Elastodynamics Tutorials - Advanced Exercises

Mohd Afeef Badri

Abstract

This document details some tutorials of elastodynamics module of PSD. These tutorials are not verbose, but does instead give a kick start to users/developers for using PSD's elastodynamics module.

Exercise 1

You are encouraged to try out timelogging and find out if the code (parallel/sequential) is any faster when we use Newmark-β or Generalized-α. Read the documentation for other types of time discretizations that can be performed with PSD, try each one out with -timelog and compare.

Exercise 2

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

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

Exercise 3

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 OpebGL live plotting of the problem, with displaced mesh. You are encouraged to try it out

```
PSD_PreProcess -dimension 2 -problem elastodynamics -dirichletconditions 1 -tractionconditions 1 \
-timediscretization newmark_beta -postprocess uav -timelog -debug
```

Then to run the problem we need aditional -wg flag

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

Exercise 4

PSD comes with additional set of plugins/functions that are highly optimized for performing certain operations during solving. These operations are handled by GoFast Plugins (GFP) kernel of PSD (optimize C++ classes/templates/structures), by default this functionality is turned off and not used. You are encouraged to try out using GFP functions in a solver by using -useGFP flag flag to PSD_PreProcess For example, the PSD solver workflow for the first 2D example in this tutorial would be:

[1] Mesh refinement is performed after partitioning.

```
PSD_PreProcess -dimension 2 -problem elastodynamics -dirichletconditions 1 -tractionconditions 1 \
-timediscretization newmark_beta -postprocess uav -useGFP
```

Once the step above has been performed, we solve the problem using, with the given mesh file bar-dynamic.

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

Try it out for other problems of this tutorial. -useGFP should lead to a faster solver, it might be a good idea to always use this option. To go one step further, use -timelog flag and determine if you have some speed up.