PLASMA

Limits and Stresses Mesh Analysis

New Mexico Adventures in Supercomputing Challenge Final Report

April 7, 2004

Team 008
Albuquerque Academy

Team Members
Jim Adolf
Zach LaBry
Josh Langsfeld
Ryan McGowan
Matt Strange

Teacher
Dr. Jim Mims

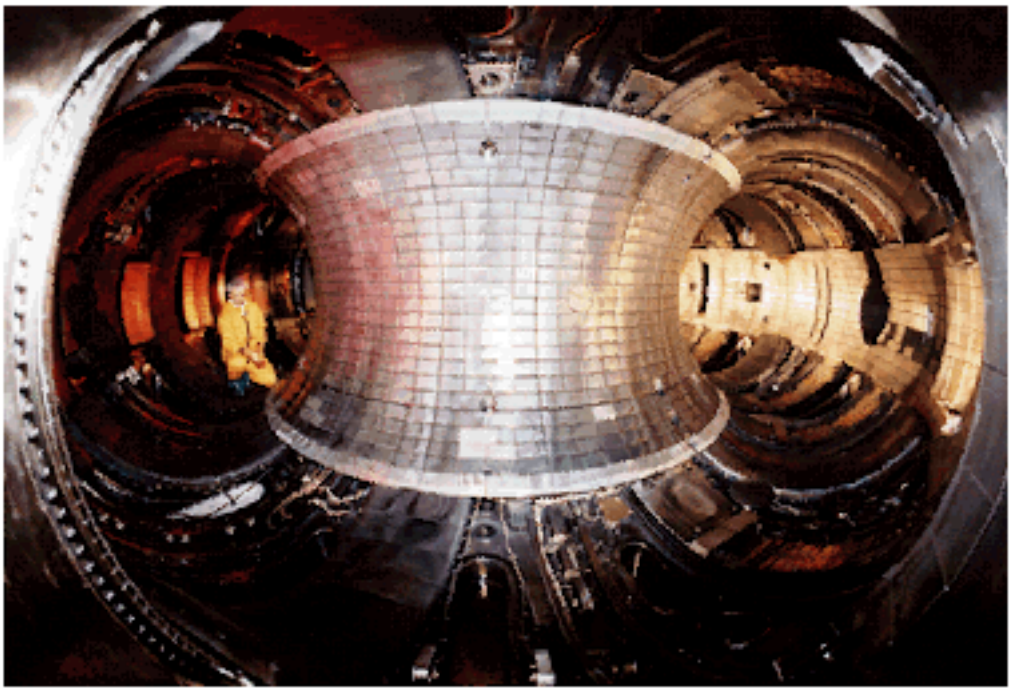
Executive Summary

This year, our project involves optimizing the shape of a fusion tokamak to provide the best results in a fusion experiment. There are two primary problems with using fusion as a power source. The first is that it is difficult to sustain, and the second is that it is difficult to contain. The purpose of this project is to solve the latter problem.

There are three major parts of this project that we had to work on over the course of this year. First, we used the open source library LibMesh to create a model of the geometry of the torus. The parameters of the torus include the size and eccentricity of the cross-section, which can be modified during optimization. Once this model is in existence we use our physics model to apply forces to the walls of the torus, simulating the stresses associated with fusion. In this part, we model the plasma inside the torus as numerous charged particles and forces applied to the walls come from collisions with the particles.

When the program has calculated the forces on the torus, we run them through the finite element analysis built into LibMesh, which returns a fitness value for the torus based on the magnitude and relative intensity of the stresses. Using our optimization algorithms of simulated annealing and a genetic algorithm, the parameters of the torus are adjusted towards a torus with a better fitness value. This process is repeated until the program reaches the best possible torus available within the parameters.

Finally, providing a easy way for the user to visualize both the shape of the torus and the stresses put on it is essential to creating an effective program. Here we use the free program General Mesh Viewer to see both these things. LibMesh has the ability to export its data directly to GMV, which makes it an ideal choice for visualizing our results.



A picture of the inside of an actual tokamak

Problem Statement

Our goal is to solve the problem of which geometry is the best for containing a fusion reaction inside of a fusion reactor. We are writing a computer program that has four major components in order to solve this problem. First, it models the physical geometry of the container and the stresses on the container due to the plasma. Second, it models the movement and forces on the plasma due to the magnetic fields produced by the Tokamak, and uses this to generate the stresses used in the previous module. Third, it uses optimization algorithms to minimize the stresses on the container by modifying a set of design factors. Lastly, it provides a graphical representation of our data, which allows anybody to understand our results easily and intuitively.

Finite Element Method

The choice of using the finite element method over the other available methods was based on several factors. The first of these factors is suitability. The finite element method is clearly suitable for use in our particular application. The finite element method breaks the geometry model down into a discretized version of the real physical system, and then analyzes stresses and displacement at certain nodes on the elements (the discrete members of the system). Our particular application is to evaluate how our Tokamak holds up under the stress and strain of the plasma forces inside the container.

The second reason for choosing the finite element method was efficiency. Although any method used to analyze a physical system such as this one will certainly run into problems of low efficiency, the finite element method makes this problem more manageable by the fact that the efficiency is directly related to the number of elements that are being used, that is, it is directly related to how fine the mesh is that is modeling the system.

Another reason for the use of the finite element method is the availability of suitable libraries. Building a finite element library from scratch, or a library for any other method suitable for this problem would be a very time-consuming, and redundant task. Libraries already exist for such analyses, and these libraries have already undergone development and debugging. So, to capitalize on this, we chose to use the LibMesh open-source library for finite element analysis. Use of this library (or a similar library) prevents the creation of redundant code, and increases reliability. Now, we need only verify the accuracy of our own physics model; we do not also need to verify the validity of the method we are using to analyze that model. This saves a significant amount of time, and allows us to divert our attention elsewhere.

Plasma Model

Methodology

The method chosen to model the plasma interactions with the boundary of the containment vessel, both in terms of thermal stress and mechanical stress, is an n-body simulation. The n-body simulation models individual particles, or groups of particles, and calculates the force upon each object imposed by its nearby neighbors. The forces under which the particles interact are electromagnetic forces. Each particle contains a mass, a velocity, and a charge. As the electromagnetic force drops off as a square of the distance, and the interior of the containment vessel, the tokamak, shields particles from direct interactions with particles significantly distant from it.

The n-body simulation is bounded by the geometry of the interior of a torus. As the velocity of each particle is known, as well as its mass, the kinetic energy of each particle can be computed, and hence, the temperature, that is, the average kinetic energy, can be computed. For the mechanical stresses, the collisions of the plasma particles with the boundary are computed. Whenever a particle tries to leaves the boundary, it is reflected back by the surface of the container. The result is a significant change in momentum, which represents a force being exerted on the container. As the time interval is known and fixed, and the change in momentum can be calculated, the force exerted on that part of the surface, which is represented by a face of the mesh could be calculated.

The vertices of the boundary mesh correspond to the nodes of the finite elements. This enables us to readily transfer the mechanical forces from the plasma model to the finite element mesh.

Implementation

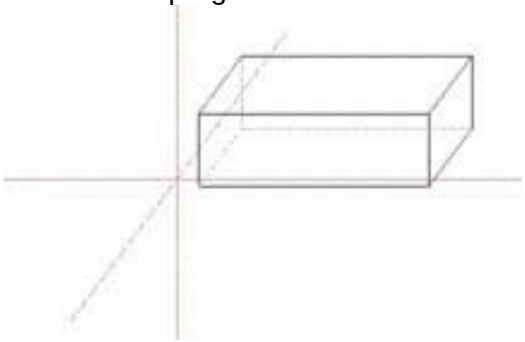
The plasma model is implemented in C++ using the Blitz++ linear algebra library. This library provides for fast computation of vector operations within the system. There are two major difficulties in this implementation: ensuring numerical integrity, and collision detection with the mesh faces.

The issue of numerical integrity comes with the finite precision representation of floating-point numbers on the machine. Floating-point numbers are represented with a mantissa and an exponent. For numbers of significantly differing magnitude, the addition of a large number and a small number may yield simply the large number with no change. Without proper care taken while adding floating-point numbers, this can result in numbers that are significantly incorrect (for example, taking a single large number, and then adding its equivalent in the form of many small numbers, instead of doubling the original, will leave it unchanged). Taking a list of floating-point numbers, and summing them until the sum becomes too large to add successive numbers, storing the large number in a list, continuing until the entire list has been converted into larger numbers, and then repeating this process, has corrected for this error.

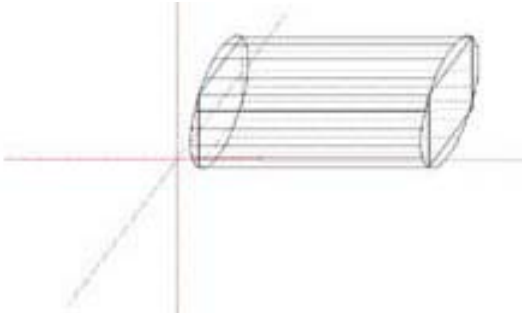
The second issue is a bit more complex to deal with. The mesh must be represented so that panels, groups of adjacent vertices know about their neighbors, so that the angle of reflection of the particles can be properly computed.

Mesh Implementation

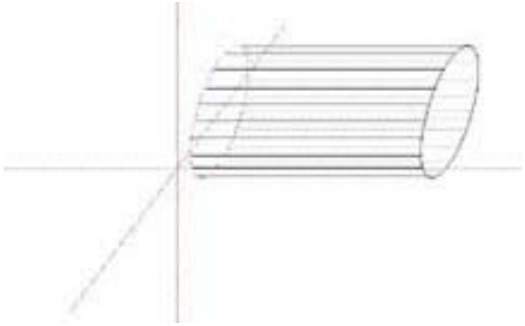
The torus is build by expanding on primordial three-dimensional shapes. The program begins with a connected cube object, which extends a previously written cube to allow for linked faces. This allows the cube to become a full wraparound torus without gaps in its structure. The program creates this connected cube and places it in space.



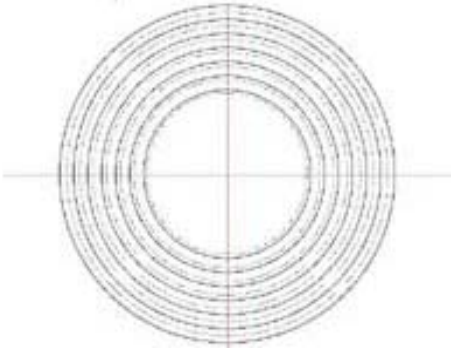
The program then circumscribes an imaginary cylinder around the connected cube.



A number of iterations move the nodes of the connected cube to the position of the imaginary cylinder, thus expanding the connected cube into a cylinder. This makes a cylinder with connectable end faces.



The cylinder nodes are then scaled to the geometry of a circle. This makes a three-dimensional torus with connected endpoints, and therefore an unflawed geometry.



Optimization Algorithms

Both optimization algorithms-the genetic algorithm and the simulated annealing algorithm- will optimize by minimizing a function in three variables. The three variables are: cross sectional horizontal axis, cross sectional axial ratio, toroidal major axis, and the equation to minimize is:

$$F(csha, csar, tma, te) =$$
$$a \cdot SurfaceArea(csha, csar, tma) +$$
$$b \cdot Volume(csha, csar, tma) +$$
$$c \cdot Stress(csha, csar, tma) +$$

Genetic Algorithm

Genetic Algorithms (GA's) are based on the idea of the evolutionary process, and the idea of "survival of the fittest". The GA starts with some set of initial solutions, which could have been generated randomly or by some other process, represented by some set of data called a chromosome. This population of initial data members are combined and crossed to create new solutions, hopefully with better characteristics than their 'parents'. Individual population members are chosen to be parents according to how well they solve the problem: their fitness. The better they solve the equation, the more likely they are to breed. An important fact to remember is that genetic algorithms always contain randomness, and this tends to prevent getting stuck in a local best. This process of creating a new generation is repeated until a solution is reached that is good enough. This does not guarantee the best solution, but it does find a very good solution faster than most other algorithms.

Our particular GA uses a steady state algorithm to ensure that only the best data is kept, and members of the new generation only replace those in the older if the new member is better than an older member. It uses a large search space, diverse and extensive population, and a greedy crossover function to maximize both the efficacy and speed of our GA.

Simulated Annealing

Simulated annealing is another type of optimization algorithm inspired by the process of annealing in metallurgy. Solutions are modeled as a cooling metal, in which the total internal energy is minimized, thus reaching the best solution. The algorithm starts at some given point; we are starting with a perfect torus. During each step, it chooses a 'neighbor' solution and moves towards that point with a certain probability based on the efficacy difference of the two solutions as well as a global time-dependent parameter temperature. In the beginning, the temperature is nearly infinite, and by the end it decreases to zero. The neighbor chosen is determined by the 'perturbations' of the solution, which is analogous to the metal being shaken. This prevents the algorithm from becoming stuck in local bests. Our algorithm uses 100 perturbations/samplings per 1º temperature drop.

Graphical Method

The graphics for our project are rendered using the General Mesh Viewer, a proprietary product of Los Alamos National Laboratories. The graphics demonstrate the relative stresses imparted upon the mesh by heat energy. A heat variable exists within the LibMesh equation protocol and the graphics demonstrate the relative value of this variable on the mesh.

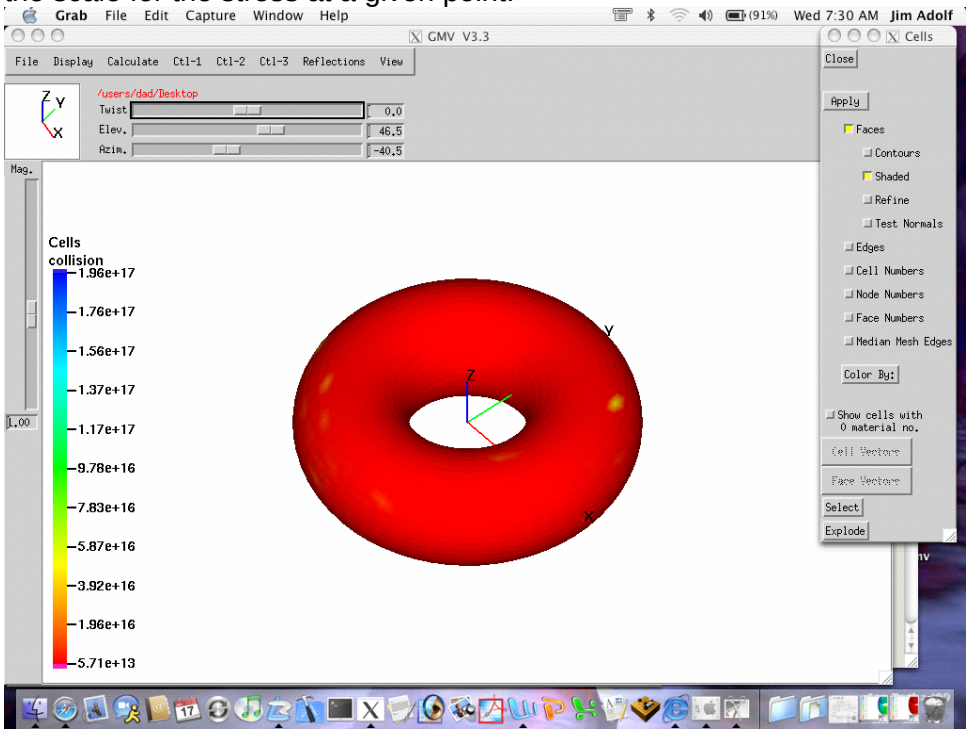
For the current simulation, we are using a torus (see **Mesh Implementation**). The torus is made from a cylinder, which is made of many subdivided hexahedrons, and these parts assume distinct parts of the finite element analysis.

When each hexahedron has received a heat-variable value, from the combined efforts of, then the results are stored in a standard file format that can be viewed with the General Mesh Viewer. They will allow us to view the success of a future simulation by viewing its heat distribution upon the mesh.

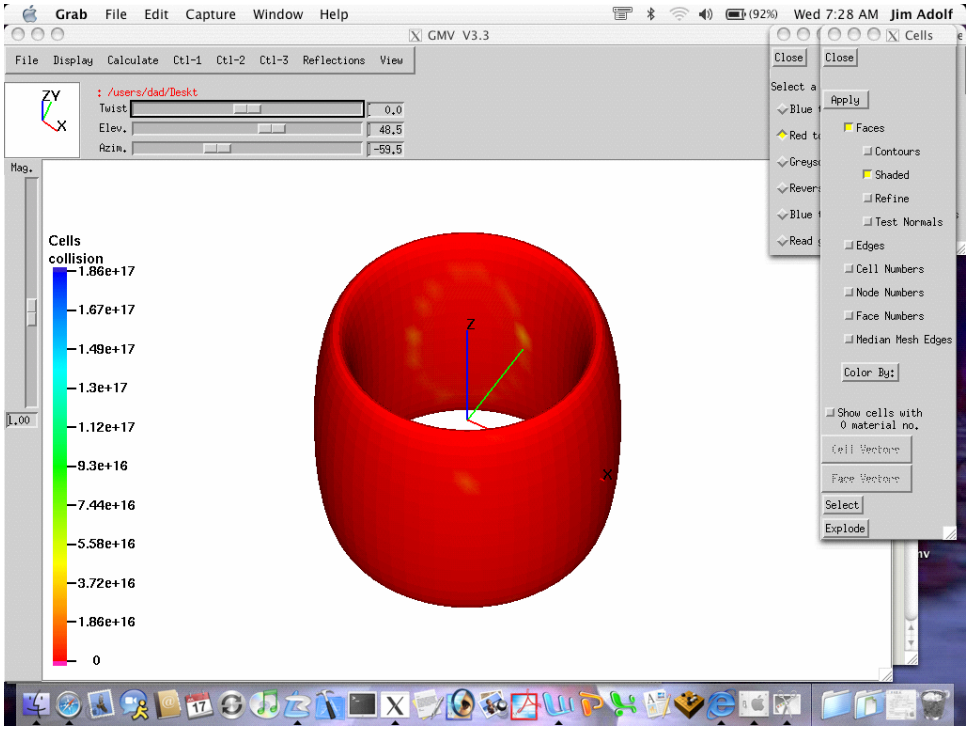
Results

Our program now can model the physics of a magnetically confined fusion tokamak. We have run many test runs using various geometries to ascertain that the physics model works correctly, and we have come to the conclusion that it does indeed work. Our Finite Element Analysis interfaces seamlessly with both the physics model and the graphics module, and the only thing left to do is interface the model of the tokamak with the optimization algorithms.

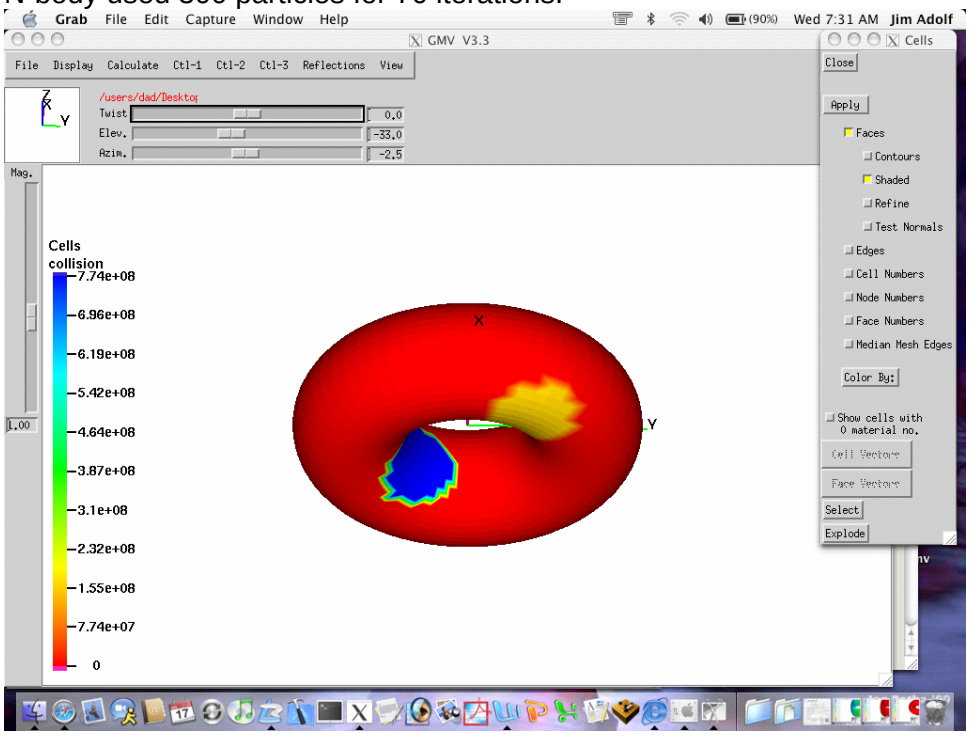
Here are a few pictures of various geometries. The colored bar on the left gives the scale for the stress at a given point.



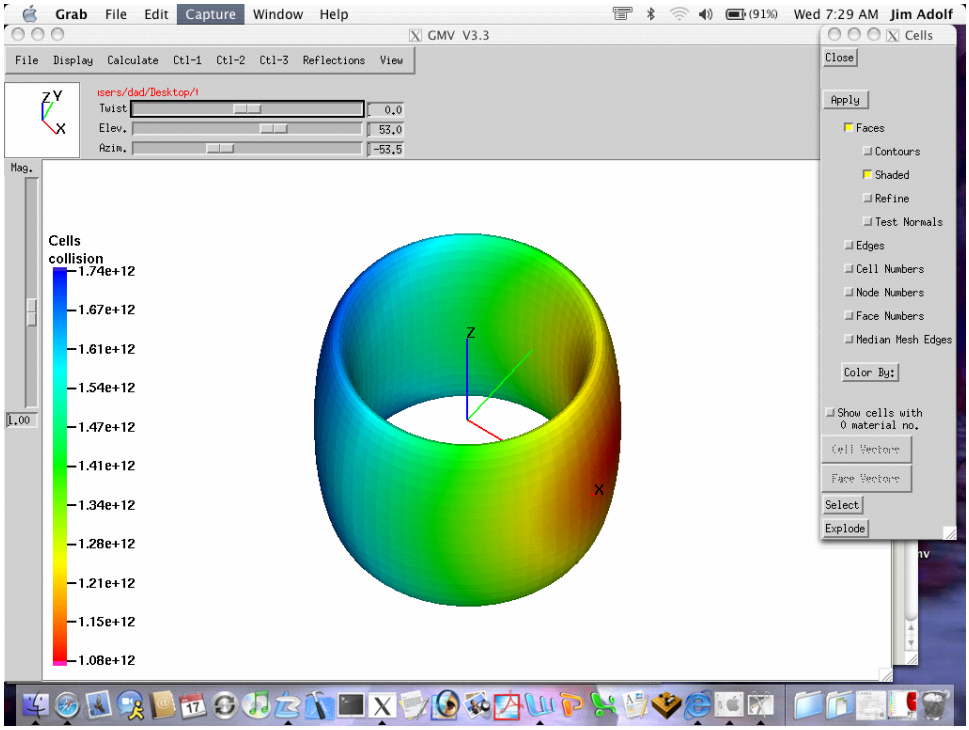
N-body used 30 particles for 20 iterations.



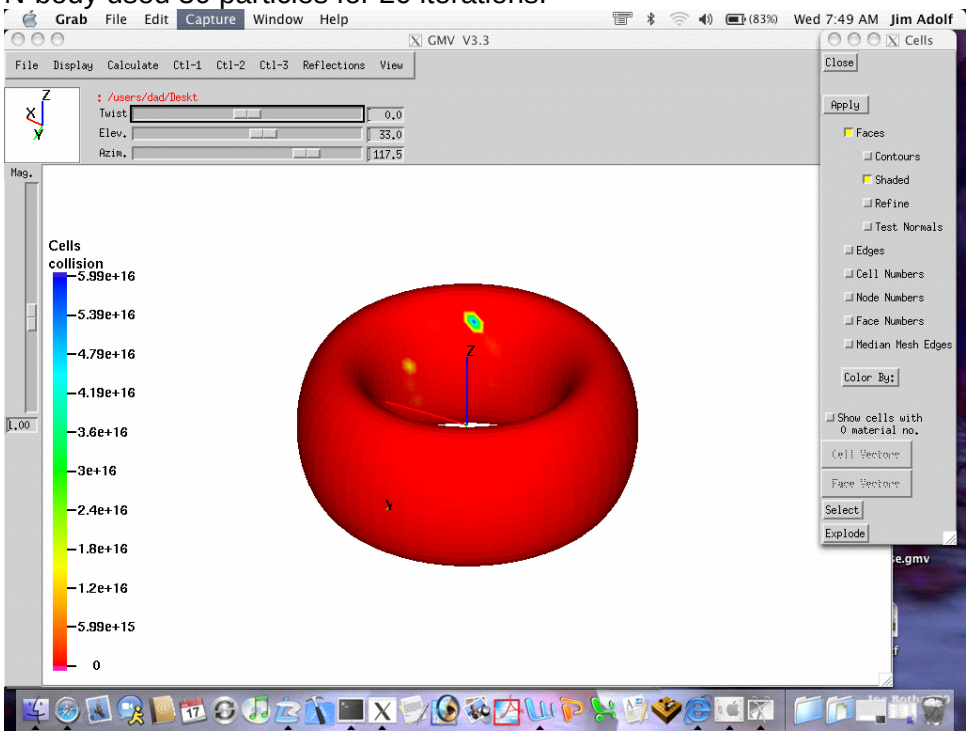
N-body used 300 particles for 70 iterations.



N-body used 2 particles for 20 iterations.



N-body used 30 particles for 20 iterations.



N-body used 30 particles for 20 iterations.

Conclusion

In summary, our program has the capability to solve all four parts of the original problem. It uses LibMesh, a finite element analysis program, to create an accurate mesh of a tokamak. It uses the n-body simulation to model plasma flow, and transfers the forces to LibMesh for analysis. It has working optimization algorithms, as well as functional visualization capabilities. The only thing remaining in the project is to integrate the tokamak model into the optimization algorithm.

Code

We have attached any code we have modified or written in a .tar.gz formatted file. All other code comes from the LibMesh and Blitz++ libraries.

Significant Achievements

We feel that our most significant achievements are the creation of a parameterized toroidal mesh, the use of the n-body simulation for the physics model, and the creation of accurate optimization algorithms.

The toroidal mesh was important because it allowed us to have the basis of the geometry, which is a major portion of our program. The physics model has had many problems throughout the process, and getting it to work makes the tokamak viable. The optimization algorithm is significant because it allows us to complete the goal of our project.

Acknowledgements

Jim Mims from Albuquerque Academy
Dave Humphreys from General Atomics

References

Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning*.
Luger, George F. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.*
Benenson, W., Harris, J. W., Stocker, H., Lutz, H. *Handbook of Physics*.
Felippa, Carlos A. *Introduction to Finite Element Methods.*
"Simulated Annealing" http://cs.felk.cvut.cz/~xobitko/ga/
"C++ Manual: Genetic Algorithms" http://www.alpcentauri.info
Numerical Recipes in C++, 2nd Edition, Press et. al.