

# L-shape cracking

Mohd Afeef Badri

## Abstract

This document details a tutorial of ‘fracture mechanics’ module of PSD. This tutorial involves cracking of L shaped specimen, where loading is controlled by a point boundary condition.

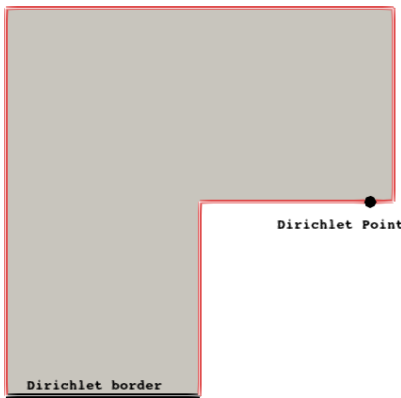


Figure 1: Geometry of the L-shaped test used in this tutorial.

## Preprocessing

You can either solve the problem using vectorial approach (recommended) or using staggered approach. To generate the solver use either from below.

### Generation of solver (vectorial)

```
1 PSD_PreProcess -dimension 2 -problem damage -model hybrid-phase-field \  
2 -dirichletconditions 1 -dirichletpointconditions 1 -debug -postprocess ud \  
3 -energydecomp -constrainHPF -vectorial -getreactionforce -plotreactionforce \  
4 -reactionforce variational-based
```

### Generating solver (staggered)

```
1 PSD_PreProcess -dimension 2 -problem damage -model hybrid-phase-field \  
2 -dirichletconditions 1 -dirichletpointconditions 1 -debug -postprocess ud \  
3 -energydecomp -constrainHPF -getreactionforce -plotreactionforce \  
4 -reactionforce variational-based
```

## Edit Cycle

### Edit ControlParameter.edp:

- Update physical parameter, change

```

1  real lambda = 121.15e3 ,
2    mu = 80.77e3 ,
3    Gc = 2.7 ;

```

to

```

1  real lambda = 6.16e3 ,
2    mu = 10.95e3 ,
3    Gc = 8.9e-2 ;

```

- Update solver parameter , change

```

1  real lfac = 2.0 ,
2    maxtr = 7e-3 ,
3    tr = 1e-5 ,
4    dtr = 1e-5 ,
5    lo ;

```

to

```

1  real lfac = 2.0 ,
2    maxtr = 1 ,
3    tr = 1e-2 ,
4    dtr = 1e-2 ,
5    lo ;

```

- Enter the correct Point boundary condition, change

```

1  real[int,int] PbcCord = [
2  //----- [ x , y ] -----//
3              [ 0. , 0. ] // point 0
4  //-----
5              ];
6
7  macro Pbc0Ux -0. //
8  macro Pbc0Uy -0. //

```

to

```

1  real[int,int] PbcCord = [
2  //----- [ x , y ] -----//
3              [ 470., 250. ] // point 0
4  //-----
5              ]
6  ;
7  macro Pbc0Uy tr //

```

### Edit LinearFormBuilderAndSolver.edp:

- To postprocess correct reaction forces in LinearFormBuilderAndSolver.edp for vectorial solver, change

```

1  for(int i=0; i < Th.nv; i++){
2    if(abs(Th(i).y-1.)<.000001){
3      forcetotx = forcetotx + F[][i*3]*DP[i*3];
4      forcetoty = forcetoty + F[][i*3+1]*DP[i*3+1];
5    }
6  }

```

to

```

1  if(mpirank==mpirankPCi[0]){
2      forcetotx = forcetotx + F[][PCi[0]*3+0]*DP[PCi[0]*3+0];
3      forcototy = forcototy + F[][PCi[0]*3+1]*DP[PCi[0]*3+1];
4  }

```

- To postprocess correct reaction forces in LinearFormBuilderAndSolver.edp for staggered solver, change

```

1  for(int i=0; i < Th.nv; i++){
2      if(abs(Th(i).y-1.)<.000001){
3          forcetotx = forcetotx + F[][i*2]*DP[i*2];
4          forcototy = forcototy + F[][i*2+1]*DP[i*2+1];
5      }
6  }

```

to

```

1  if(mpirank==mpirankPCi[0]){
2      forcetotx = forcetotx + F[][PCi[0]*2+0]*DP[PCi[0]*2+0];
3      forcototy = forcototy + F[][PCi[0]*2+1]*DP[PCi[0]*2+1];
4  }

```

- Finally to include cyclic loading, change

```

1  //--------------------------------updating traction-----//
2
3  tr += dtr;

```

to

```

1  //--------------------------------updating traction-----//
2
3  if(iterout<50)
4      tr += dtr;
5  if(iterout>=51 && iterout<110)
6      tr -= dtr;
7  if(iterout>=111)
8      tr += dtr;

```

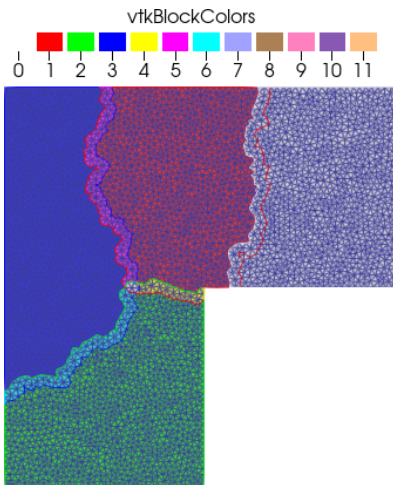


Figure 2: Finite element mesh of the L-shaped test.

## Solving

Irrespective of whether vectorial or staggered mode is used solve the problem using [PSD\\_Solve](#)

```
1 PSD_Solve -np 4 Main.edp -wg -v 0 -mesh ../Meshes/2D/L-shaped-crack.msh
```

## Postprocessing

Use ParaView to post process results.

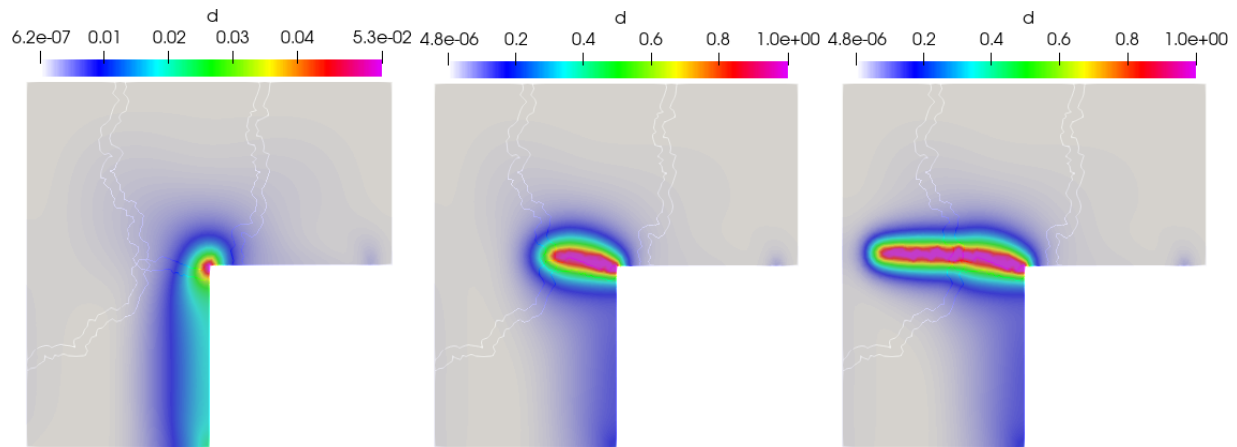


Figure 3: Finite element solution showing: Crack initiation, movement, and development.

On your screen, the force displacement curve which plots (force.data) should look something like this

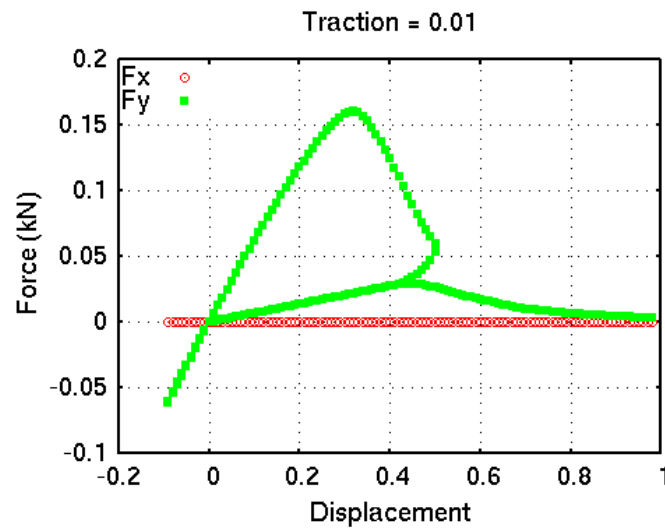


Figure 4: Force-displacement curve with cyclic loading.