PRECEPT 7

CEE 361-513: Introduction to Finite Element Methods

Monday Nov. 13

SECTION 1: FEniCS on Adroit

You must have an account with Adroit to perform the following steps. If you haven't registered for the account yet, you can do so by registering at `https://www.princeton.edu/researchcomputing/computational-hardware/adroit/`. We would be using miniconda for the setting up FEniCS.

1. Log into adroit by doing ssh netid@adroit.princeton.edu.
   Once into your home directory, perform:
2. wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh
3. bash Miniconda2-latest-Linux-x86_64.sh
4. export PATH="/home/netid/miniconda2/bin:$PATH"
5. conda config –add channels conda-forge
6. conda install fenics
7. conda install -f cmake=3.6.1

This should set up fenics for your account.
We would test, as done in precept, the installation using the fenics_truss.py code from course website. For that you would need to upload the files to cluster.
Uploading files:

1. Make a folder in your local machine, say adroit_upload
2. Transfer the python file to this folder.
3. In the terminal, perform
   rsync -rLvz ~/path_to_folder/adroit_upload/fenics_truss.py netid@adroit.princeton.edu:~/
   NOTE : This sends the file into your home directory. You can also set up a folder on cluster where you want the files to be uploaded.
   NOTE : Here we are only uploading a file. You can also upload directories.
4. You should now see a file fenics_truss.py in your home directory on cluster.

Next we test the files. Before testing we modify the code so that the plot is saved as an image which we can later download.

1. Change the line which says plt.show() to plt.savefig('Exact and FEM solution.png')
2. Add plt.close()

This would save the plot as an image which we can download and later view.

Next you can run the code as you do in your terminal by typing `python fenics_truss.py`.

This was possible to be do since the code was short and we were testing the code. In general you should submit the job to cluster by following the bash script at :
`https://www.princeton.edu/researchcomputing/education/online-tutorials/getting-started/running-serial-j`

Once you have the results you can download the files by performing:
NOTE: Assuming that you saved the files in a folder called fenics

1. rsync -rLvz netid@adroit.princeton.edu:~/fenics /path_to_folder/home_machine
   NOTE : path_to_folder/home_machine refers to the directory you want to save the files in your home machine.

SECTION 2: GMSH

The goal is to create a mesh for the section shown below using GMSH and import it into FEniCS.

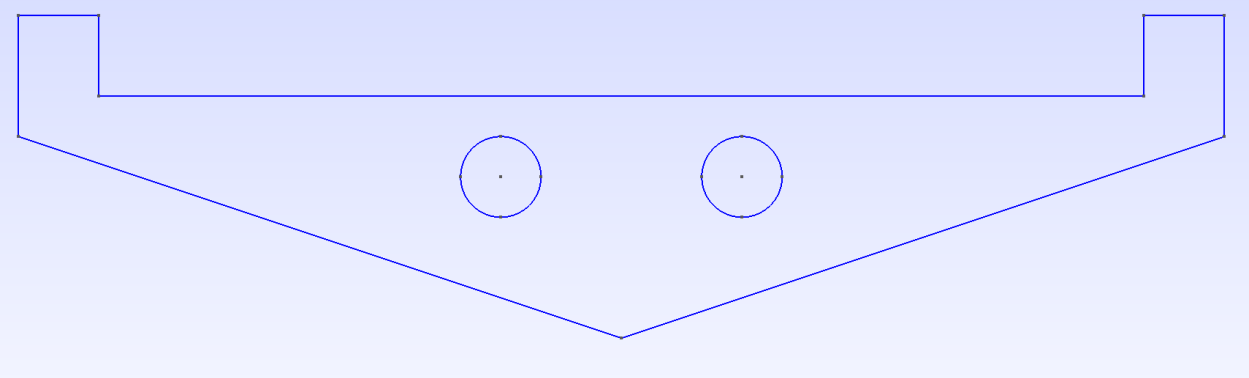

Figure 1: The section geometry

The following steps should be followed:

1. Open GMSH
2. From the right panel, select Geometry → Elementary entities → Add → Point
3. Add the following corner points [(0.0,0.0), (1.5,0.5), (1.5,0.8) , (1.3,0.8), (1.3, 0.6), (-1.3, 0.6) , (-1.3,0.8), (-1.5,0.8), (-1.5, 0.5)] of the section. Add the center and four points on the perimeter for the holes [(0.3,0.4), (0.3, 0.5), (0.3,0.3), (0.4,0.4), (0.2, 0.4)] and [(-0.3,0.4), (-0.3, 0.5), (-0.3,0.3), (-0.4,0.4), (-0.2, 0.4)]

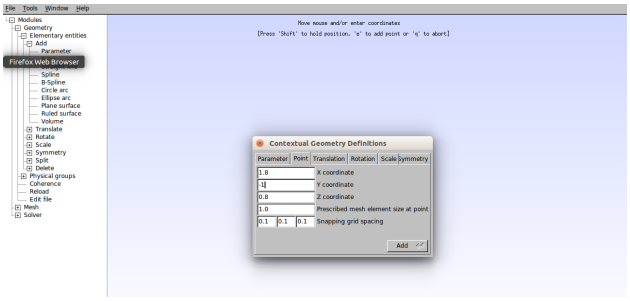

Figure 2: Adding the points

4. Click on Add → Straight Line. Then connect the points as lines

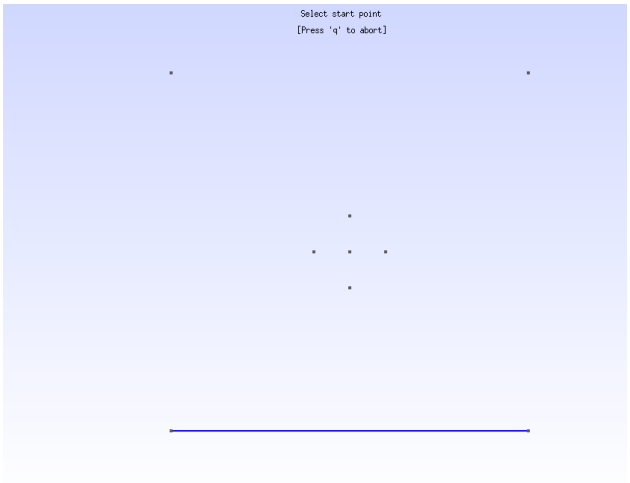

Figure 3: Adding the lines

5. Next click on Circle arc, and connect the points in the center to create circular arcs.
6. The initial geometry looks like the following.

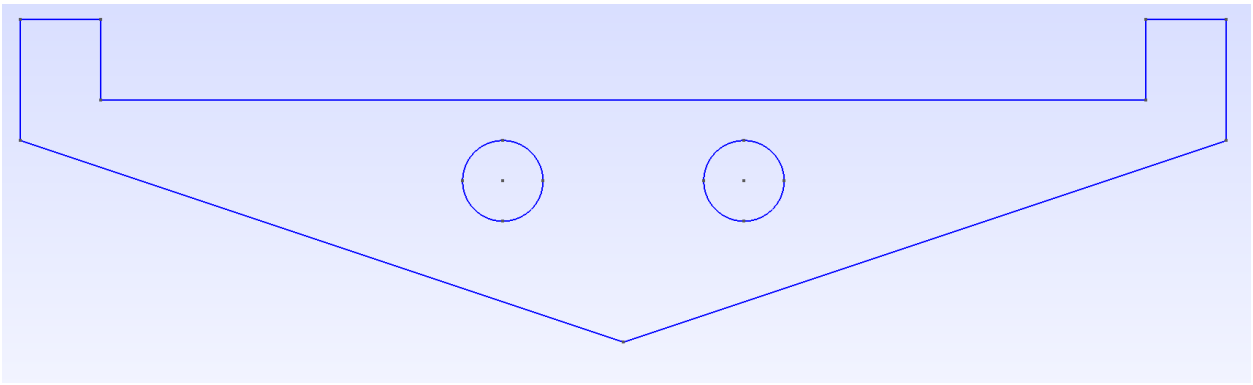

Figure 4: Initial geometry

7. Next click on Plane Surface from the same panel and select the outer boundaries and the holes when directed. This should result in the following:

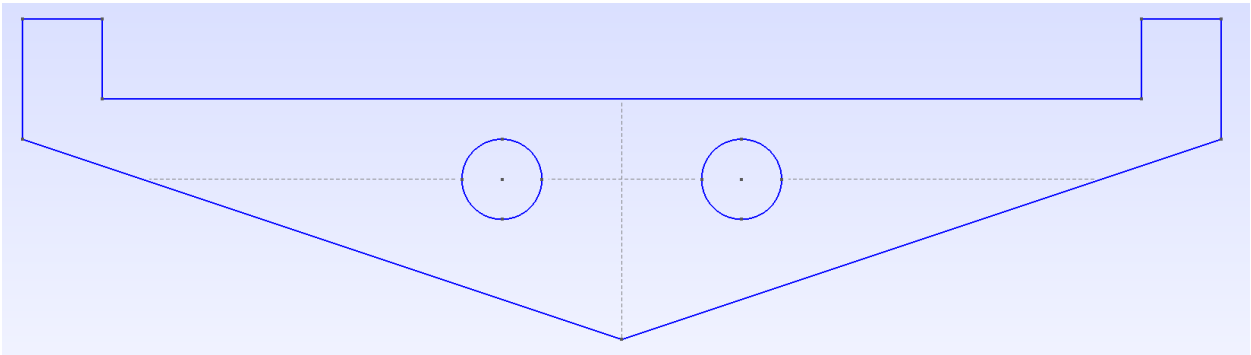

Figure 5: Initial geometry with surface lines

8. Click on Physical groups under Elementary entities and select Line. Click on all the lines in the geometry
9. Next select Surface under Physical groups and click on the surface.
10. Finally click Mesh → 2D and you should have obtained a coarse mesh as below:

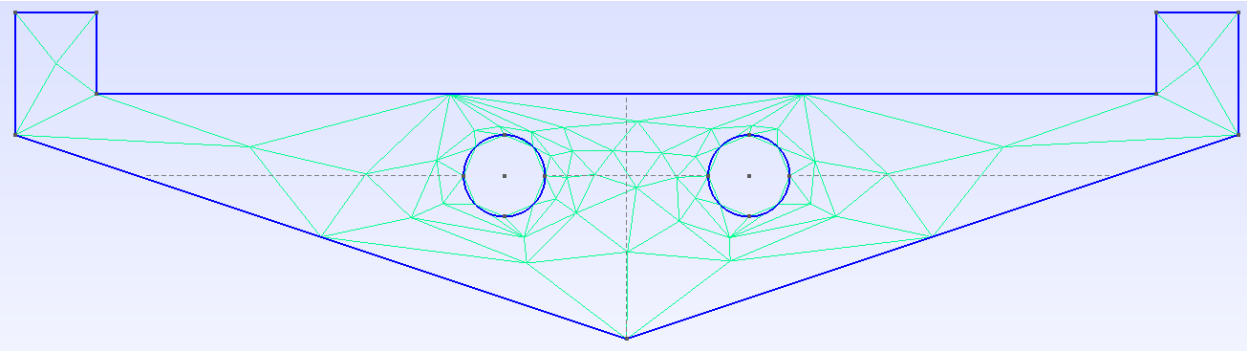

Figure 6: Coarse Triangular Mesh

11. We can refine the mesh by selecting Refine by splitting to obtain a refined mesh.

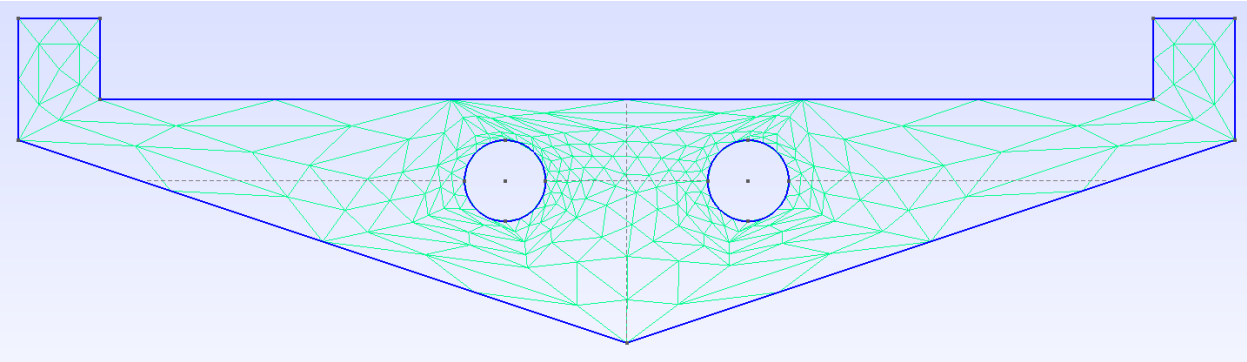

Figure 7: Refined Triangular Mesh

12. Lastly you can save the mesh

Once the mesh is saved we need to import it in FEniCS as was done in previous precept

1. Let us say the mesh was saved as bridge_section.msh
2. In terminal, perform:
   - `python`
   - `from dolfin import *`
   - `import os`
   - `os.system('dolfin-convert bridge_section.msh bridge_section.xml')`
     This should create a mesh which we can import to dolfin as:
   - mesh = Mesh("square_hole.xml")

SECTION 3: Tensor Review

Here we review a few results from our lessons on Tensors.
Gradients:

1. $\nabla f = \dfrac{\partial f}{\partial x_i}\mathbf{e}_i$
2. $\nabla \boldsymbol{u} = \dfrac{\partial \boldsymbol{u}}{\partial x_i} \otimes \mathbf{e}_i$
3. $\nabla \boldsymbol{\sigma} = \dfrac{\partial \boldsymbol{\sigma}}{\partial x_i} \otimes \mathbf{e}_i$

Divergence:

1. $\nabla \cdot \boldsymbol{u} = \dfrac{\partial u_j \mathbf{e}_j}{\partial x_i} \cdot \mathbf{e}_i = \dfrac{\partial u_i}{\partial x_i}$

Laplacian:

1. $\Delta f = \nabla \cdot (\nabla f)$

We briefly review the integral theorems:

1. Integration by parts. We have extensively used this so far. Let $f(x)$ and $g(x)$ be two scalar functions in 1-D

$$\int_a^b \frac{df}{dx} g \; dx = [f(x)g(x)]|_a^b - \int_a^b f \frac{dg}{dx} \; dx$$

2. Gauss' Theorem
   Let $\boldsymbol{u}$ be a vector defined on the domain Ω with the boundary represented as Γ. Let the normal to boundary be denoted by n. Then the Gauss' theorem states:

$$\int_\Omega \nabla \cdot \boldsymbol{u} \; dV = \int_\Gamma \boldsymbol{u} \cdot n \; dS$$

   where dS is the differential element on the boundary. Now let η be a scalar function. Then let us evaluate the following:

$$\int_\Omega \nabla \cdot \boldsymbol{u}\eta \; dV$$

   We know:

$$\begin{aligned} \nabla \cdot (\boldsymbol{u}\eta) &= \frac{\partial \eta}{\partial x_i}\mathbf{e}_i \cdot \boldsymbol{u} + \frac{\partial \boldsymbol{u}}{\partial x_i} \cdot \mathbf{e}_i \eta \\ &= \nabla\eta \cdot \boldsymbol{u} + (\nabla \cdot \boldsymbol{u})\eta \end{aligned}$$

   Hence:

$$(\nabla \cdot \boldsymbol{u})\eta = \nabla \cdot (\boldsymbol{u}\eta) - \nabla\eta \cdot \boldsymbol{u}$$

   Substituting we obtain:

$$\begin{aligned} \int_\Omega \nabla \cdot \boldsymbol{u}\eta \; dV &= \int_\Omega \nabla \cdot (\boldsymbol{u}\eta) \; dV - \int_\Omega \nabla\eta \cdot \boldsymbol{u} \; dV \\ &= \int_\Gamma (\boldsymbol{u}\eta) \cdot n \; dS - \int_\Omega \nabla\eta \cdot \boldsymbol{u} \; dV \end{aligned}$$

With the above review we can now derive the weak form for the Poisson's equation:
Find u such that on the domain Ω:

$$\begin{aligned} \Delta u &= f \quad \forall x \in \Omega \\ s.t\ u &= g \quad \forall x \in \Gamma_D \\ \nabla u \cdot \boldsymbol{n} &= h \quad \forall x \in \Gamma_D \end{aligned}$$

where $\boldsymbol{n}$ is the normal vector to the boundary. The set of trial and test functions are:

$$\mathcal{S} = \{u | u \in H^1(\Omega), u = g \text{on} \Gamma_D\}$$

$$\mathcal{V} = \{v | v \in H^1(\Omega), v = 0 \text{on} \Gamma_D\}$$

The residual is:

$$R(x) = \Delta u - f$$

Multiplying by test function and integrating over the domain:

$$\begin{aligned} &\int_\Omega (\Delta u - f) v d\Omega = 0 \\ &\int_\Omega (\nabla \cdot (\nabla u) v) - f v d\Omega = 0 \\ &\int_\Omega (\nabla \cdot (\nabla u v) - \nabla u \cdot \nabla v) - f v d\Omega = 0 \\ &\int_\Gamma \nabla u v \cdot \boldsymbol{n}\ d\Gamma - \left(\int_\Omega \nabla u \cdot \nabla v + f v\ d\Omega \right) = 0 \end{aligned}$$