Linear Elasticity Tutorial 2 PSD simulation of bar problem using a sequential solver (non parallel)

Mohd Afeef Badri

Abstract

This tutorial details how to use 'linear elasticity' module of PSD, however we will not use a parallel solver but a sequential one. Naturally, this tutorial leads to a slow solver than the previous tutorial 1. So this tutorial is not for speed lovers, but rather for detailing the full capacity of PSD. Also sequential solvers are easier to develop and understand hence this tutorial.

Same problem of Linear elasticity as in tutorial 1 – 2D bar which bends under its own load –, is discuss here. The bar 5×1 m^2 in area is made up of material with $\rho = 8 \times 10^3$, $E = 200 \times 10^9$, and $\nu = 0.3$. To avoid text repetition, readers are encouraged to go ahead with this tutorial only after tutorial 1.

As the problem remains same as tutorial 1, simply add -sequential flag to PSD_PreProcess flags from tutorial 1 for sequential solver. The flag -sequential signifies the use of sequential PSD solver. So the work flow for the 2D problem would be:

```
PSD_PreProcess -problem linear_elasticity -dimension 2 -bodyforceconditions 1 \
-dirichletconditions 1 -postprocess u -sequential
```

Similar to tutorial 1, We solve the problem using the given mesh file bar.msh. However now we need to use PSD_Solve_Seq instead of PSD_Solve, as such:

```
PSD_Solve_Seq Main.edp -mesh ./../Meshes/2D/bar.msh -v 0
```

Users are encouraged to try out the 3D problem with sequential solver.

Comparing CPU time

Naturally, since we are not using parallel PSD for solving, we lose the advantage of solving fast. To testify this claim checking solver timings can be helpful. PSD provides mean to time log your solver via -timelog flag. What this will do when you run your solver, on the terminal you will have information printed on what is the amount of time taken by each step of your solver. Warning, this will make your solver slower, as this action involves 'MPI_Barrier' routines for correctly timing operation.

An example work flow of 2D solver (parallel) with timelogging:

```
PSD_PreProcess -problem linear_elasticity -dimension 2 -bodyforceconditions 1 \
-dirichletconditions 1 -postprocess u -timelog
```

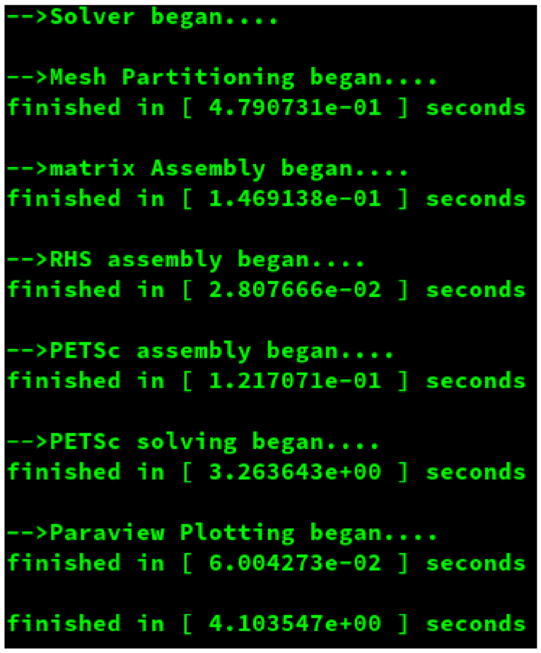
We solve the problem using four MPI processes, with the given mesh file bar.msh.

```
PSD_Solve -np 4 Main.edp -mesh ./../Meshes/2D/bar.msh -v 0
```

The figure~1 shows the time logging output produced for parallel run on 4 processes using -timelog flag. Take note of timings produced for different operations of the solver.

Now let us repeat the procedure but this time use sequential solver:

```
PSD_PreProcess -problem linear_elasticity -dimension 2 -bodyforceconditions 1 \
-dirichletconditions 1 -postprocess u -timelog -sequential
```



```
-->Solver began....

-->Mesh Partitioning began....
finished in [ 4.790731e-01 ] seconds

-->matrix Assembly began....
finished in [ 1.469138e-01 ] seconds

-->RHS assembly began....
finished in [ 2.807666e-02 ] seconds

-->PETSc assembly began....
finished in [ 1.217071e-01 ] seconds

-->PETSc solving began....
finished in [ 3.263643e+00 ] seconds

-->Paraview Plotting began....
finished in [ 6.004273e-02 ] seconds

finished in [ 4.103547e+00 ] seconds
```

Figure 1: Time logging output produced for parallel run on 4 processes.

We solve the problem now in sequential, with the given mesh file bar.msh.

```
PSD_Solve_Seq Main.edp -mesh ./../Meshes/2D/bar.msh -v 0
```

You should now see timings that are higher in comparison to the parallel solver. Approximately, for large meshes using 4 MPI processes should lead to 4 times fast solver.