Interactive Visualization of Bezier and Hermite Curves in Python

Yusef Mostafa

Introduction:

While basic visualization and understanding of the methods of plotting curves and surfaces is critical to understanding the structure of 3D models, there is a lack of a simple interactable Python program that can visualize basic Bezier and Hermite curves and surfaces. Currently, the MATLAB app "ME6104_VisCurves_Cahn_Aaron_Ovsyannikov_Lucas.mat" is made available by the professor to the students taking ME6104 as a method of visualizing curves and surfaces. While this tool is helpful, it operates only in MATLAB, while the class uses primarily python, and is limited in its application of Bezier curves such that a Bezier curve of only 4 control points can be visualized in the program.

This project, *Interactive Visualization of Bezier and Hermite Curves in Python* aims to provide a simple program to visualize curves and surfaces learned in the class, to demonstrate competency in the learned material, as well as to provide an accessible, standalone program for others to visualize implicit surfaces. The general framework and scope of the project is as follows: First, the curves and surfaces desired to be implemented in the python program will be identified. Bezier and Hermite curves and surfaces will be implemented, as well as simple point to point graphing and interpolation. Functions to calculate the data points for each respective curve and surface for any given size will be developed. Additionally, functionality to perform 2D and 3D transforms will be included, such as rotations, translations, and scaling of the plot. Using these functions, a UI will be implemented in python, where the user can input control points and visualize curves and surfaces in real time.

Once a basic program and UI are developed, as time permits, user friendly options will be implemented to flush out the program, such as having multiple plots on one graph and visualization of bicubic Hermite and Bezier surfaces. Once the program is complete, the source

code will be exported into an executable file in the windows operating system such that it does not require an understanding of Python or MATLAB to visualize these complex geometries and use the program. An interactive curve and surface visualization program is relevant as it can serve as a learning assistant to students for the visualization of complex geometries or error checking of homework questions to anyone, despite their background in coding.

Methodology:

Bezier curves of any order, and cubic Bezier surfaces will be computed using the methodology discussed in lecture. The Bernstein polynomial at each point will be calculated (Eq 1), and used as an input in the Bezier computation, along with the desired control points (Eq 2-3).

$$B_{i,n} = \binom{n}{i} t^i (1-t)^{n-i} \tag{1}$$

$$\vec{P}(u) = [u^3 \;\; u^2 \; u \;\; 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{P}_0 \\ \vec{P}_0 \\ \vec{P}_0 \\ \vec{P}_0 \end{bmatrix} \tag{2}$$

$$\vec{P}^{uw}(u,w) = [-3(1-u)^2 \quad 3(1-u)^2 - 6u(1-u) \quad -6u(1-u) - 3u^2 \quad 3u^2] \begin{bmatrix} \vec{P}_{0,0} & \vec{P}_{0,1} & \vec{P}_{0,2} & \vec{P}_{0,3} \\ \vec{P}_{1,0} & \vec{P}_{1,1} & \vec{P}_{1,2} & \vec{P}_{1,3} \\ \vec{P}_{2,0} & \vec{P}_{2,1} & \vec{P}_{2,2} & \vec{P}_{2,3} \\ \vec{P}_{3,0} & \vec{P}_{3,1} & \vec{P}_{3,2} & \vec{P}_{3,3} \end{bmatrix} \begin{bmatrix} -3(1-w)^2 \\ 3(1-w)^2 - 6w(1-w) \\ -6w(1-w) - 3w^2 \\ 3w^2 \end{bmatrix} \tag{3}$$

Similarly, the cubic Hermite curve and surface equations can be computed (Eq. 4-5):

$$\vec{P}(u) = [u^3 \;\; u^2 \; u \;\; 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{P}(0) \\ \vec{P}(1) \\ \vec{P}^u(0) \\ \vec{P}^u(1) \end{bmatrix} \tag{4}$$

$$\vec{P}^{uw}(u,w) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} \vec{P}_{3,3} & \vec{P}_{3,2} & \vec{P}_{3,1} & \vec{P}_{3,0} \\ \vec{P}_{2,3} & \vec{P}_{2,2} & \vec{P}_{2,1} & \vec{P}_{2,0} \\ \vec{P}_{1,3} & \vec{P}_{1,2} & \vec{P}_{1,1} & \vec{P}_{1,0} \\ \vec{P}_{0,3} & \vec{P}_{0,2} & \vec{P}_{0,1} & \vec{P}_{0,0} \end{bmatrix} \begin{bmatrix} w^3 \\ w^2 \\ w \\ 1 \end{bmatrix} \quad (5)$$

Additionally, the rotation and translation matrices can be applied to each point of any of the curves and surfaces (Eq 6-9):

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (6)$$

$$R_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \quad (7)$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (8)$$

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \\ 1 \end{bmatrix} \quad (9)$$

To develop a basic visualization UI in python, basic data processing functions must first be coded to process the 2D or 3D data, and perform the desired Hermite, Bezier, or transformation operations. From a previous homework assignment, the functions BernsteinPoly(), BezierCurve(), and BezierSurface() will be used in processing Bezier curves and surfaces. The BezierCurve() function will be altered to accept inputs of any number of control points. Additionally, functions to compute cubic Hermite curves and surfaces will also be realized (HermiteCurve() and HermiteSurface()). Similarly, rotations and translations will be performed using a function Transform(), and scaling separately in a Scale() function. These are some of the

needed functions to visualize data, however 15 separate functions will be created to visualize and plot these curves and surfaces (see ME6104_Final_Project_Yusef_Mostafa.py).

The PySimpleGUI python library will be used for the UI design. A simplistic UI design will be used, consisting of a single panel, buttons, input boxes, and a plot to visualize the input data. Basic error handling will be implemented such that the program is functional and does not crash. Additionally, this program will function for the windows operating system of varying system display resolutions. The final program will be exported to an executable file for ease of access in future use.

Results:

Using the PySimpleGUI python library, a UI was created to perform the desired operations to the input data (Figure 1). Each button and text box were linked to desired functions within the python code. The program can be used by first inputting (x,y,z) coordinates into the input box parsed by commas and spaces, and clicking the button for the desired curve or surface plot. The same page interface is used for 2D and 3D plots. The program differentiates 2D and 3D plots by the shape of the input data (x,y) versus (x,y,z). Additional basic error handling was implemented in the code, where if a mistake is made, such as inputting data incorrectly, or attempting to plot, rotate, translate, or scale a 3D surface with 2D inputs (or vice versa) the program will not respond, and the user can try again.

In a few test cases, the GUI can be seen modeling complex surfaces, i.e. a 3D Bezier curve with 16 control points (Figure 2). This curve can then be rotated and translated as desired, and scaled individually among the x,y,z axes. Additionally, the GUI can be used to visualize a few other curves and surfaces to test the functionality of and error handling of the program interface (Figures 2-4). It is important to note that the transform operation will currently only work if

values are entered in both rotation and translation input boxes. If only a rotation or translation is desired, the other input box cannot be left blank, and must receive an input of (0,0) or (0,0,0).

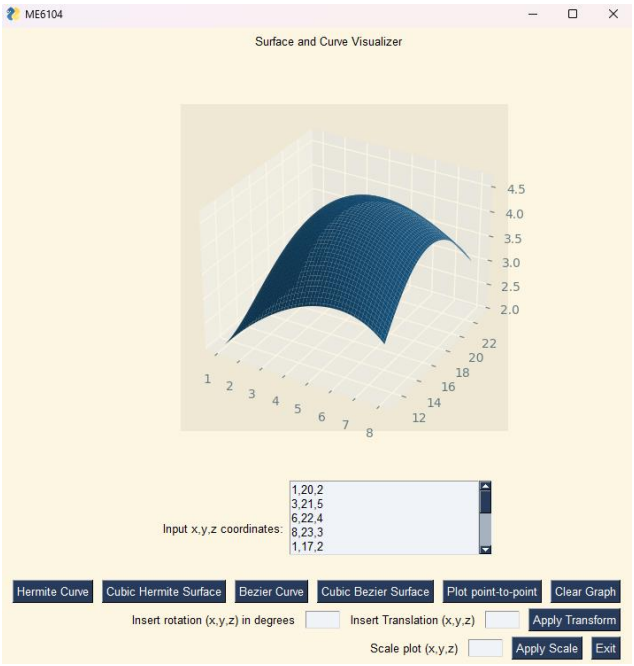

Figure 1: Bezier surface display on GUI

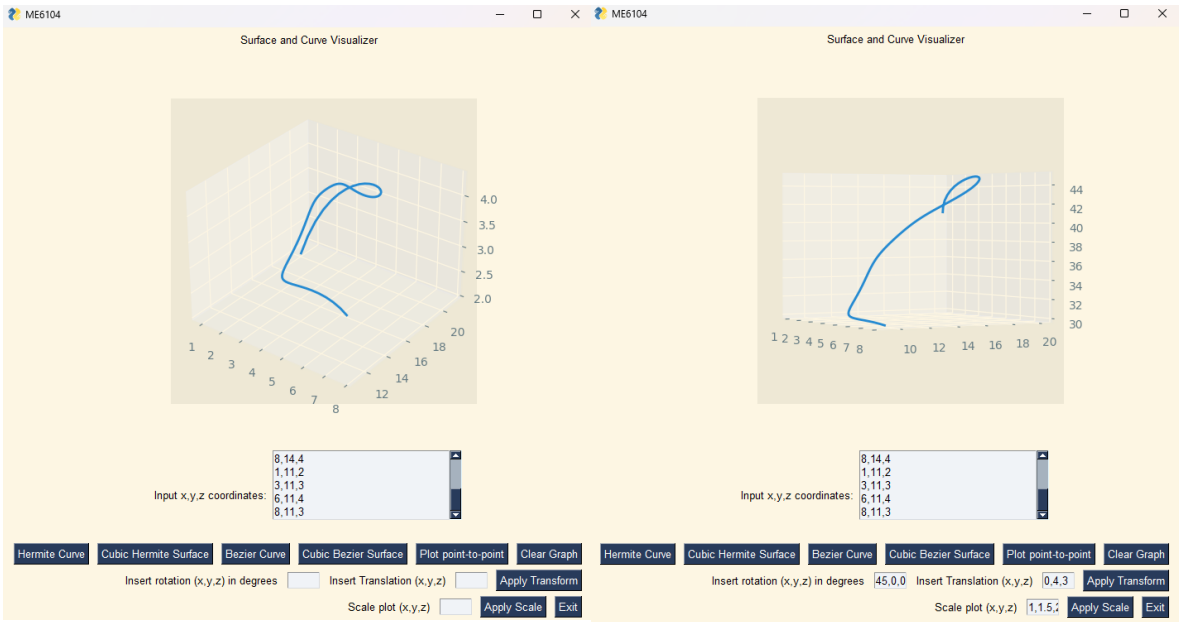

Figure 2: Use cases for Bezier curve with and without transform applied

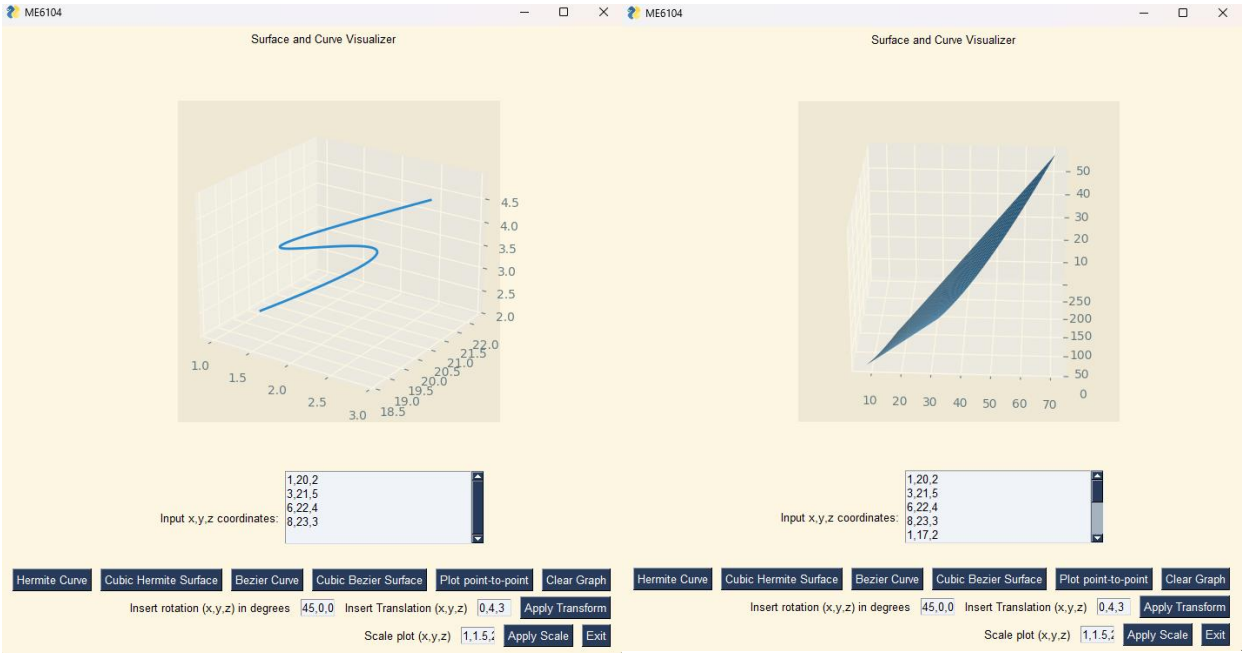

Figure 3: GUI use case for cubic Hermite curve and surface

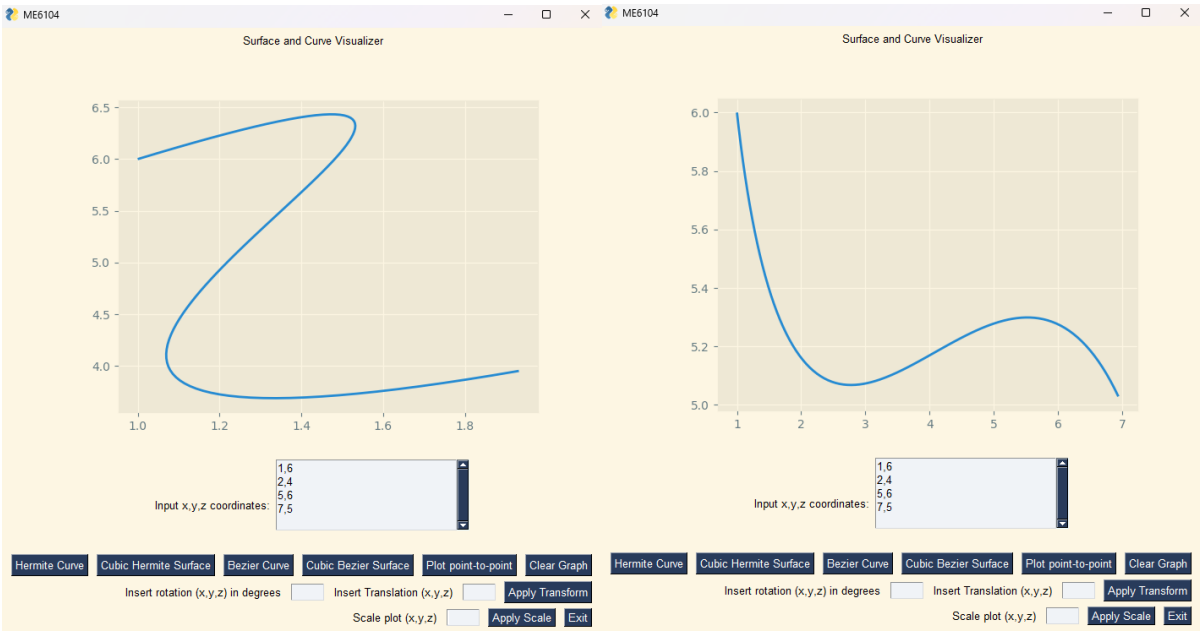

Figure 4: GUI use case for 2D Hermite (left) and Bezier curves (right)

Summary and Future Work:

Using basic function implementations along with a GUI, a curve and surface visualization program was developed to input control points of any type and output a 2D or 3D plot of the resulting curve or surface. After the initial plot, rotations, translations, and scale operations can be performed to further manipulate the plot. While the *Surface and Curve Visualizer* GUI can be used to plot complex curves and surfaces more simply, it is limited in its functionality. For example, only a single plot can be visualized at a time, and the plot window moves with the scale and translate function when plotting transforms, making the visualization of some of the transforms performed less effective. Given additional time, the functionality to view multiple plots overlayed on top of one another would be implemented, as well as better plotting window ranges and better error handling of the string input coordinates. While this program can be flushed out further in the future, it effectively acts as an easy to use, standalone visualization program for students to use as a learning aid.

References:

[1] K. Leon, “Integrating pyplot and pysimplegui,” *Medium*, 07-Oct-2021. [Online]. Available: https://towardsdatascience.com/integrating-pyplot-and-pysimplegui-b68be606b960. [Accessed: 24-Apr-2023].