An Experiential Learning Project to Bridge the Gap Between Programming and CAD

Jordan Cavender and Nathan Washuta
The Citadel, Charleston, SC

Abstract

In the first two years of the Mechanical Engineering curriculum at The Citadel, students take a computer applications course related to programming in MATLAB and a separate course related to CAD modeling in SolidWorks. Typical student feedback is very positive in the CAD course due to the hands-on nature and real world applications evident in the coursework, but students often struggle to appreciate the useful applications of programming, which leads to lower engagement in the course. In this paper, the authors present an independent study project that focuses on modifying an STL file in MATLAB to optimize its shape in response to applied forces. Through this ongoing exercise, the student has gained an understanding of data types, programming structures, algorithm development, and other coding best practices through a tangible, real-world application. The present paper will discuss the problem formulation and approach, the learning process undergone by the student, and a discussion of the applicable ABET outcomes to this project, as well as recommendations for adapting this independent study project into a course module that would increase student engagement in programming.

Keywords

MATLAB, SolidWorks, STL, FEA, Mesh

Introduction

Computational tools and methods are important skills for aspiring engineers and have only grown in importance over time. Indeed, the ability of engineers to adapt to state of the art technologies is a fundamental skill for engineering students and professionals, alike. In 2004, the National Academy of Engineering published The Engineer of 2020: Visions of Engineering in the New Century, which laid out the skills and competencies necessary for engineers of the future. In that report, the word technology is used 167 times, and it is emphasized that "Given the expected role of computers in the future, it is essential that engineers of all disciplines have a deep working knowledge of the fundamentals of digital systems as well as fluency in using contemporary computer systems and tools" [1]. The need for programming and 3D modeling knowledge is becoming ever more important in the context of Mechanical Engineering with the rise of programmable micro-controllers and 3D printing technologies. Courses teaching students these skills have become a mainstay of engineering programs across the country.

Students at The Citadel begin taking computer applications courses early in the Mechanical Engineering curriculum, with a MATLAB-based programming course typically taken during the second semester of their freshman year and a SolidWorks-based CAD modeling course during the first semester of their sophomore year. While these courses are often taught by the same

instructors, using similar teaching methods, student perceptions of these courses differ significantly. In institutional surveys over the last 2 academic years, student responses to the prompt "I learned a lot in this course" produced an average score of 4.40 in the programming course, compared with a score of 4.73 in the CAD course. In free response answers, students often comment on the usefulness of CAD for their future careers, but lament the lack of real-world, hands-on applications that they can imagine in programming. Indeed, the literature shows that non-computing majors are often less inherently interested in programming and struggle to find value in these methods for their future job prospects [2]. It is suggested that engineering curricula should emphasize the utility of programming as a skill within engineering careers [3] to improve student outcomes. There is a clear need to introduce more physically-relevant and tangible real-world engineer applications into the curriculum of programming courses.

In contrast with the dense syntax and abstract nature of programming structures, CAD modeling presents a much more tangible result that students can easily associate with real-world objects and applications. With the wide availability of 3D printers, these objects can also be easily and rapidly translated into physical models. Many students encounter these models in the form of STL files, after these parametric models have been interpolated into polygonal approximations. These STL files are the default file format used by most 3D printers and a wide variety of freely-available, user-generated models can be found on websites like Thingiverse or GrabCAD. Because an STL file simply contains a list of coordinates of vertices that make up the surface of a 3D object, CAD software is not well-suited to modifying these models. Using MATLAB's matrix manipulation, the power of programming can be leveraged to perform a variety of operations on these models.

This paper describes an independent study project that attempts to bridge the gap between programming and CAD modeling through the manipulation and modification of 3D models in MATLAB. In this project, a senior-level undergraduate Mechanical Engineering student at The Citadel has attempted to use MATLAB to perform design optimization on a 3D representation of a coat hanger bracket by modifying its shape in response to applied forces and fixtures. Design optimization is a critical focus in engineering given modern advances and the rise of 3D printing and advanced manufacturing techniques. Companies are continuing to spend significant time and resources trying to reduce waste and increase manufacturing efficiency, and the use of rapid prototyping has aided in these initiatives.

While the short-term goal of this project is to perform iterative design optimization on an STL file through MATLAB, the authors believe that this project can also form the basis for a programming course module and project in which students apply programming knowledge to manipulate 3D models. Through this independent study project, the student has encountered many concepts relevant to an introductory programming course, such as modular programming, manipulation of data structures, algorithm development, recursive programming, and code debugging. While this module would primarily focus on the physical manipulation and modification of the model through MATLAB, students would also receive their first introduction to concepts that they will encounter later in the curriculum, such as 3D modeling, numerical methods, and optimization. The authors believe this project has the ability to provide students with a more tangible representation of their programming work, which would in turn provide students with more intrinsic motivation to improve their skills. Further development of the

process that the student followed through this independent study project, recommendations for future curriculum development, and applicable ABET student outcomes addressed in this project is presented.

Project Methods

MATLAB has a Partial Differential Equation Toolbox that can solve complex engineering problems including structural mechanics, heat transfer, electromagnetics, diffusion, and vibration as seen on their Finite Element Analysis page [4]. This package can be used by importing or creating geometry, preprocessing the geometry by meshing and defining initial conditions, solving, and postprocessing [4]. This package is a separate add-on from the main MATLAB Student R2020a package which was the version used for this project.

Students can use MATLAB's provided Deflection Analysis of Bracket example to gain a basic understanding of the functions used to conduct finite element analysis on a part and what their inputs and outputs are. The main functions and basic short description of their use described by MATLAB is provided below in order of how they are used [5]:

1. `structuralmodel = createpde('structural',StructuralAnalysisType)` returns a structural analysis model for the specified analysis type.
2. `importGeometry(model,geometryfile)` creates a geometry container from the specified STL geometry file, and includes the geometry in the `model` container.
3. `pdegplot(g)` plots the geometry of a PDE problem, as described in `g`.
4. `structuralProperties(structuralmodel,'YoungsModulus',YMval, 'PoissonsRatio',PRval)` assigns the Young's modulus and Poisson's ratio for the entire geometry.
5. `structuralBC(structuralmodel,RegionType,RegionID,'Constraint',Cval)` specifies one of the standard structural boundary constraints.
6. `structuralBoundaryLoad(structuralmodel,RegionType,RegionID, 'SurfaceTraction',STval,'Pressure',Pval,'TranslationalStiffness',TSval)` specifies the surface traction, pressure, and translational stiffness on the boundary of type `RegionType` with `RegionID` ID numbers.
7. `generateMesh(model)` creates a mesh and stores it in the `model` object.
8. `pdeplot3D(model,'ColorMapData',results.NodalSolution)` plots the solution at nodal locations as colors on the surface of the 3-D geometry specified in `model`.
9. `structuralStaticResults = solve(structuralStatic)` returns the solution to the static structural analysis model represented in `structuralStatic`.
10. `TR = stlread(filename)` returns a triangulation object TR containing the triangles defined in an STL file.
11. `trimesh(TR)` plots the mesh defined by a 2-D or 3-D triangulation object.

Since the purpose of this activity is to bridge the gap between CAD and mathematical programming, the MATLAB portion of the lesson can hold two objectives: run a basic finite element analysis demonstration using the object geometry as defined above, and refine and manipulate the STL mesh which will reinforce previously learned math concepts while building a stronger foundation for the programming language and its uses.

SOLIDWORKS Part

When treating this idea as a lesson plan the student would first begin by creating or downloading a simple geometry in SOLIDWORKS. This could range a basic shape such as a cube to a simple object that could have real world value to further the idea of real-world applications. Many basic geometries could be used, but a very complex or intricate part would increase mesh complexity, leading to an increase in computational expense and increase wait times for the student. This would begin to lose the core objective of the programming and CAD integration learning exercise. Two different geometries were used for this project's use cases: a unit cube for initial testing and processing and a single coat hanger wall bracket, shown in Figure 1 and Figure *2*.

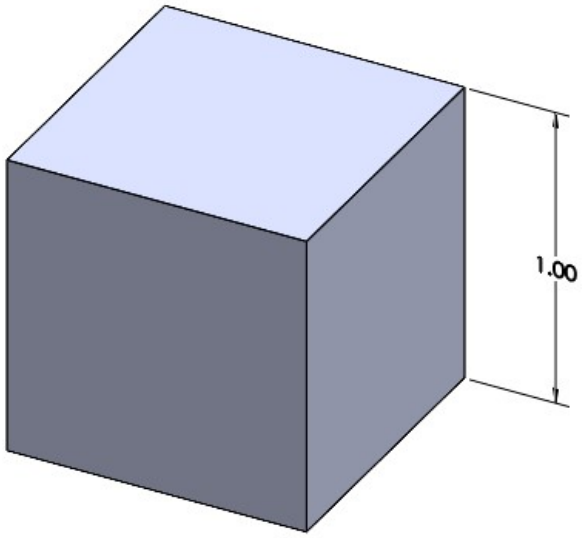

Figure 1: Simple cube CAD model in SolidWorks

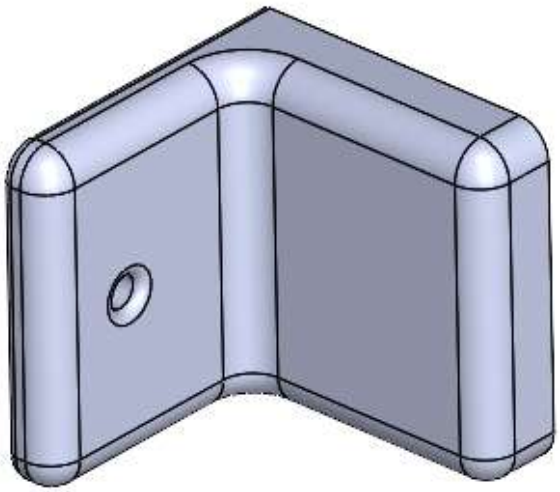

Figure 2: Coat Hanger Bracket CAD model in SolidWorks

Finite Element Analysis

Many models can be downloaded as STL files, but if it is created in the CAD software, it can be saved from the CAD program as an STL file. Using the MATLAB functions described earlier, the STL file can then be imported into MATLAB in order to perform FEA on the model. The first step of this process is to use functions 1 and 2 to import the model geometry for static analysis and set up a structure array that simulation settings will be added to in later steps. As shown in Figure *3*: Imported model in MATLAB, this does not retain the original geometry contained in the STL file, but breaks the model into faces so that loads and fixtures can be applied.

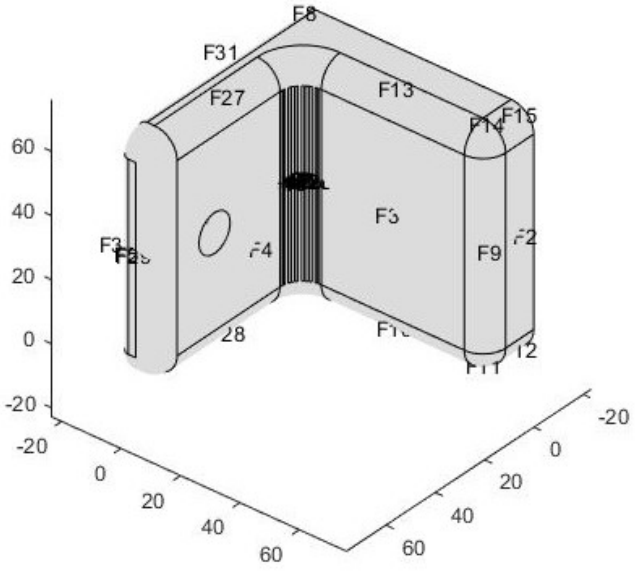

Figure 3: Imported model in MATLAB

Once imported, the student uses functions 4-6 to modify various properties of this structure array to manipulate material properties and applied loads and fixtures on the model. These loads and fixtures are applied to the faces of the model that correspond to what is shown in Figure 3. This requires the student to manipulate the model and determine the correct faces once the model is loaded and plotted. After these properties have been set, a mesh is generated using function 7, which interpolates the imported STL file into a tetrahedral mesh, as seen in Figure 4. While the STL file only includes vertices and edges defining the surface of the model, this finite element mesh divides the entire volume into tetrahedrons.

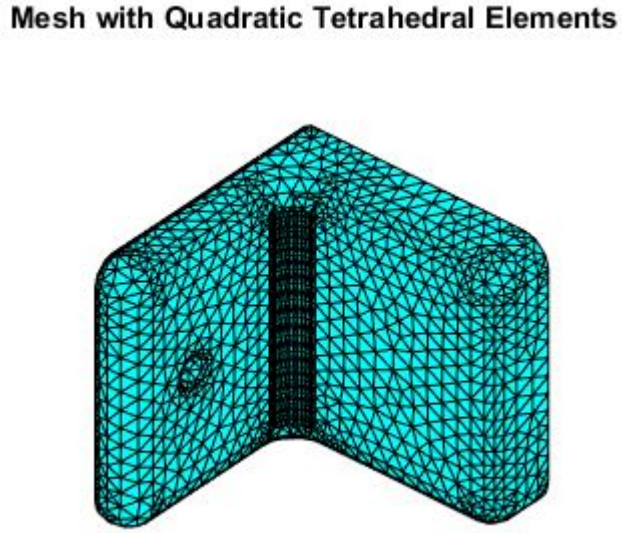

Figure 4: Volumetric mesh generated in MATLAB

MATLAB's finite element analysis solver is then used to solve the finite element solid mechanics equations, using function 9. Throughout this process, students are able to graph and view the model manipulations being applied by each function, reinforcing the tangible nature of these operations. A number of different results can be generated from these solved equations, such as Von Mises stress or y-direction displacement, shown in Figure 5.

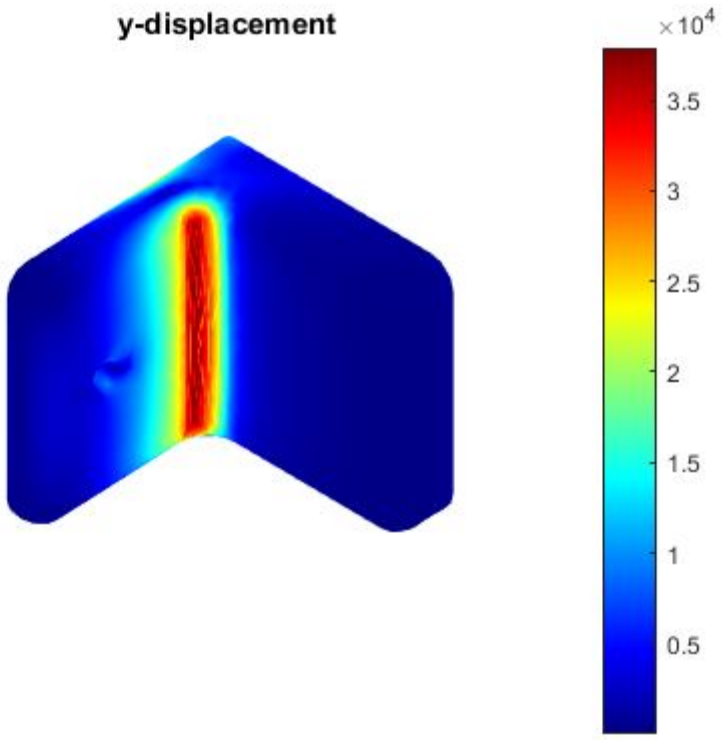

Figure 5: FEA displacement results generated in MATLAB

In order to begin the optimization process, the student decided that it was most effective to move the individual vertices of the model in order to add or remove material from the existing shape. To do so, the STL file must also be read into MATLAB and plotted separately using functions 10 and 11. This imports the model with the exact vertex locations defined in the STL file rather than the interpolated volumetric mesh created while performing the simulation. First, the `stlread` function is used to import the data again as a structure array. This `struct` includes a variable listing the locations of the vertices that define the surface as well as a variable that contains a list of which vertices connect to form surface triangles, called the connectivity list. Once loaded into MATLAB, the shape can also be plotted using `trimesh`, as shown in Figure 6 below.

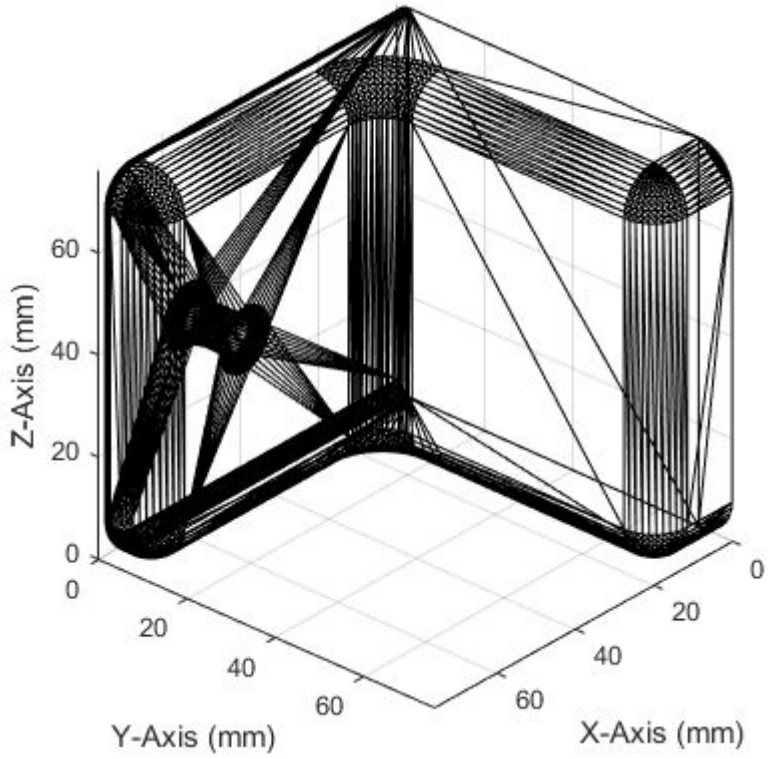

Figure 6: Coat hanger STL mesh in MATLAB

Once this mesh is imported into MATLAB with the associated vertices and connectivity list, it is necessary to refine the mesh from the original STL file. As with Figure 6, an STL file exported from CAD software typically includes as few points as possible to define the shape properly. On large flat faces, this leads to a small number of large triangles. If a single vertex of one of those large triangles was to be moved a small amount, as proposed for this project, the deformation of the model would not be localized to the desired location.

To overcome this issue, it was necessary to develop an algorithm through which new vertices could be added to the STL file in order to break the largest triangles into smaller, more manageable pieces. To accomplish this, the length of each edge was determined by cross referencing the connectivity list with the vertex array and calculating a pythagorean distance between each pair of vertices. The length of each edge in the shape was found, and the lengths were organized into a list with the greatest length being at the top. Each length was associated with triangle vertices and edges, and the longest edge was bisected to create two triangles from one. The vertex list and connectivity list were then updated to include the new triangle. Using loops, this process was iterated throughout the geometry until threshold value for maximum edge length was met. This created a more even mesh with smaller triangles, as seen in Figure 7a. By decreasing the threshold value, the maximum triangle size can be easily decreased, as shown in Figure 7b, although this increases computational expense significantly.

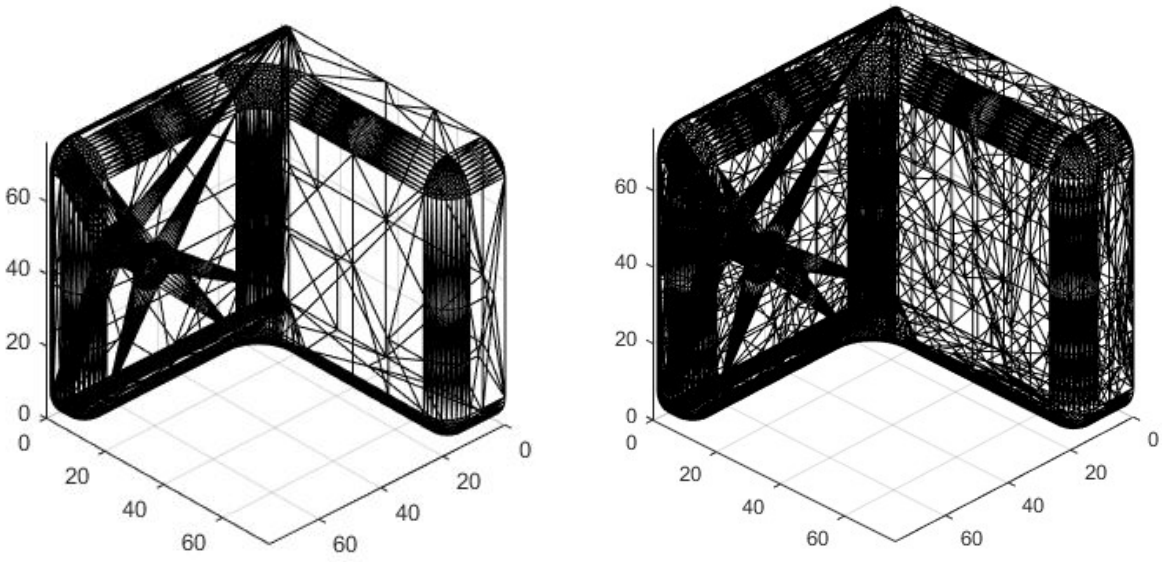

Figure 7: Refined STL meshes in MATLAB

Future Steps

The next step in this independent study project is to combine the results of these two methods to begin the optimization process. To accomplish this, regions of low stress in the FE results would first be identified, as these areas would be ideal candidates to remove material from the model. Because the FEA results exist on a separate tetrahedral mesh than what exists in the STL file, these results need to be interpolated onto the refined STL mesh. Once this interpolation is complete, the vertices can be moved in a direction that opposes the outward unit normal to reduce the amount of material in that location and the FEA analysis can be re-run. This process can be repeated iteratively to remove more material and improve the strength to weight ratio of the part. A similar method could be used to add material to any areas that experience high levels of stress.

ABET Student Outcomes

Through this independent study project, the student involved gained a tremendous amount of knowledge and was able to apply and reinforce a number of the ABET student outcomes that prepare graduates to enter the professional practice of engineering [6]. The primary outcomes addressed through this project were outcomes 1, 6, and 7. Student outcome 1 discusses the ability of students to identify, formulate, and solve complex engineering problems. While many engineering problems address this outcome, this project particularly focused on identifying and formulating a problem, as this idea was initially thought up by the student and each step of the process provided a new challenge that had to be overcome using engineering tools. Student outcome 6 discusses the ability to conduct experimentation, analyze and interpret data, and use

engineering judgement. Throughout this project, the student worked to develop new code and adapt the methods that were used based on the data and figures generated using MATLAB. Without the ability to interpret data, very little progress would have been made on this problem. Finally, student outcome 7 focuses on the ability to acquire and apply new knowledge. While many of the fundamentals of this project helped to reinforce existing programming knowledge that the student possessed, the specific application of that knowledge required the ability to learn new methods and strategies that lie well beyond what was learned in the classroom.

Conclusion

This paper describes an independent study project in which a student used MATLAB programming to manipulate and modify an STL file with the goal of optimizing the model based on FEA simulations. This project has the potential to be applied as a module and project in an introductory programming course in order to increase student engagement and motivation through tangible, real-world applications. Through this, programming can be used in conjunction with CAD modeling to increase student understanding, participation, and enjoyment of programming and its beneficial uses in their future engineering careers, as well as introduce and reinforce concepts that they will learn throughout their Mechanical Engineering coursework.

References

[1] National Academy of Engineering, "The Engineer of 2020: Visions of Engineering in the New Century," National Academies Press, Washington, DC, 2004.

[2] L. W. A. a. E. R. Lingar, "Work in Progress - Motivation Profiles of Non-Major Computer Programmers in a Flipped Classroom Environment," in *Proceedings of the 2017 FYEE Conference*, Daytona Beach, FL, 2017.

[3] K. J.-E. M. T. K. W. C. M. N. B. B. a. U. I. L. Steelman, "Work in Progress: Student Perception of Computer Programming Within Engineering Education: An Investigation of Attitudes, Beliefs, and Behaviors," in *Proceedings of the 2020 ASEE Virtual Annual Conference*, Virtual Online, 2020.

[4] MathWorks, "Finite Element Analysis (FEA)," MathWorks, [Online]. Available: https://www.mathworks.com/discovery/finite-element-analysis.html. [Accessed 8 December 2020].

[5] MathWorks, "Deflection Analysis of Bracket," MathWorks, [Online]. Available: https://www.mathworks.com/help/pde/ug/deflection-analysis-of-a-bracket.html. [Accessed 8 December 2020].