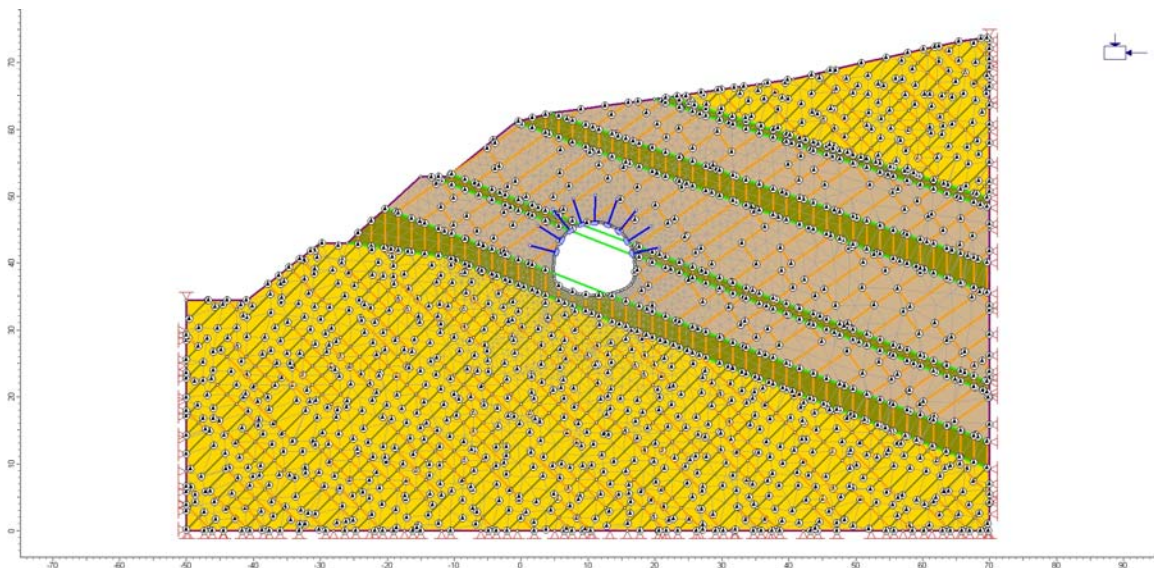


Phase² Now Using Intel[®] Performance Libraries for Faster Solutions

The problem

One of the goals of Phase² 7.0 was to enable users to easily generate large, complex joint networks. The problem was that these models quickly overwhelmed the computer's ability to compute a solution! With the new joint set generation capabilities, it is possible to create models with over 100,000 degrees of freedom (50,000 nodes). The solver in Phase² 6.0 could not handle such large problems and generally the computer's memory would overflow (gauss elimination solver) or the solution would take an extraordinarily long time (conjugate gradient solver). A new type of solver was needed.



Model created in Phase² 7.0 showing a tunnel excavation in a jointed rock. The model has 11,500 degrees of freedom.

The solution

Instead of spending time and money developing a new solver, we decided to go for a third party solution. After some investigation it was determined that the Intel[®] Math Kernel Library would suit our needs. To quote from the [webpage](#):

"Intel[®] Math Kernel Library (Intel[®] MKL) offers highly optimized, extensively threaded math routines for scientific, engineering, and financial applications that require maximum performance."

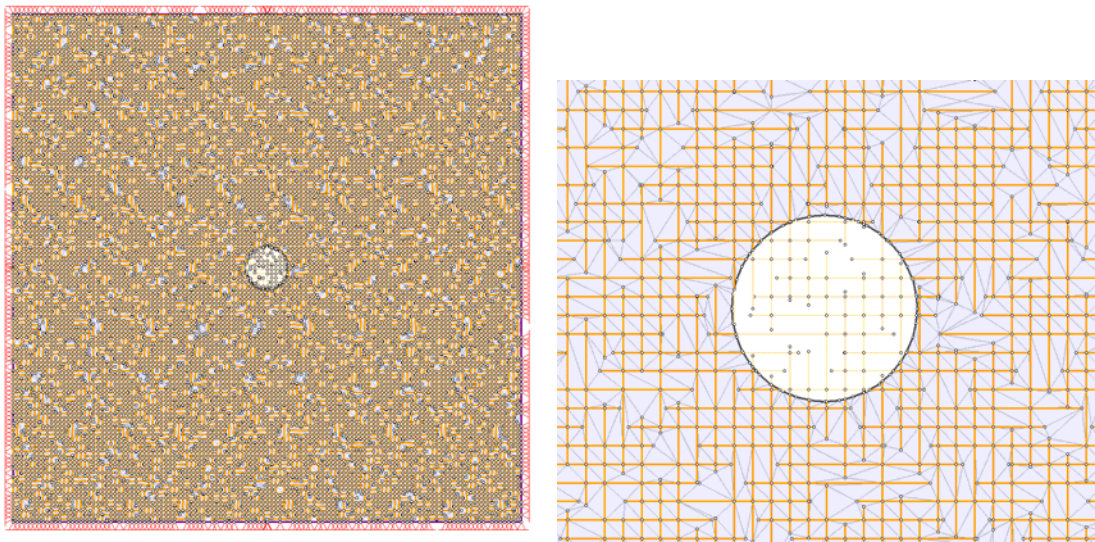
Of course, it was not simply a matter of calling the new functions and getting the answer. The problem matrix needed to be updated from the old *skyline* storage scheme to the more efficient *compressed sparse row* format used by the Intel[®] solver. And of course extensive testing was required ...

Results

The model shown in the above figure was solved using Gaussian elimination with the old skyline solver and the new Intel[®] solver¹. Results are shown in the table below.

	Time (s)	Memory (MB)
Old solver:	31	78
New solver:	23	63
Difference:	26 %	19 %

A 26% speed up for this model is quite encouraging. Now the question is, can the solver handle a much larger problem? We came up with a jointed tunnel model with ~80,000 degrees of freedom as shown.



Model of a tunnel in jointed rock. The plot on the right shows a close-up of the centre of the model. The model has approximately 80,000 degrees of freedom.

The results for this model are shown below.

	Time (m:s)	Memory (MB)
Old solver (gauss elimination):	Aborted – out of memory	
New solver (gauss elimination):	2:42	317
New solver (gauss elimination, out-of-core)	9:52	292
Old solver (conjugate gradient):	28:20	600
New solver (conjugate gradient):	18:20	266

¹ All tests were conducted on a Dell Dimension 8300 with a 3.2 GHz Pentium 4 processor and 1 GB of RAM

You can see that *Phase*² was unable to solve the problem with the old gauss elimination solver. There was not enough memory in our test computer. Before *Phase*² 7.0, the only way to solve this problem was to use the conjugate gradient solver. This uses less memory but takes much longer to solve (almost half an hour for this model). The old conjugate gradient solver will also swap memory between the RAM and the hard drive if necessary – so it will never crash due to lack of memory.

The new gauss elimination solver uses a much more efficient data storage scheme – so the memory use is significantly less. For this model, the problem was easily solved by our test computer in less than 3 minutes – more than ten times faster than the old conjugate gradient solver.

The new gauss elimination solver will also automatically swap memory with the hard drive if there is not enough RAM memory (called out-of-core solution). This was not necessary for the current model, but to test the performance, we forced the solver to perform the out-of-core calculations. The results are shown in the above table. This approach is quite a bit slower than the ‘in memory’ solution (3.6× longer to solve), however it is quite a bit faster than the old conjugate gradient solver (2.9× faster) and even about twice as fast as the new conjugate gradient solver.

Comments

The benefits of the new solver are really only realized for large models. For small models with many iterations, you will not see a significant speed up.

The figure below shows the time taken to solve a simple hole-in-plate model with Mohr-Coulomb failure (verification problem #2 from the *Phase*² manual). The same model was solved multiple times with increasing degrees of freedom. You can see that for small models, there is little difference in the solution time, but as the model size increases, the benefits of the new solver are realized.

The combination of a new data storage scheme and the fast Intel[®] solver means that even your largest models can now be solved in a reasonable amount of time on a standard PC.

