Linear Elasticity Tutorials

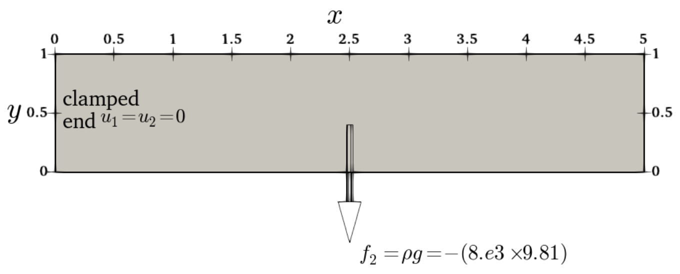


Figure 1: The 2D clamped bar problem.

Parallel 2D linear-elasticity

The problem of interest is a single Dirichlet condition (clamped end 2D bar) and body force source term (see, figure~1). Additionally postrocessing is demanded for displacement u.

```
PSD_PreProcess -problem linear-elasticity -dimension 2 -bodyforceconditions 1 \
-dirichletconditions 1 -postprocess u
```

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
```

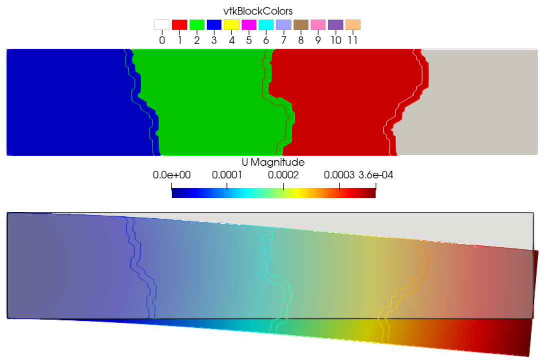


Figure 2: The 2D clamped bar problem: partitioned mesh and displacement field visualization in ParaView.

Using ParaView for postprocessing the results that are provided in the VTUs... folder, results such as those shown in figure~2 can be extracted.

Parallel 3D linear-elasticity

The problem of interest is a single Dirichlet condition (clamped end 3D bar) and body force source term. Additionally postrocessing is demanded for displacement u.

```
PSD_PreProcess -problem linear-elasticity -dimension 3 -bodyforceconditions 1 \
-dirichletconditions 1 -postprocess u
```

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

```
PSD_Solve -np 4 Main.edp -mesh ./../Meshes/3D/bar.msh -v 0
```

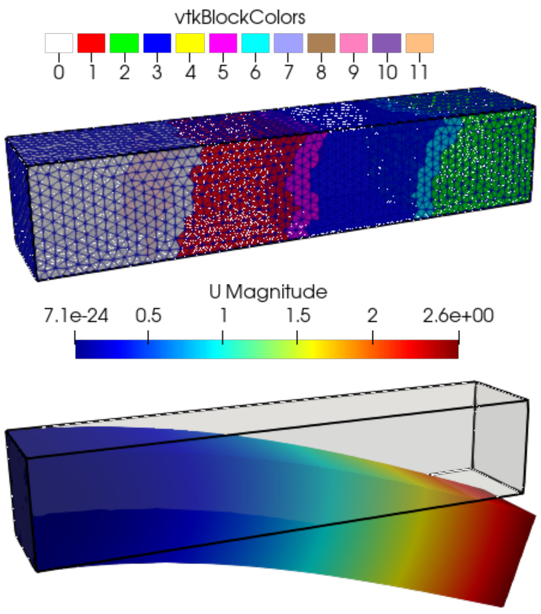


Figure 3: The 3D clamped bar problem: partitioned mesh and displacement field visualization in ParaView.

Using ParaView for postprocessing the results that are provided in the VTUs... folder, results such as those shown in figure~3 can be extracted.

Sequential problems

To the same problems above Add -sequential flag to PSD_PreProcess for sequential solver, but remember to use PSD_Solve_Seq instead of PSD_Solve. So the work flow for the 2D problem would be:

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

We solve the problem using the given mesh file bar.msh.

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

Similarly try out the 3D problem as well.

Comparing CPU time

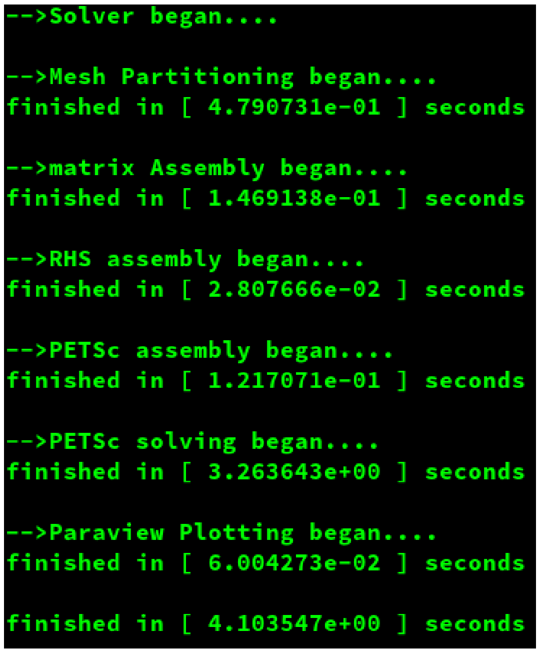
PSD provides mean to time log your solver via -timelog flag. What this will do when you run your solver, on the terminal you will have information printed on what is the amount of time taken by each step of your solver. Warning, this will make your solver slower, as this action involves MPIbarrier routines for correctly timing operation.

An example work flow of 2D solver 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
```



```
-->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 4: Time logging output produced for parallel run on 4 processes.

The figure~4 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.

Exercise 1

There is a solver run level flag for mesh refinement [1]. This flag is called -split [int] which splits the triangles (resp. tetrahedrons) of your mesh into four smaller triangles (resp. tetrahedrons). As such -split 2 will produce a mesh with 4 times the elements of the input mesh. Similarly, -split n where n is a positive integer produces 2^n times more elements than the input mesh. You are encouraged to use this -split flag to produce refined meshes and check, mesh convergence of a problem, computational time, etc. Use of parallel computing is recommended. You could try it out with PSD_Solve or PSD_Solve_Seq, for example:

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

for splitting each triangle of the mesh bar.msh into 4.

Exercise 2

There is a preprocess level flag -debug, which as the name suggests should be used for debug proposes by developers. However, this flag will activate OpenGL live visualization of the problems displacement field. You are encouraged to try it out

```
PSD_PreProcess -problem linear-elasticity -dimension 2 -bodyforceconditions 1 \
-dirichletconditions 1 -postprocess u -timelog -debug
```

Then to run the problem we need aditional -wg flag

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

[1]Mesh refinement is performed after partitioning.