Linear Elasticity Tutorial Using Mfront-PSD interface

Mohd Afeef Badri

Abstract

This document details a tutorial that allows one to use PSD-MFront interface for linear elasticity problem. The same problem from tutorial 1 is repeated, however now MFront is used for building certain finite element essentials. It is advised to follow this tutorial after tutorial 1.
Note that, linear elasticity merely provides means of getting started with Mfornt, the real potential lies in using nonlinear materials and laws which Mfront provides. So this tutorial should be considered as baptism to the world of PSD-MFront which we believe has a lot of potential to solve some non trivial problems.

We reintroduce the problem from tutorial 1, an example of a 2D bar which bends under its own load – typical case of linear elasticity. 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 $\nu = 0.3$.

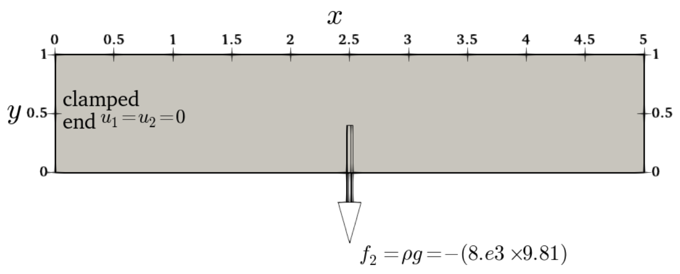

x
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
y
1
0.5
0
clamped end $u_1 = u_2 = 0$
$f_2 = \rho g = -(8.e3 \times 9.81)$

Figure 1: The 2D clamped bar problem.

Step 1: Preprocessing

First step in a PSD simulation is PSD preprocessing, at this step you tell PSD what kind of physics, boundary conditions, approximations, mesh, etc are you expecting to solve. More importantly for this tutorial we will signify to PSD that MFront has to be used.

In the terminal cd to the folder /home/PSD-tutorials/linear-elasticity. Launch PSD_PreProcess from the terminal, to do so run the following command.

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

After the PSD_PreProcess runs successfully you should see many .edp files in your current folder.

What do the arguments mean ?

- -problem linear_elasticity means that we are solving linear elasticity problem;
- -dimension 2 means it is a 2D simulation;
- -bodyforceconditions 1 with applied body force acting on the domain;
- -dirichletconditions 1 says we have one Dirichlet border;
- -postprocess u means we would like to have ParaView post processing files.
- -useMfront activates MFront interface for PSD.

At this stage the input properties E, ν can be mentioned in ControlParameters.edp, use E = 200.e9, and nu = 0.3. In contrast to tutorial 1, notice that these values of E and nu are fed to a vector PropertyValues = [E, nu]; verbosed by PropertyNames = "YoungModulus PoissonRatio";. We also signify that we will be solving linear elasticity via MforntMaterialBehaviour = "Elasticity"; and also MaterialHypothesis = "GENERALISEDPLANESTRAIN"; which signifies the hypothesis to be used for the Linear elasticity [1]. PropertyValues, PropertyNames, and MaterialHypothesis will eventually be provided to MFront in FemParameters.edp file via mfrontElasticityHandler(...) function [2]. The volumetric body force condition is mentioned in the same file via variable Fbc0Fy -78480.0, i.e $(\rho * g = 8.e3 * (-9.81) = -78480.0)$. One can also provide the mesh to be used in ControlParameters.edp, via ThName = "../Meshes/2D/bar.msh" (*note that mesh can also be provided in the next step*) .In addition variable Fbc0On 1 has to be provided in order to indicate the volume (region) for which the body force is acting, here 1 is the integer volume tag of the mesh. Dirichlet boundary conditions are also provided in ControlParameters.edp. To provide the clamped boundary condition the variables Dbc0On 2, Dbc0Ux 0., and Dbc0Uy 0. are used, which means for Dirichlet border 2 (Dbc0On 2) where 2 is the clamped border label of the mesh Dirichlet constrain is applied and Dbc0Ux 0., Dbc0Uy 0 i.e., the clamped end condition $(u_x = u_y = 0)$.

Step 2: Solving

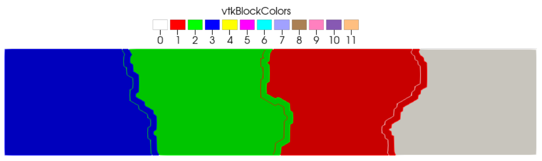
As PSD is a parallel solver, let us use 4 cores to solve the 2D bar case. To do so enter the following command:

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

Here -np 4 denote the argument used to enter the number of parallel processes (MPI processes) used while solving. -mesh ./../Meshes/2D/bar.msh is used to provide the mesh file to the solver. -v 0 denotes the verbosity level on screen. PSD_Solve is a wrapper around FreeFem++ or FreeFem++-mpi. Note that if your problem is large use more cores. PSD has been tested upto 13,000 parallel processes and problem sizes with billions of unknowns, surely you will now need that many for the 2D bar problem.

Step 3: Postprocessing

PSD allows postprocessing of results in ParaView. After the step 2 mentioned above finishes. Launch ParaView and have a look at the .pvd file in the VTUs... folder. Using ParaView for postprocessing the results that are provided in the VTUs... folder, results such as those shown in figure~2 can be extracted.



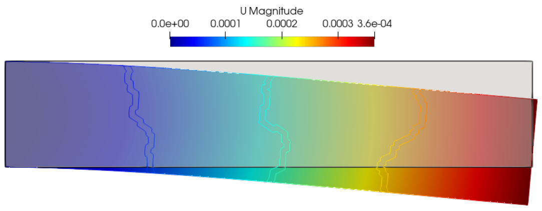


Figure 2: The 2D clamped bar problem: partitioned mesh and displacement field visualization in ParaView.

You are all done with your 2D linear-elasticty simulation with Mfront interface.

How and what is being done in PSD-MFront interface?

To explain how PSD-MFront interface works we will compare how a PSD solver acts when using MFront or without. In other words what is different when -useMfront is used at preprocessing. Note that ultimately the problem results (displacement fields, stresses, strains) will be the same.

To put it briefly, what MFront does for linear elasticity problem here is build the Material tensor (stiffness matrix) at each quadrature point. So, there are two points

[1]The MaterialHypothesis accepts "GENERALISEDPLANESTRAIN", "PLANESTRAIN", "PLANESTRESS", and "TRIDIMENSIONAL" as arguments.

[2]User is encouraged to have a look at FemParameters.edp file.

- We need to communicate to Mfront the nature of the problem and the material involved.
- We need to provide Mfornt the stiffness matrix at each quadrature point so that it can fill it up.

The two raised points are handled using mfrontElasticityHandler(...) in FemParameters.edp file.

Firstly, the arguments E = 200.e9, nu = 0.3, MforntMaterialBehaviour = "Elasticity";, PropertyValues = [E, nu];, PropertyNames = "YoungModulus PoissonRatio";, PropertyValues = [E, nu]; and MaterialHypothesis = "GENERALISEDPLANESTRAIN"; form ControlParameters.edp takes care of the first point (the nature of the problem and the material involved). The latter three arguments well define that we have a 2D problem, with given values of properties (E, ν). The snippet from ControlParameters.edp (produced after using -useMfront argument for PSD_PreProcess) file shows these variables which define the nature of the problem and characteristics of material involved

```
//==============================================================================
// -------Material parameters -------
// -----------------------------------------------------------------------
// E, nu : Modulus of Elasticity and Poisson ratio of the material
// PropertyNames : String of material property names (space seperated)
// that are provided to Mfront.
// PropertyValues : Values of material properties provided to Mfront
//
// -----------------------------------------------------------------------
// NOTE: Please note that PropertyNames should be the same as
// as in the Elasticity.mfront file
// -----------------------------------------------------------------------
//==============================================================================

 macro E() 200.e9 //
 macro nu() 0.3 //

 string MaterialBehaviour = "Elasticity";
 string MaterialHypothesis = "GENERALISEDPLANESTRAIN";
 string PropertyNames = "YoungModulus␣PoissonRatio";
 real[int] PropertyValues = [ E, nu ];
```

Secondly, the to get the stiffness matrix from Mfornt we use a quadrature finite element space with vector finite elements built on it (6 components) that represent the 6 components of symmetric material tensor ($\mathbb{R}^{3\times3}$). The snippet from MeshAndFeSpace.edp file shows the 6 component Quadrature finite element space for building material tensor.

```
//================================================================================
// -------The finite element space -------
// -------------------------------------------------------------------------------
// Qh : Quadratur finite element space for material tensor
// FEQF2 implies 3 dof for a triangular cell in the mesh
// A vectorial FEM space is built with 6 components
//================================================================================

fespace Qh ( Th ,[ FEQF2, FEQF2, FEQF2,
                       FEQF2, FEQF2,
                              FEQF2] );
```

Finally in file FemParameters.edp the mfrontElasticityHandler() is called to build the material tensor Mt provided with the previously built material properties and nature of problem. Please see the snippet below

```
//==============================================================================
// -------Material Tensor using Quadrature FE space -------
// -----------------------------------------------------------------------
// Mt[int] : is an array of finite element variable belonging to quadratu
// re space Qh. This array is used to define components of the
// material tensor. 3X3 in 2D and 6X6 in 3D
// In 2D the material tensor looks like
```

```
//
// [ 2*mu+lambda , lambda , 0 ] [ Mt11 , Mt12 , Mt13 ]
// Mt = [ lambda , 2*mu+lambda , 0 ] = [ Mt12 , Mt22 , Mt23 ]
// [ 0 , 0 , mu] [ Mt13 , Mt23 , Mt33 ]
//
// mfrontElasticityHandler : is a function in mfront interface that helps
// building the material tensor Mt given with
// material prpts. from ControlParameters.edp
//==============================================================================

 Qh [ Mt11 , Mt12 , Mt13 ,
             Mt22 , Mt23 ,
                    Mt33 ];


 mfrontElasticityHandler( MforntMaterialBehaviour ,
                          mfrontBehaviourHypothesis = MaterialHypothesis ,
                          mfrontPropertyNames = PropertyNames ,
                          mfrontPropertyValues = PropertyValues ,
                          mfrontMaterialTensor = Mt11[]
                        );
```

Note that in the snippet above you might be seeing Mt11[] being provided as mfrontMaterialTensor, in fact the Mt11[] calls the full matrial tensor not just the first component, so user should not get confused [3].

The material tensor Mt built is used in the finite element variational formulation to build the bilinear $a(\mathbf{u}, \mathbf{v})$ which is used to assemble the finite element matrix $\mathbf{A}$ for the linear system $\mathbf{Ax} = \mathbf{b}$

$$a(\mathbf{u}, \mathbf{v}) = \int_\Omega (\varepsilon\,(\mathbf{u}) : \mathbf{Mt} : \varepsilon(\mathbf{v}))$$

Here, $\mathbf{u} : \mathbf{Mt}$ is nothing but the stress $\sigma(\mathbf{u})$ operator. User is encourage to have a look at the VariationalFormulation.edp file that contains the variational formulation (weak form) of the problem described.

[3]This is more technical note, Mt11[] is the cast of Mt vector to a single array for memory optimization. One can also simply use Mt12[], Mt13[], Mt22[], ... all these are acceptable and are simply aliases to material tensor.