Solvers and Scaling Lab: Flow Solvers and Parallel Scaling Comparison

1. Download problem1.tcl – problem10.tcl GitHub

Notes:

Problems 1-8 solve fully-saturated, steady state groundwater flow with a uniform rectangular domain and constant-head boundary conditions on two sides, no flow boundary conditions on all other sides. Problems 9 and 10 solve for variably saturated flow using Richards' equation with a domain similar to 1-8.

Problems 1-8 automatically run with *four* different *linear* solvers:

a. PCG- a multigrid-preconditioned conjugate gradient solver
b. MGSemi – an algebraic mulitgrid solver
c. PPCG- a preconditioned conjugate gradient solver
d. CGHS- a conjugate gradient solver

Problems 9-10 automatically run with different *linear preconditioners* and *nonlinear solver options*.

All binary output (pressure solution, subsurface) is suppressed, only log files are output. The solver-specific output is in a file called *problemX.SOLVER.out.log* where *X* is the problem number and *SOLVER* is e.g. pcg, MGSemi, PPCG and CGHS or the different linear and nonlinear options for each case.

In the log file the iteration count and convergence is printed along with the total run time and breakdown of run time for each component. This information is *also* written in the <OUT>.timing.csv file. Note that the total run time is everything combined and that the solver (see below) the important numbers you want to record are the solver time *minus* the subsurface sim time (which includes time to generate the domain, etc). This would be 118.636 – 3.3667 = 115.269 sec in the example below.

Solver Setup:
wall clock time = 0.069100 seconds
wall MFLOPS = 0.000000 (0)
Solver:
wall clock time = 118.636000 seconds
wall MFLOPS = 629.384504 (7.46677e+10)
Solver Cleanup:
wall clock time = 0.024000 seconds
wall MFLOPS = 0.000000 (0)
Matvec:
wall clock time = 26.162200 seconds
wall MFLOPS = 787.716629 (2.06084e+10)
MGSemi:
wall clock time = 115.029200 seconds
wall MFLOPS = 649.119180 (7.46677e+10)
Geometries:

wall clock time = 0.048500 seconds
wall MFLOPS = 0.000000 (0)
SubsrfSim:
wall clock time = 3.366700 seconds
wall MFLOPS = 0.000000 (0)
Porosity:
wall clock time = 0.002700 seconds
wall MFLOPS = 0.000000 (0)

Total Run Time: 119.079800 seconds

Also, in the log file is the iteration count along with the solver residual. All cases should be converging to *below* 1E-10 (the third column, below), you should confirm this and record the iteration count for the last iterative step. This would be 1923 in the example below. Pay close attention to this file and these numbers. Solvers may diverge, and not be able to solve a particular problem at all, or may not solve in the maximum number of iterations allowed for a particular case (set as the *Solver.Linear MaxIter* key in the input script). This will be particularly important for problems 3 and 4.

1919	*2.227576e-06*	*1.160993e-10*	*4.562257e+04*	*0*
1920	*2.090630e-06*	*1.089618e-10*	*4.562257e+04*	*0*
1921	*1.987819e-06*	*1.036033e-10*	*4.562257e+04*	*0*
1922	*1.933075e-06*	*1.007501e-10*	*4.562257e+04*	*0*
1923	*1.754106e-06*	*9.142243e-11*	*4.562257e+04*	*0*

2. For all cases, along with different parameters values for cases 3,4 and 7, you should run the case and plot the iteration count and the simulation time (as described in 6, above) for each solver. This should be tabulated as a table and as simple line graphs.
3. First run cases 1, 2, 5 and 6. (*problem1.tcl problem2.tcl*, *problem5.tcl* and *problem6.tcl*) The problems are all homogenous and vary only in size. Case 1 is 1M cells (100x100x100), case 2, 500K cells (100x50x100), case 5 250K cells (50x50x100) and case 6 125K cells (50x50x50 cells). As you plot up the results pay careful attention to the solver time and problem size. Does the time spent in each solver change relative to problem size? How should it change? Is the order (i.e. fastest to slowest) the same for each given problem size?
4. Then run case 3 (problem 3.tcl). This case is heterogeneous, with a uniform block in the middle of the domain and is 50x50x50 cells. Change the hydraulic conductivity of the block (*Geom.block.Perm.Value*) from 10 [L/T] to 0.0001 [L/T] in log increments noting that the remainder of the domain is 1.0 [L/T]. How does this change solver performance? Doe all the solvers take the same amount of time? The same number of iterations? Should they?
5. Then run case 4 (problem4.tcl) . This problem is heterogeneous, using a correlated, Gaussian random field to represent heterogeneity. This problem

can be made more or less heterogeneous by adjusting the standard deviation of the lognormal distribution used to generate the field. This is controlled using the *Geom.domain.Perm.Sigma* key. Note the examples in the script range from very little heterogeneity (std dev=0.1, variance=0.01) to really large variances (5, 10, 25, 50) that are exceedingly difficult problems to solve. Step up the variance gradually. Do all solvers perform the same for different variances/strengths of heterogeneity? Do all solvers take the same amount of time to solve? The same number of iterations? Do they all solve?

6. The next problem (*problem7.tcl*) is set up to be run in parallel using so-called "strong-scaling" where the problem size is held constant and split onto more processors. The base problem is homogeneous, with 2M cells (100x100x200), and is slit onto 2p in the script. The computers in the lab have 4 cores and should be able to be split in either 2x2 or 4x1 in either direction by changing *P Q* and *R* in the *Process.Topology* keys. Change the keys and note the solver performance and runtimes for each solver. Do all solvers show the same speedup? Is the performance from this problem the same as smaller problems (1, 2, 5, 6) run in serial? Should they be? Is the problem speedup linear? What might contribute to good or bad parallel performance? Does the answer (or the residual norm) change with parallelization?
7. Problems 8 and 10 are so-called "weak scaling" example where the processor size increases linearly with the number of parallel processes. You can references the papers on scaling, measures of scaling and scaled parallel efficiency. There is also information on the ParFlow blog article on scaling http://parflow.blogspot.com/2007/09/running-parflow-in-parallel.html

Vary the size/processor and splitting to determine if that affects SPE.