The assignment required that a program be written to solve the projects finite element problem, however several simplifications had to be done to the original goal of modeling a gasket between a flange. Ultimately it was decided to model a two-dimensional cross section of a flange, gasket and bolt, and only model the pressure of the bolt. In addition to being able to more directly leverage techniques taught in the course, it will provide an interesting comparison to the more thorough and complex model of Abaqus.

From a software architecture point of view, a finite element solver program was written was tailored for this specific simplified flange problem in MATLAB. Although much of the code written is specific to this problem, there was a goal to make the code applicable to as many FEM problems as possible. An object-oriented approach was chosen in an attempt to keep as much of the program as general. Often a class would wrap a more primitive MATLAB construct, such as the GlobalStiffnessMatrix type wrapping a sparse matrix. These helper types would also include functions to handle common operations, such as an AddLocalStiffnessMatrix function on the GlobalStiffnessMatrix. Different element types were developed, however only the base Element type and the Triangular3Node2DElement classes were ultimately used for this problem.

The two-dimensional cross section of the flange, gasket and bolt were originally exported from Abaqus, and from that a mesh was created in GMSH. There were several issues with this process. First, the size of the element out of GMSH was 1/100th of what it was in Abaqus. This was corrected in the MATLAB code where, when the mesh was read in, all of the node coordinates were multiplied by 100. Also the export process did not take into account any curves and fillets on the model.

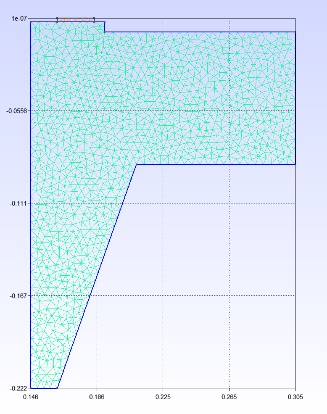


Figure : Mesh for Finite Element Program

One final issue was that the MATLAB FEM solver was not able to enforce continuity between the flange and the gasket. As such, the gasket was ignored in the analysis.

For simplicity, 3 node triangular elements were used for the mesh, with a maximum size of 0.005. Combined with the material properties used in Abaqus, the local stiffness matrices were derived. Instead of modeling the contact stresses of the bolts, equivalent forces were calculated with equations and values from the manufacture of the flange:

<TODO actual pretty equations>

Equivalent Pressure = (4\*BoltForce/EffectiveGasketDiameter \*\*2+ 16 \* Bending Moment on Flange/EffectiveGasketDiameter\*\*3) / 6894.7

Equivalent Force = Equivalent Pressure \* span of bolt forces

With the magnitudes of the forces evaluated, they were applied to the node closest to the center of each area the pressure was being applied. The global stiffness matrix wrapped an instance of a sparse matrix that is built into MATLAB, and the global load vector was assembled. For this problem, the pressure was modeled as a boundary condition on the left side of the flange preventing the flange from moving in the X direction. To account for symmetry the top of the gasket was fixed in the y direction. The boundary conditions were hard-coded into the program, eliminating rows and columns for node elements that were fixed or where the axis of symmetry cut the cross section. The displacements of the remaining nodes was found by inverting the remainder of the global stiffness matrix and multiplying it with the load vector that remained after eliminating rows due to boundary conditions. The reaction stresses and strains where then computed and plots of the stress and displacements were generated.

In order to model the ramp up of the pressure, the main algorithm of evaluating displacements was done in a loop where the pressure was increased in increments of 1/10 of the maximum load. The results for the maximum loading were reasonably close to the Abaqus results, being only 5.4% less than the displacements evaluated in Abaqus:

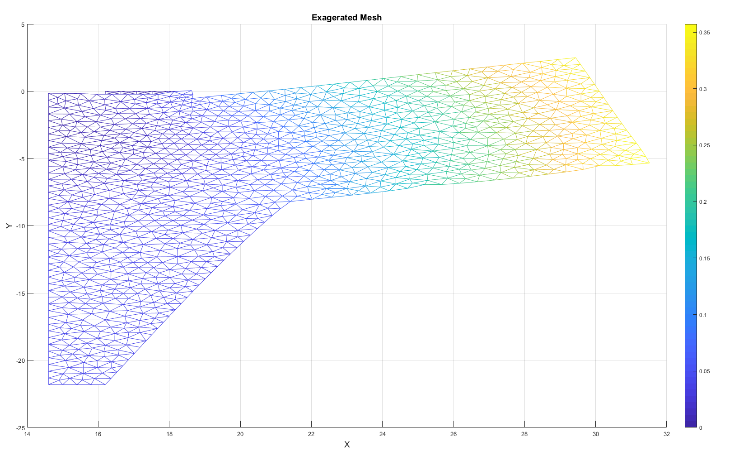


Figure : Mesh under load, displacements exaggerated by a factor of 100

The stress forces on the flange had a similar distribution as the Abaqus model, but were different in their value.