iterations=10000, initial guess is zero
    tolerances:  relative=1e-05, absolute=1e-50, divergence=10000.
    left preconditioning
    using NONE norm type for convergence test
  PC Object: (saddle_sub_) 1 MPI processes
    type: ildl
      ILDL: matching : 1
      ILDL: droptol : 0.001
      ILDL: ordering : metisn
    linear system matrix = precond matrix:
    Mat Object: 1 MPI processes
      type: seqaij
      rows=8127, cols=8127
      total: nonzeros=1442623, allocated nonzeros=1843242
      total number of mallocs used during MatSetValues calls=0
        using I-node routines: found 2925 nodes, limit used is 5
  linear system matrix = precond matrix:
  Mat Object: Asaddle 2 MPI processes
    type: mpiaij
    rows=15468, cols=15468
    total: nonzeros=2900776, allocated nonzeros=6623847
    total number of mallocs used during MatSetValues calls=0
      using I-node (on process 0) routines: found 2925 nodes, limit used is 5
```

1.4 ABF Solves

ABF solves are achieved with the help of a substantial amount of setup in the source code to define fields, a multigrid hierarchy, and an auxiliary operator (a weighted pressure mass matrix) to construct the Schur complement preconditioner. The `-fs` command line option sets this up. Other options control the standard PETSc `PCFIELDSPLIT` and `PCMG` objects. As an example,

```
$./exSaddle3d -diagnostics -eta0 1 -eta1 1e4 -fs -model 1 -mx 32
    -saddle_fieldsplit_p_ksp_type preonly -saddle_fieldsplit_p_pc_type bjacobi
    -saddle_fieldsplit_u_ksp_rtol 1e-2 -saddle_fieldsplit_u_ksp_type gcr
    -saddle_fieldsplit_u_mg_coarse_pc_factor_mat_solver_type umfpack
    -saddle_fieldsplit_u_mg_levels_ksp_chebyshev_esteig 0,0.2,0,1.1
    -saddle_fieldsplit_u_mg_levels_ksp_max_it 8
    -saddle_fieldsplit_u_mg_levels_ksp_norm_type none
    -saddle_fieldsplit_u_mg_levels_ksp_type chebyshev
    -saddle_fieldsplit_u_mg_levels_pc_type jacobi -saddle_fieldsplit_u_pc_mg_galerkin
    -saddle_fieldsplit_u_pc_mg_levels 4 -saddle_fieldsplit_u_pc_type mg
    -saddle_ksp_converged_reason -saddle_ksp_monitor_true_residual -saddle_ksp_type
    fgmres
ModelType: StokesThreeSinker
  params: eta0 1.0000e+00
  params: eta1 1.0000e+04
  params: rad  1.0000e-01
  Residual norms for saddle_ solve.
  0 KSP unpreconditioned resid norm 2.278031045078e-03 true resid norm
      2.278031045078e-03 ||r(i)||/||b|| 1.000000000000e+00
  1 KSP unpreconditioned resid norm 1.238140855553e-03 true resid norm
      1.238140855552e-03 ||r(i)||/||b|| 5.435136005838e-01
  2 KSP unpreconditioned resid norm 1.845156332083e-04 true resid norm
      1.845156332039e-04 ||r(i)||/||b|| 8.099785716376e-02
  3 KSP unpreconditioned resid norm 1.265891044760e-04 true resid norm
      1.265891044066e-04 ||r(i)||/||b|| 5.556952556908e-02
  4 KSP unpreconditioned resid norm 1.264971076325e-04 true resid norm
      1.264971075612e-04 ||r(i)||/||b|| 5.552914120048e-02
  5 KSP unpreconditioned resid norm 8.954226006787e-05 true resid norm
      8.954226000674e-05 ||r(i)||/||b|| 3.930686554962e-02
  6 KSP unpreconditioned resid norm 5.238928261668e-05 true resid norm
      5.238928258037e-05 ||r(i)||/||b|| 2.299761572327e-02
  7 KSP unpreconditioned resid norm 3.870280190575e-05 true resid norm
      3.870280187272e-05 ||r(i)||/||b|| 1.698958491208e-02
  8 KSP unpreconditioned resid norm 2.378563111831e-05 true resid norm
      2.378563110670e-05 ||r(i)||/||b|| 1.044131121834e-02
  9 KSP unpreconditioned resid norm 1.027753186333e-05 true resid norm
      1.027753185036e-05 ||r(i)||/||b|| 4.511585508270e-03
 10 KSP unpreconditioned resid norm 2.829804826129e-06 true resid norm
     2.829804822497e-06 ||r(i)||/||b|| 1.242215214148e-03
 11 KSP unpreconditioned resid norm 2.317359346406e-06 true resid norm
     2.317359321416e-06 ||r(i)||/||b|| 1.017264152928e-03
 12 KSP unpreconditioned resid norm 1.973189171382e-06 true resid norm
     1.973189131869e-06 ||r(i)||/||b|| 8.661818442430e-04
 13 KSP unpreconditioned resid norm 1.183114493199e-06 true resid norm
     1.183114477982e-06 ||r(i)||/||b|| 5.193583645569e-04
```

```
 14 KSP unpreconditioned resid norm 4.887317032905e-07 true resid norm
    4.887316979199e-07 ||r(i)||/||b|| 2.145412807151e-04
 15 KSP unpreconditioned resid norm 1.860725577509e-07 true resid norm
    1.860725689659e-07 ||r(i)||/||b|| 8.168131394343e-05
 16 KSP unpreconditioned resid norm 3.622406258094e-08 true resid norm
    3.622405234457e-08 ||r(i)||/||b|| 1.590147439950e-05
 17 KSP unpreconditioned resid norm 1.391510111378e-08 true resid norm
    1.391511173052e-08 ||r(i)||/||b|| 6.108394247124e-06
Linear saddle_ solve converged due to CONVERGED_RTOL iterations 17
|u,v,w|_1   +1.448198e+00 , +9.195233e+00 , +2.509789e+00
|u,v,w|_2   +3.749246e-03 , +2.270849e-02 , +6.411538e-03
|u,v,w|_inf +3.740597e-05 , +1.330357e-04 , +4.139363e-05
|u,v,w|_min -3.740597e-05 , -1.330357e-04 , -4.139363e-05
|u,v,w|_max +2.668742e-05 , +7.424197e-05 , +4.139001e-05
|p|_1          +1.799549e+04
|p|_2          +1.104631e+02
|p|_inf        +1.001764e+00
|p|_min        -3.625674e-04
|p|_max        +1.001764e+00
```

1.5 PARDISO Direct Solves

Our custom interface is available at `https://bitbucket.org/psanan/petsc/branch/psanan/pardiso-3.12.4`

You may use this branch of PETSc directly or you may `git cherry-pick` the last commit onto your preferred branch. We use the following non-default PARDISO settings.

- `MTYPE =-2` (Symmetric indefinite system),
- `IPARM(1) = 1` (Use non-default settings),
- `IPARM(11)= 1` (Scaling vectors),
- `IPARM(13)= 2` (Maximum weighted matching permutation),
- `IPARM(21)= 1` (2x2 Bunch-Kaufman pivoting)

See the PARDISO Users' Manual at `http://www.pardiso-project.org/manual/manual.pdf` for more information

This amounts to command line options as in the following example, which runs the 3-sinker Stokes example with a 10^4 viscosity contrast on a small, 8^3 element mesh.

```
./exSaddle3d -diagnostics -eta0 1 -eta1 1e4 -mat_pardiso_1 1 -mat_pardiso_11 1
    -mat_pardiso_13 2 -mat_pardiso_21 1 -model 1 -mx 8 -saddle_ksp_converged_reason
    -saddle_ksp_monitor_true_residual -saddle_ksp_type fgmres
    -saddle_pc_factor_mat_solver_type pardiso -saddle_pc_type cholesky -sbaijhack
Boundary Conditions: SolCx
ModelType: StokesThreeSinker
  params: eta0 1.0000e+00
  params: eta1 1.0000e+04
  params: rad  1.0000e-01


================================================================================
CONTAINS Runtime Modules of Parallel Sparse Linear Solver PARDISO Vers. 6.0
Copyright Universita della Svizzera Italiana 2000-2018 All Rights Reserved.
```

```
TIME-LIMITED ACADEMIC SINGLE-USER LICENSE: YOU ARE USING THE PARDISO SOLVER
UNDER THE CONTROL OF AN ACADEMIC SINGLE-USER LICENSE. YOU AS  AN INDIVIDUAL
MAY INSTALL AND USE PARDISO ON AN  UNLIMITED  NUMBER  OF COMPUTERS PROVIDED
THAT YOU  ARE THE  ONLY INDIVIDUAL USING  THE  PARDISO SOLVER. THIS LICENSE
IS TIED TO A SINGLE (IDENTICAL) USER NAME THAT YOU NEED ON THESE COMPUTERS.
A SEPARATE LICENSE IS REQUIRED FOR EACH INDIVIDUAL USER IN ALL OTHER CASES.
USI LUGANO  MAY  PROVIDE  YOU  WITH  A LICENSE  CODE KEY  THAT  ENABLES THE
PARDISO  SOLVER  FOR  ADDITIONAL  USERS.   THE  DURATION  (TIME  PERIOD) OF
YOUR LICENSE  AND  YOUR  ABILITY TO USE  THE PARDISO SOLVER  IS  LIMITED TO
THE TIME PERIOD OF THE OBTAINED LICENSE, WHICH IS CONTROLLED BY THE LICENSE
KEY CODE. THE ACADEMIC LICENSE PERMITS THE USE OF PARDISO ONLY FOR ACADEMIC
RESEARCH. IF YOU ARE USING THE SOFTWARE FOR ANY OTHER  PURPOSE, YOU MAY  BE
IN  AN UNLAWFUL VIOLATION  OF THE  TERMS  OF THIS  LICENSE,  AND  YOU  MUST
OBTAIN  A VALID  ALTERNATIVE  LICENSE.
USI LUGANO DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING
ALL IMPLIED  WARRANTIES  OF  MERCHANTABILITY  AND  FITNESS. IN  NO  EVENT
SHALL USI  LUGANO  OR  ANY  OF  ITS  ENTITIES  BE  LIABLE FOR  ANY SPECIAL,
INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER  RESULTING FROM
LOSS OF USE, DATA OR PROFITS,  WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE
OR OTHER TORTIOUS ACTION,  ARISING OUT OF OR IN  CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.

       Please visit http://www.pardiso-project.org for more details.

 - You can suppress this message by setting the environment variable
   with e.g. export PARDISOLICMESSAGE=1
*******************************************************************************
  Residual norms for saddle_ solve.
  0 KSP unpreconditioned resid norm 1.792493041351e-02 true resid norm
      1.792493041351e-02 ||r(i)||/||b|| 1.000000000000e+00
  1 KSP unpreconditioned resid norm 2.350514761899e-16 true resid norm
      2.362879025743e-16 ||r(i)||/||b|| 1.318208200107e-14
Linear saddle_ solve converged due to CONVERGED_RTOL iterations 1
|u,v,w|_1   +5.527023e-03 , +6.024710e-02 , +1.351639e-02
|u,v,w|_2   +1.332324e-04 , +9.994750e-04 , +3.495187e-04
|u,v,w|_inf +1.591308e-05 , +3.115286e-05 , +2.587964e-05
|u,v,w|_min -1.591308e-05 , -1.592002e-05 , -2.587964e-05
|u,v,w|_max +1.268139e-05 , +3.115286e-05 , +2.587964e-05
|p|_1          +3.649459e+02
|p|_2          +1.608723e+01
|p|_inf        +1.004530e+00
|p|_min        -1.755514e-04
|p|_max        +1.004530e+00
```

1.6 Parallel solves

Parallel solves using rank-local ILDL sub-preconditioners are specified from the command line. Our application code uses a custom domain decomposition based on finite elements to define overlapping subdomains, but a purely algebraic approach based on the system matrix showed comparable results.

Example options for GMRES / ASM / ILDL:

```
mpiexec -np 27 ./exSaddle3d \
-eta0 1 -eta1 1e2 \
-model 6 -sinker_r 0.25 \
-mx 48 \
-saddle_ksp_type gmres -saddle_ksp_pc_side right -saddle_ksp_gmres_restart 60 \
-saddle_pc_type bjacobi \
-saddle_sub_pc_ildl_droptol 1e-3 \
-saddle_sub_pc_type ildl \
-log_view \
```

Example options for GMRES / Block Jacobi / ILDL:

```
mpiexec -np 27 ./exSaddle3d \
-eta0 1 -eta1 1e2 \
-model 6 \
-sinker_r 0.25 \
-mx 48 \
-saddle_ksp_type gmres -saddle_ksp_gmres_restart 60 -saddle_ksp_pc_side right \
-saddle_ksp_max_it 100000 \
-saddle_pc_asm_dm_subdomains \
-dmdafe_overlap 1 \
-set_ksp_dm \
-saddle_pc_type asm \
-saddle_sub_pc_ildl_droptol 1e-3 \
-saddle_sub_pc_type ildl \
-log_view \
```

We note that the largest ABF solve used for comparison required the use of PCTELESCOPE, corresponding to an options stack, for example:

```
mpiexec -np 512 ./exSaddle3d \
-eta0 1 -eta1 1e2 \
-fs \
-model 6 \
-sinker_r 0.25 \
-mx 128 \
-saddle_ksp_max_it 100000 \
-saddle_ksp_type fgmres \
-saddle_fieldsplit_p_ksp_type preonly \
-saddle_fieldsplit_p_pc_type bjacobi \
-saddle_fieldsplit_u_ksp_rtol 1e-2 \
-saddle_fieldsplit_u_ksp_type gcr \
-saddle_fieldsplit_u_pc_type mg \
-saddle_fieldsplit_u_pc_mg_galerkin \
-saddle_fieldsplit_u_pc_mg_levels 5 \
-saddle_fieldsplit_u_mg_levels_ksp_chebyshev_esteig 0,0.2,0,1.1 \
-saddle_fieldsplit_u_mg_levels_ksp_max_it 8 \
-saddle_fieldsplit_u_mg_levels_ksp_norm_type none \
-saddle_fieldsplit_u_mg_levels_ksp_type chebyshev \
-saddle_fieldsplit_u_mg_levels_pc_type jacobi \
-saddle_fieldsplit_u_mg_coarse_pc_factor_mat_solver_type umfpack \
-saddle_fieldsplit_u_mg_coarse_pc_telescope_reduction_factor 16 \
```

```
-saddle_fieldsplit_u_mg_coarse_pc_type telescope \
-saddle_fieldsplit_u_mg_coarse_telescope_pc_mg_galerkin \
-saddle_fieldsplit_u_mg_coarse_telescope_pc_mg_levels 2 \
-saddle_fieldsplit_u_mg_coarse_telescope_pc_type mg \
-saddle_fieldsplit_u_mg_coarse_telescope_mg_levels_esteig_ksp_norm_type none \
-saddle_fieldsplit_u_mg_coarse_telescope_mg_levels_ksp_chebyshev_esteig 0,0.2,0,1.1 \
-saddle_fieldsplit_u_mg_coarse_telescope_mg_levels_ksp_max_it 8 \
-saddle_fieldsplit_u_mg_coarse_telescope_mg_levels_ksp_norm_type none \
-saddle_fieldsplit_u_mg_coarse_telescope_mg_levels_ksp_type chebyshev \
-saddle_fieldsplit_u_mg_coarse_telescope_mg_levels_pc_type jacobi \
-saddle_fieldsplit_u_mg_coarse_telescope_mg_coarse_pc_type redundant \
-saddle_fieldsplit_u_mg_coarse_telescope_mg_coarse_redundant_pc_type lu \
-saddle_fieldsplit_u_mg_coarse_telescope_mg_coarse_redundant_pc_factor_mat_solver_type
    umfpack \
-log_view \
```

1.7 Model Settings

For the Stokes problems presented, viscosity contrasts were induced by setting `-eta0 1` and `-eta1 <constrast>`. Similarly, Lamé parameters were controlled with `-lambda0`, `-lambda1`,`-mu0`, and `-mu1`. Values of `-model` and other parameters used were

- Stokes (2D), 8 inclusions: `-model 2 -sinker_n 8 -sinker_r 0.05`
- Stokes, 1 inclusion: `-model 6 -sinker_r 0.25`
- Stokes, 3 inclusion: `-model 1`
- Stokes, 8 inclusion: `-model 7 -sinker_n 10 -sinker_r 0.1`
- Lamé, 1 inclusion: `-model 6 -sinker_r 0.25`
- Lamé, 3 inclusion: `-model 2 -sinker_n 3`
- Lamé, 8 inclusion:`-model 2 -sinker_n 8`
- Lamé, 1 inclusion, compression : `-model 10 -sinker_r 0.25`