

CIV H194

Report

By Yara Mubarak

Professor Sanjay Govindjee

Ahmed Bakhaty

Preface :

The goal of this course was primarily to help Phd candidate Ahmed Bakhaty on his finite element model of a heart valve. The prerequisite knowledge required to be able to contribute to this project was mainly basic mechanics and sufficient knowledge of the Finite Element Method, specifically including non-linear materials and finite deformation. Then regarding the specifics of this project, I was to learn how to work with feap (the finite element program being used already). I was also required to work with some meshing and modeling programs that are needed in order to run the specifics I was tasked with.

This report will first start with an overview of what I was tasked with. It will then describe how I gained the sufficient knowledge needed to work on the task under the context of this project. After that, it will describe the work done, discussing in detail the challenges faced and how they were overcome. The results will then be showed, with future work emphasized , as well as useful tools created while in the process of accomplishing this task.

Introduction:

The main task I was given was to produce a working model of the heart valve in full geometry as shown in Figure ###. I was not asked to take into account the material constraints, chemical reactions, and the embedded FEA model aspects of this model. Moreover, I was given a stereolithography model of the heart valve with the goal of being able to run FEA on them, either non-linear or linear. Of course, the first step was to be able to produce a working simple linear FEA model.

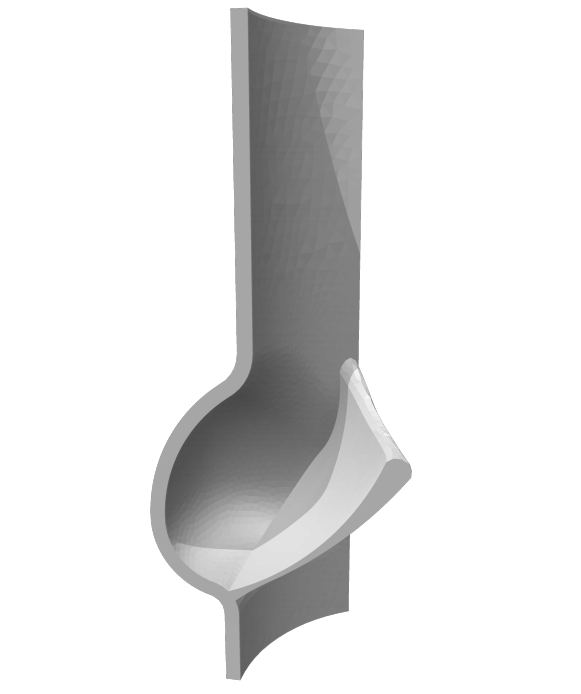


Figure ### : Image of the Heart Valve After Symmetrical Cutting

As seen in the figure above, symmetry was exploited in order to run the most efficient algorithms on this model. The full heart valve was first separated into 3 parts and then each of these parts was also cut in half. Figure ### shows how that is possible.

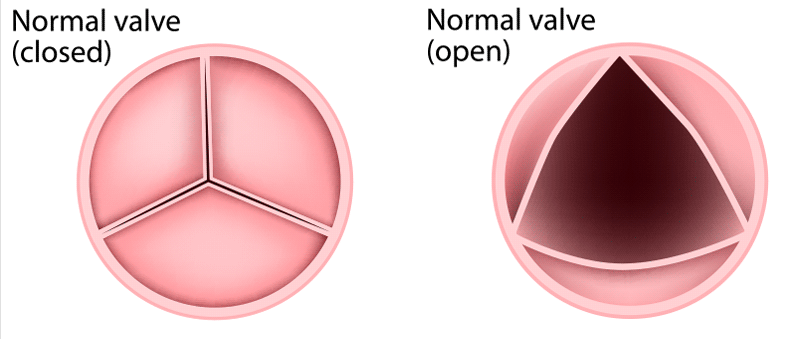


Figure ### : Heart Valve 2D Slice [[1]](#footnote-1)

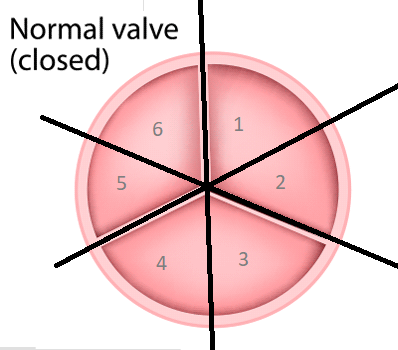


Figure ### : Symmetrically Cut Closed Valve

The flap you see in the 3D model in Figure ### is just the part that opens and closes as seen in the 2D slice. Then the walls around it are just the walls of the heart valve as cut in any of the parts numbered 1-6 in the figure above.

# Gathering the Appropriate Knowledge :

As an undergraduate mechanical engineer, I had sufficient knowledge of mechanics and some background in continuum mechanics. Therefore the main idea of this was to get acquainted with the finite element model and specifically how it could be applied to this project.

## Translating the CE130N labs:

At first I started with a basic linear finite element problem. Following Professor Sanjay’s instructions to translate the CE130N lab 7 and lab 10 from matlab to python, I learned the basic structure of a linear FEA problem in code. Although I was reading some material about FEA on the side, I did not get the specifics. I, however, observed a common pattern. First, one must specify the mesh properties (eg. Number of elements or shape of elements) , the material properties (eg. Nu, Young’s) and the geometry (eg . length , width). Then one must initialize a load array for each node the size of n by 1 where n is the number of global nodes. There was also something that that resembled a stiffness matrix for each of the nodes of size d\*n by d\*n where d is the dimensions of the problem (eg. x, y , and maybe an angle makes for 3 dimensions ), and n is the number of global nodes. Then there was a looping over the number of elements to create the appropriate load and stiffness for each of the nodes, understanding that there was some method to discretize our continuous space that depended on the size and shape of our elements as well as the type of extrapolation between them. There was a local matrix k and f made for every loop, then they were strategically placed into the global K and F matrices where K is what I perceived at that time to be a sort of stiffness tensor and F was the array of forces at each node.

The method of the solution was very interesting after all the K and F was set up. Since we had some nodes that had free forces and some nodes that had specified displacements, then the Kx = F equation we had to be split up in order to appropriately account for all the constraints and be solved. Here, x would be the variables we were trying to solve for, an array of d\*n by 1 where d is the dimensions of variables we were try to solve for each node, n. According to CE130N vocabulary, they were split by driven and free parts, driven were the parts of the F and K matrix that were constrained and the free were the opposite. We first solved for the driven Force and known displacement to get the free displacement. The next step was to solve for the force or reactions on the constrained displacement.

I learned more practical application than theory from this translation from matlab to python but that was very useful for my next step.

## MATLAB linear learning task:

After understanding how to code a simple linear FEA problem, I was given basic material on FEA to understand more of what I did alongside a task to write my own Finite Element solver from scratch. This program had to be later adaptable to a non-linear version of itself, therefore I couldn’t just write something as simple as the labs I had translated.

I first started reading the CE222 course reader by Professor Khalid M. Mosalam , which Ahmed had gave me along a few other useful readers. He specified the chapters that would help the most in the construction of this solver.

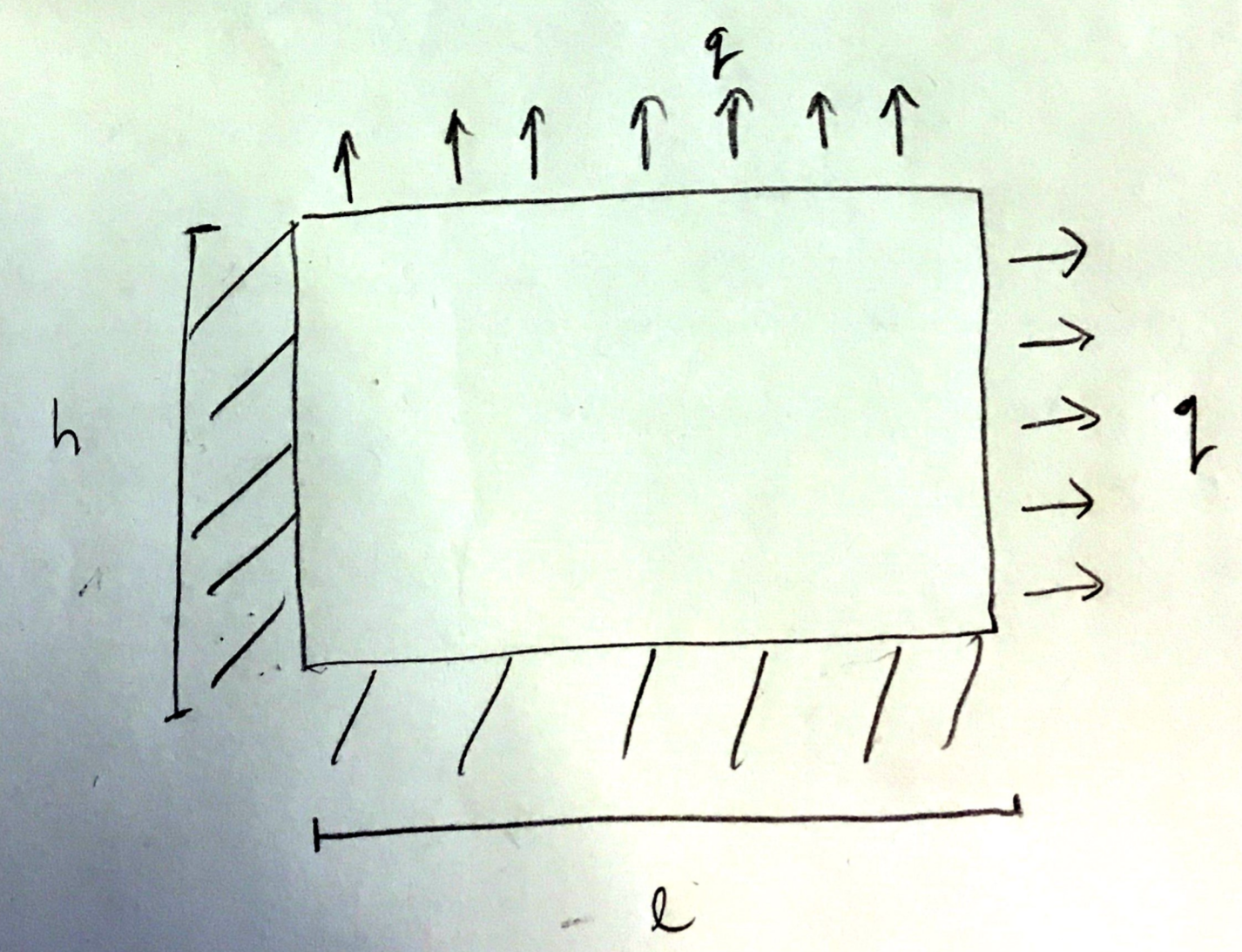
The first problem to solve was a basic biaxial and plane stress case. The problem looks like this :   


Figure ### : Plane, Biaxial 2d and Stress Problem

First step was to write the mesh generator given a number of elements and the length and the height of this box. This mesh generator had to output numbered element’s coordinates and then how they were connected. We thought it would be best to have rectangular elements in this case. That was a fairly easy task with no learning curve.

The next step was to understand what I had thought were interpolating functions while working on the labs. Through this process I learned what shape functions are. After some reading, I understood that shape functions were a way to discretize certain spaces. In FEA there are many shape functions one can have depending on the sort of material, deformation expected, and geometry of the object undergoing stress. Ahmed explained to me that we were going to rely on simple linear shape functions for our heart valve problem since anything above that would unreasonably add to the number of mesh nodes we had. That would increase our computation time significantly and might have adverse effects of allowing for deformations that the experiment we are testing against does not support. Following this discussion, I concluded that going with the traditional hat functions for my 2D box problem would be suitable and doing anything else would not be beneficial for learning at this point.

The Stiffness and force matrices came next. Those were challenging to understand mathematically. However, conceptually understanding the weak form was very interesting and provided a new mindset of how one might extrapolate thinking of averaging approach to a problem with multi dimensions. After Ahmed went over the math with me a few times and along with my readings , I was able to reach a comfortable understanding of the linear stiffness matrix constructed as such.

D was for a plane stress linear and isotropic material. B was constructed by mapping back and forth to a natural coordinate system in order to create B robustly with all sorts of elements and our hat shape functions. Then for our integration, a standard gauss integration was performed on each element to get the local stiffness matrix, k.

The local force array was constructed simply for this problem since our shape functions were simple and our tractions were constantly normal to our object. Since I understood how force could be set up, I could modify it in the future if need be. I understood the force array to be a discretized version of tractions using our shape functions with the following equation:

Where N are the shape functions, t are the tractions and a is the surface area of our object.

After understanding how the construction of the local force and stiffness matrices, the compilation of them was standard having done it in the labs before.

The results of my linear MATLAB algorithm was compared to the same problem in feap that Ahmed wrote. There were a few bugs in how I constructed my B matrix and a few issues with setting up the gauss point integration. However after a sitting with Ahmed and going back and forth with the reader to confirm that I had my equations correct with debugging. We were finally able to get the same results. You can see an example of the results below:



Figure ###: Displacement on Upper Boundary of Block



Figure ###: Displacement on Right Boundary of Block

As seen above, these are consistent with what is expected of this biaxial problem and they match feap’s output. For further details about this code refer to the index.

## MATLAB non-linear learning task:

As mentioned before, the MATLAB solver I wrote had to be adapted to solve for non-linear version of the box problem.

After reading some more material, I was to decide a few things about this solver. First, in what reference frame would this solver work. Since I had some background in continuum mechanics, I thought I would work with what I find to be the most visualizeable and understandable frames, the current. Second, what non-linear material would be used to get the free energy equation. A simple neo-hookian material seemed to be the most simple to code. It was also modified for the plane stress case so that the thickness always stays constant. Third, what iteration method to use as a final step when solving for the Residual. The Newton Rhapson method was also being used for the heart valve. It is also the base for many other complicated iterators.

From the basic linear weak form that I had learned while writing the linear solver, I generalized to form the final equations.

The equation instead of being independent of u is now a function of it. We create that function from our geometric equations and our free energy equation.

For notation’s purpose and literature convention we call this function R.

After a Taylor expansion we are able to put

INDEX

Insert k41element

Element

Mesh

Material

1. http://gccsiinc.com/procedures/minimally-invasive-heart-surgery/mitral-valve-repair-and-replacement/ [↑](#footnote-ref-1)