



**UTM**  
UNIVERSITI TEKNOLOGI MALAYSIA

# **GEOMETRIC MODELLING**

## **SCSV3113**

**SESSION 2019/2020 - SEMESTER 1**

**PROJECT**

**-Creative Application using Curve and Curve Surfaces :  
MATLAB**

NAME : ONG LE FOO A17CS0189

STEVEN YONG WEI B18CS0025

TAN SEE JOU A17CS0218

TAN SHI XUAN A17CS0219

LECTURER : DR NORHAIDA BTE MOHD SUAIB

SECTION : 01

# CONTENTS

BACKGROUND .....	2
1. INTRODUCTION .....	2
2. AIM .....	3
3. OBJECTIVES.....	3
4. SCOPE.....	4
5. SELLING POINTS.....	4
6. IMPLEMENTATION .....	4
6.1 3D Bezier Curve.....	5
6.2 3D Bezier Surface .....	7
6.3 Color Mapping .....	10
6.4 Reset .....	10
7. USER INTERFACE .....	11
7.1 General User Interface .....	11
7.2 Bezier 3D curve.....	15
7.3 Bezier Surface .....	19
8. CONCLUSION .....	25
9. ACKNOWLEDGEMENT .....	26

# BACKGROUND

Curve is a geometric object that is similar to a line but does not have to be straight. Even though curve is only a simple line, however, it can be seen in most of the things that we used to interact with. Looking back to prehistoric times, curve had been visualized as art and decoration. During that time, mathematicians started to be curious about it and until nowadays, curve line has become part of human life and had been utilized for graphical representation of functions. Therefore, it is important for us to know and explore the beauty of a curve.

## 1. INTRODUCTION

This project documentation is about the creation of Creative Application using Curves and Curved surface. In creation this application, instead of building different geometry model, the focus will be on 3D curves and curve surface only. Matrix Laboratory technology had become the creation tools for this application.

Matrix Laboratory or Matlab is a high-level programming language consisting of interactive environment that mainly used for numeric computation, application development, programming and visualization. Matlab is good in model and application creation, extensive data analysis and algorithms development. It also provides a lot of built-in libraries and functions for mathematical operations that involve calculation, statistics, graph plotting, and so on. Besides, Matlab become the choice of creation tools also due to its support for building applications with custom graphical user interfaces (GUI). Therefore, it is clear that Matlab is a versatile tool designed for computational mathematics and application development.

In this project, we will use the 3D bezier curves and 3D bezier surface. Both of these bezier curves are the simple curve and it uses the general bezier equation.

$$C(u) = \sum_{i=0}^n B_{n,i}(u)P_i$$

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

$$\binom{n}{i} = \frac{n!}{i! (n-i)!}$$

However, the equation above works only on the bezier curve. Therefore, for the surface bezier to work, we expand the equation by adding another parameter since the surface need two direction parameters.

$$P(u, v) = \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) P_{ij}$$

To convert a 2D curve or surface to a 3D, we simply add a z-axis onto the control point P.

## 2. AIM

The project aims to expose the knowledge of the Bezier curve and surface in 3D and 2D to the people who use 3D modelling software especially the students that are involved in the courses of Computer Science. Since the Bezier curve and surface is widely used in computer graphics related software, the application will be the function as to show how the curve can be generated using the parametric equation which shows at the figure above. Our application will focus more for the beginner who are up for learning, and may use this as their reference.

## 3. OBJECTIVES

- ❑ To develop an application that provides tool for curves and curved surfaces modelling.
- ❑ To customize a simple 3D modelling application by using Matrix Laboratory.

- ❑ To provide a simple interface for Bezier curve and surface modelling.

## 4. SCOPE

Our focus is on making a modifiable 3D Bezier Curve and Surface. The application will provide an interface for the user to key in their input of x,y,and z axis for each control points of the curve. Since Matlab is a very complicated in terms of interactive interface, Matlab are unable to make mouse click drag and drop on the object. However, we used an alternative way that allow the user to modify the control points. There is also additional customisation of the surface color gradient to make it look good.

## 5. SELLING POINTS

The application is rather simple, it gives an insight look to the 3D Bezier Curve and Surface. The application focuses more on the beginner to use it as a reference to the Bezier curve and surface. This application provides simple and attractive user interface. The application has multiple advantages such as providing the user to input the points as the axis in the provided textboxes. The points can be plotted at any axis as the plot can be stretched into bigger space. Also, the user can view at any perspective view and ability to move the curve in any direction. Last but not least, to enhance the visual on the human perspective, the colour of the Bezier surface can change to fit the current scenario.

## 6. IMPLEMENTATION

From what we had discovered an experimented in our previous assignment, we had already found a way to implement bezier curve and bezier surface on a 3D platform. Now, we want to improve that and make a way to modify the control point of the curve and surface.

## 6.1 3D Bezier Curve

Before that, let's go through on how to implement the bezier curve and surface. Firstly, the 3D bezier curve can be drawn based on user input by adding each control point. The control point will increase based on the x y z position input by the user and plotted on the 3D surface. Everytime a new control point are plotted, the bezier curve will be calculated based on the bezier equation.

```
function Q=Bezier(P,t)
for k=1:length(t)
    Q(:,k)=[0 0 0]';
    for j=1:size(P,2)
        Q(:,k)=Q(:,k)+P(:,j)*Bernstein(size(P,2)-1,j-1,t(k));
    end
end
end

function B=Bernstein(n,j,t)
    B=factorial(n)/(factorial(j)*factorial(n-j))*(t^j)*(1-t)^(n-j);
end
```

Code 1 : 3D Bezier Curve implementation

```
function A=defaultBezier3dCurve(P)

t=linspace(0,1,100);
Q3D=Bezier(P,t);

plot3(Q3D(1,:),Q3D(2,:),Q3D(3,:), 'b', 'LineWidth', 2),
hold on
plot3(P(1,:),P(2,:),P(3,:), 'g:', 'LineWidth', 2) % plot control polygon
plot3(P(1,:),P(2,:),P(3,:), 'ro', 'LineWidth', 2) % plot control points
view(3);
box;
hold off

end
```

Code 2 : Plotting 3D Bezier Curve

In these functions, the plotting function will bring in variable P, where P is a matrices that contain all the coordinates of every control point.

```
function addCP_Callback(hObject, eventdata, handles)

cpx = str2double(get(handles.CurveCP_X, 'String'));
cpy = str2double(get(handles.CurveCP_Y, 'String'));
cpz = str2double(get(handles.CurveCP_Z, 'String'));
cla reset;
axes(handles.axes1);
P = getappdata(GMproject, 'P');
P = [P [cpx;cpy;cpz]];
defaultBezier3dCurve(P);
setappdata(GMproject, 'P', P);
```

Code 3: Function in GUI for adding control point

These function will take the value of user and add a new column onto the P matrices where it will redraw the new bezier curve with new added control points.

Next, we added a modifying function where user can change the position of the control point they had plotted. First of all, Matlab doesn't support drag and drop on a 3D plotting surface, it may be done in 2D but somehow it may be very complicated and impossible to be done in 3D. However we came up with another solution which is by using the GUI button to move the coordinate. We include the button with 6 direction, which is 2 direction for every axis, x, y and z. Besides, there is also value for input to modify which control point and on how much distance to move.

```
function curveYplus_Callback(hObject, eventdata, handles)

cpNo = str2double(get(handles.curveCPNo, 'String'));
changeValue = str2double(get(handles.SurfChangeValue, 'String'));

P = getappdata(GMproject, 'P');
P(2, cpNo) = P(2, cpNo) + changeValue;
defaultBezier3dCurve(P);
setappdata(GMproject, 'P', P);
```

Code 4: Example of increase in Y-axis of one control points.

Once the button is clicked, this function will be called. The input of the control point number, **cpNo** and the distance, **changeValue** are being stored. The variable, **P** is a matrix where all the coordinates of control points are stored, using the cpNo, we can know which column of P (which control points) to be changed and how much to change on which axis. Since the example is increasing in Y-axis here, we use the 2nd row of **P** which is where all the Y-coordinate of all control points are stored.

## 6.2 3D Bezier Surface

Next is the implementation of 3D Bezier Surface. To implement this, we use a similar approach on calculating the bezier. However, we had changed the way of keeping the control points, we used **cell()** in matlab which is similar to array. Each node have a matrix of each point which indicates the coordinates of control points. Our 3D Bezier Surface included only 9 control points, this is because if we were to make more control points, the user will have a hard time to modify it and due to the lack of interactivity of the Matlab. This will surely become one of the weaknesses in our project.

```
cp=cell(3,3);

cp{1,1} = [scp1x, scp1y, scp1z];
cp{1,2} = [scp2x, scp2y, scp2z];
cp{1,3} = [scp3x, scp3y, scp3z];
cp{2,1} = [scp4x, scp4y, scp4z];
cp{2,2} = [scp5x, scp5y, scp5z];
cp{2,3} = [scp6x, scp6y, scp6z];
cp{3,1} = [scp7x, scp7y, scp7z];
cp{3,2} = [scp8x, scp8y, scp8z];
cp{3,3} = [scp9x, scp9y, scp9z];
setappdata(GMproject,'cpsurf',cp);

axes(handles.axes1);
%C = [c11;c12;c13;c21;c22;c23];
Bezier3DSurface(cp);
```

Code 5: Making the cell for each control point



```

[n,m]=size(cp);
n=n-1; % 1-direction
m=m-1; % 2-direction
% convert plotting vectors into matrices
p1=numel(V);
p2=numel(U);
V= repmat(V,p2,1);
U= repmat(U',1,p1);
% initialize
X=zeros(p2,p1,(n+1)*(m+1));
Y=zeros(p2,p1,(n+1)*(m+1));
Z=zeros(p2,p1,(n+1)*(m+1));
k=1;
for i=0:n
    for j=0:m
        niF=factorial(n)/(factorial(i)*factorial(n-i));
        Bin=niF*V.^i.*(1-V).^(n-i);
        mjF=factorial(m)/(factorial(j)*factorial(m-j));
        Bjm=mjF*U.^j.*(1-U).^(m-j);
        X(:,:,k)=Bin.*Bjm.*cp{i+1,j+1}(1);
        Y(:,:,k)=Bin.*Bjm.*cp{i+1,j+1}(2);
        Z(:,:,k)=Bin.*Bjm.*cp{i+1,j+1}(3);
        k=k+1;
    end
end
Xout=sum(X,3);
Yout=sum(Y,3);
Zout=sum(Z,3);

```

Code 6: 3D Bezier Surface Calculation

```

P=cell2mat(ctemp);
Q=cell2mat(ctemp2);

% Plotting vectors
U=0:0.005:1; % 1-direction (201 points)
V=0:0.005:1; % 2-direction (201 points)
[Xout,Yout,Zout] = BezierSurface(cp,U,V);
% plot (40401 points)
Z=Zout(:);
scatter3(Xout(:),Yout(:),Zout(:),5,Z)
hold on
colormap(pink);
plot3(P(1,:),P(2,:),P(3,:), 'g:', 'LineWidth', 2)
plot3(Q(1,:),Q(2,:),Q(3,:), 'g:', 'LineWidth', 2)
plot3(P(1,:),P(2,:),P(3,:), 'ro', 'LineWidth', 2) % plot control points
hold off
end

```

Code 7: Scatter the point to form a 3D Bezier Surface

When the bezier is being calculated, we used 3 additional array, which is Xout, Yout and Zout. These 3 variable combine will form a big amounts of coordinate of point. We plot all these points to generate a 3D bezier surface. The value of u and v are in the range of 0 to 1 where the interval are 0.005. For each line of bezier, there are 201 points. Basically the surface are formed through millions of points. Instead of using **plot3()**, we used **scatter()** which plot all the points in the 3D surface and **scatter()** can be used to modify gradient color too later on. Besides, we had also used plot3 to plot the control points and convex hull, this is to make it easier for the user to see where the control point at and to modify it.

The way to modify the point is the same as the Bezier curve. We use button to increase or decrease the value of the coordinate. To achieve that 6 buttons were made, 2 for each axis. One is to increase the value and another is to decrease the value. The control points selection for surface however are a little different, we use two parameters, which act like row and column (a1,a2) that act as the index of control points. For example, if were to select the last control point, which is the 9th, user can input (3,3).

## 6.3 Color Mapping

```
function SpringColor_Callback(hObject, eventdata, handles)
colormap(handles.axes1, spring)
shading interp
lighting phong
light
material shiny
```

Code 8: Changing the color mapping of 3D Bezier surface into Spring color.

`colormap()` is a built-in function of MATLAB. It sets the `colormap` property of a figure. After adding `colormap` to the 3D Bezier surface, `shading`, `lighting`, `light` and `material` are added. These properties will produce a more realistic 3D Bezier surface.

## 6.4 Reset

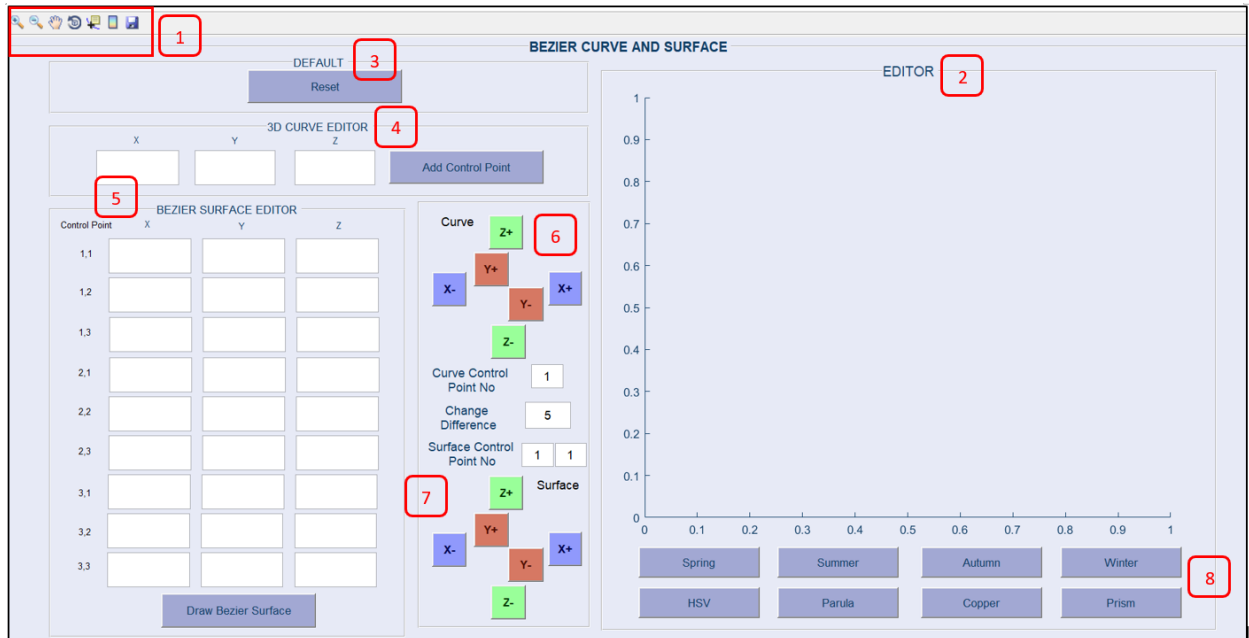
```
function default3dCurve_Callback(hObject, eventdata, handles)
cla reset;
axes(handles.axes1);
rmappdata(GMproject, 'P');
rmappdata(GMproject, 'cp');
```

Code 9: Reset function

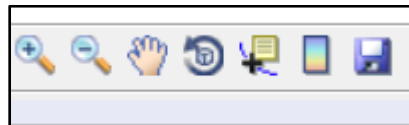
By clicking the button “reset” everything is reseted back to the default. The data of “p” and “cp” are removed at the same time. Both of these variable p and cp are the control point for bezier curve and bezier surface respectively.

## 7. USER INTERFACE

### 7.1 General User Interface



#### 1. TOOL MENU



The Tool Menu consists of (*from left to right*)

- Zoom In : Provide a detailed way to investigate the surface
- Zoom Out : Provide an overall vision of the surface
- Pan : Move around the surface/graph
- Rotate 3D : Rotate the surface in 3 dimensional way
- Data Cursor : Check the details of the points(X, Y, Z) location
- Insert ColorBar: Indicate the surface/graph color
- Save : Save the current model

## 2. EDITOR panel

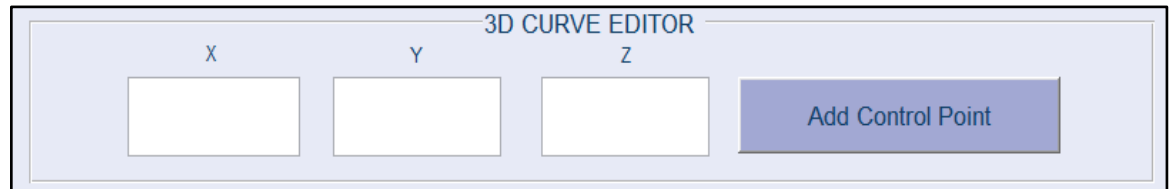
For displaying model purpose.

## 3. DEFAULT panel

*Reset Curve* button : Reset the curve in the Editor Panel

*Reset Bezier Surface* button : Reset the curved surface in the Editor Panel

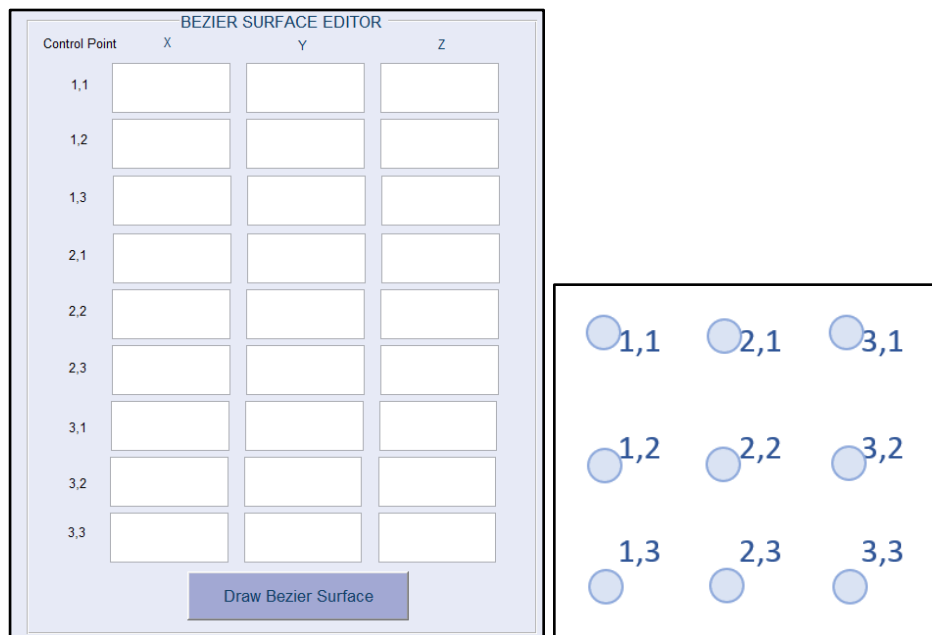
## 4. 3D CURVE EDITOR panel



The 3D CURVE EDITOR panel features a light blue background with a title bar at the top. Below the title bar, there are three input fields labeled 'X', 'Y', and 'Z' in a row. To the right of these fields is a blue button labeled 'Add Control Point'.

This panel is for 3D curve. User has to insert X, Y and Z location of control point. After inserted the location information of the control point, the user has to click the *Add Control Point* button to add it to the graph. Then a control point will be displayed on the Editor Panel. User is allowed to add the control points on the same graph to produce the preference curve model by repeating the same processes.

## 5. BEZIER SURFACE EDITOR panel

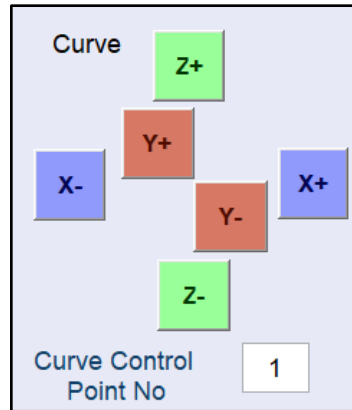


The BEZIER SURFACE EDITOR panel has a light blue background and a title bar. It contains a table for inputting control point coordinates. The table has four columns: 'Control Point', 'X', 'Y', and 'Z'. The rows are indexed from 1,1 to 3,3. Below the table is a blue button labeled 'Draw Bezier Surface'. To the right of the panel, a 3x3 grid of blue circles represents the control points, each labeled with its corresponding coordinates (e.g., 1,1, 2,1, 3,1 in the top row).

Control Point	X	Y	Z
1,1			
1,2			
1,3			
2,1			
2,2			
2,3			
3,1			
3,2			
3,3			

This panel is for Bezier curved surfaces. In this application, for Bezier surfaces, it is only 9 control points provided. Therefore, user has to insert the X, Y, Z coordinate for all the control points. After inserted all the control points, the user has to click the *Draw Bezier Surface* button and the surface will be displayed on the Editor panel.

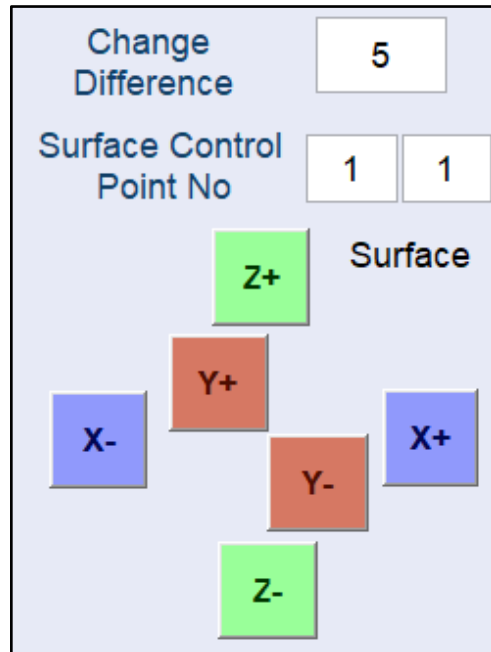
## 6. CURVE CONTROL POINT panel



This panel is for the user to control the location of each control point of the 3D curve. User can adjust the location of the control point by clicking the buttons. The reason that the placement of the y and z axis or not as the same as the general xyz axis is because the Matlab graph work in a different way. The Z-axis in Matlab are going on the same plane as the Y-axis in the general 3D axis, it goes up and down. On the other hand, Y-axis work like the Z-axis in the general 3D axis where it goes front and back.

Button	Function	Button	Function
X+	Increase the X coordinate	X-	Decrease the X coordinate
Y+	Increase the Y coordinate	Y-	Decrease the Y coordinate
Z+	Increase the Z coordinate	Z-	Decrease the Z coordinate

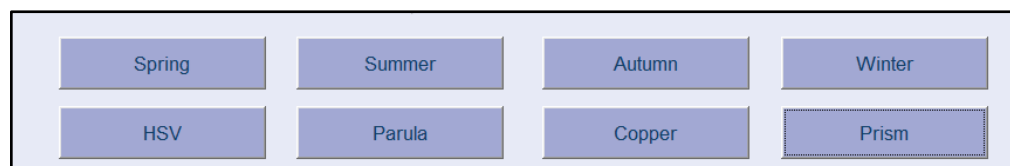
## 7. SURFACE CONTROL POINT panel



This panel is for the user to control the location of each control point of the 3D Bezier surface. User can adjust the move distance by inserting the value in ***Change Difference*** space. Besides, the user can select the desired control point by inserting the value for it, for example (1,1), (1,2) and so on. After selecting the control point, user can adjust its location by clicking the button

Button	Function	Button	Function
X+	Increase the X coordinate	X-	Decrease the X coordinate
Y+	Increase the Y coordinate	Y-	Decrease the Y coordinate
Z+	Increase the Z coordinate	Z-	Decrease the Z coordinate

## 8. COLOR panel



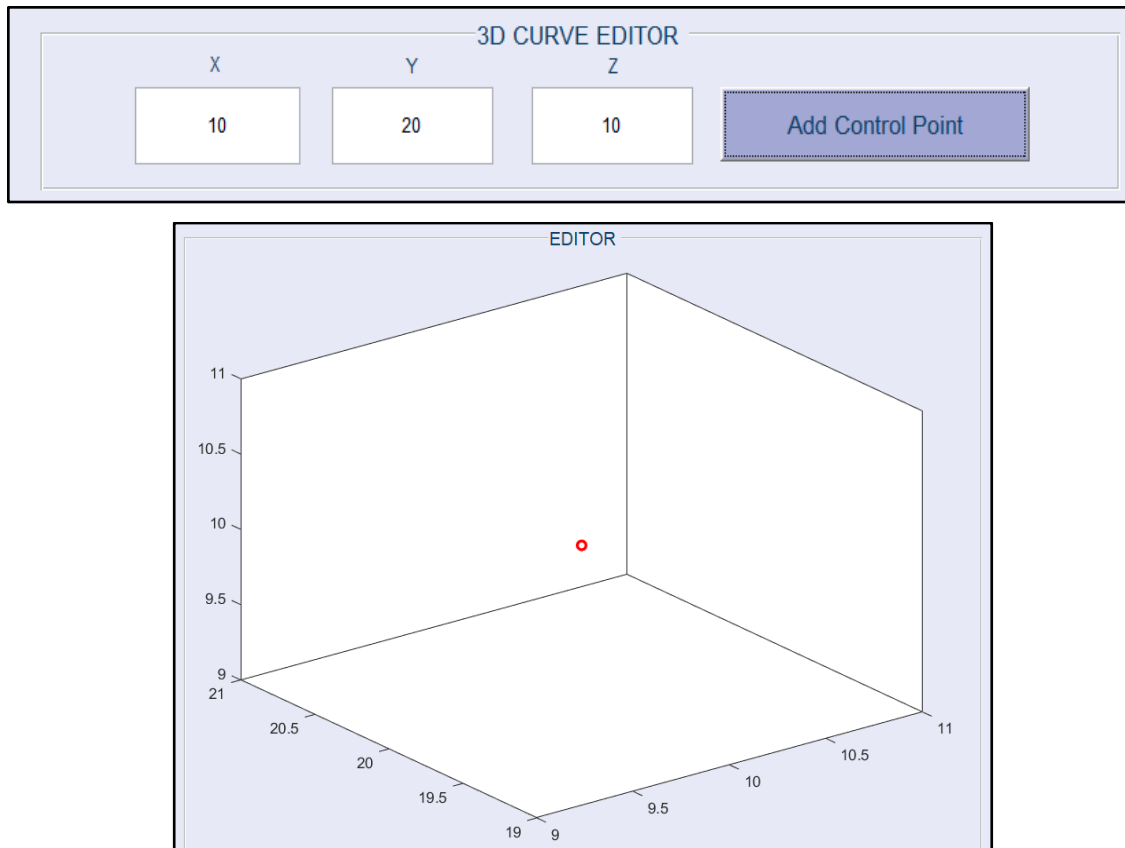
Different type of color had been provided to different preference.

\*Default color: Pink

## 7.2 Bezier 3D curve

### A. Plotting Curve

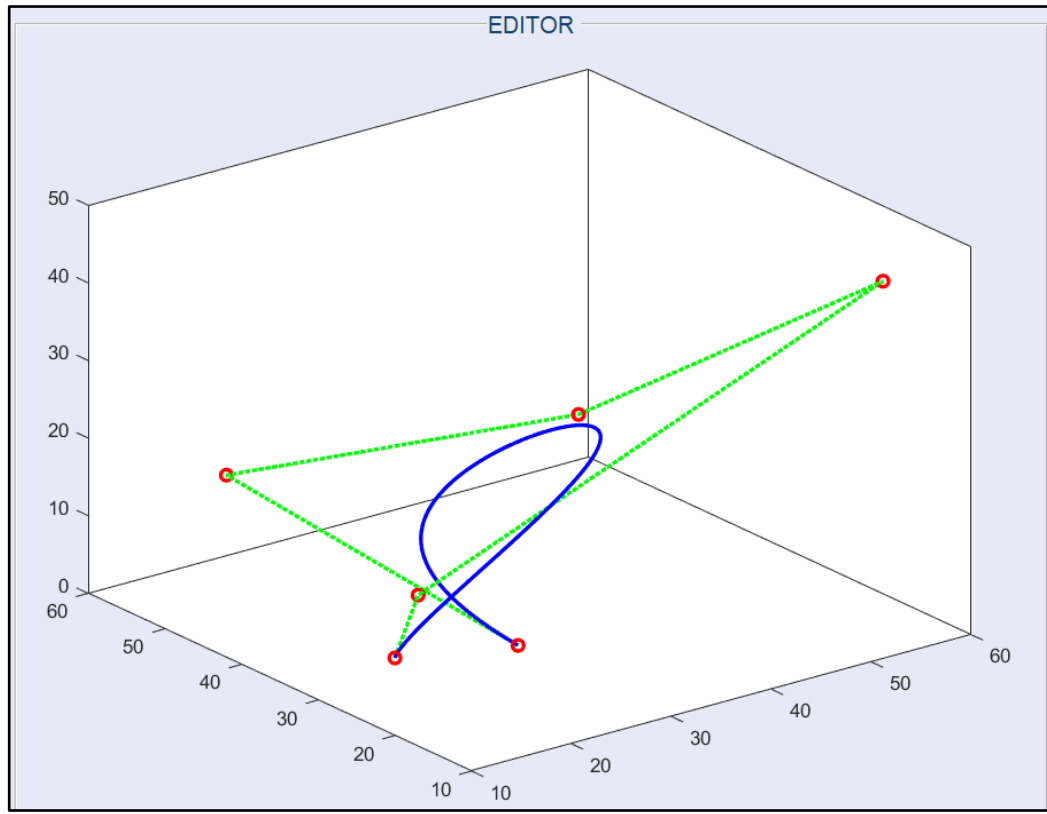
First, have to insert the X, Y and Z coordinate for the control point and click the **Add Control Point** button. Once clicked, the control point will be shown on the Editor Panel.



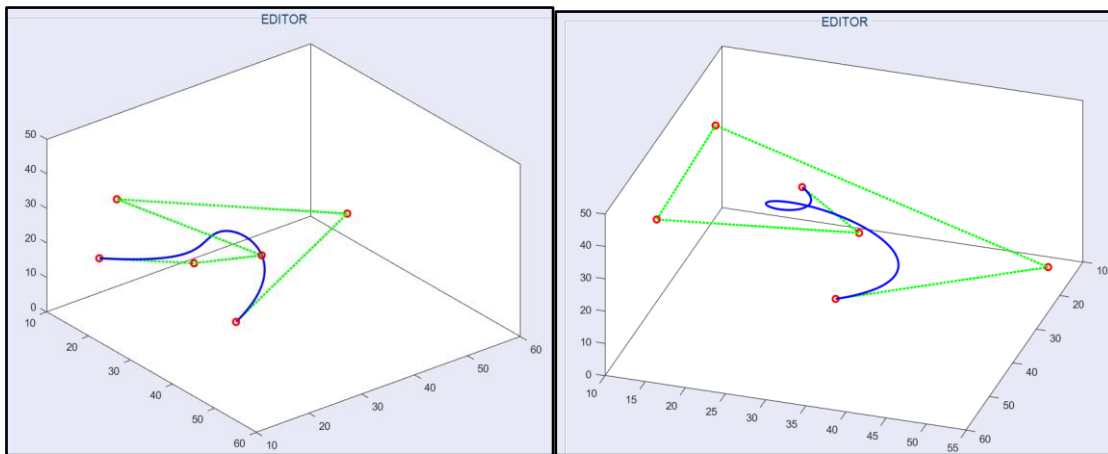
One control point is not enough to form the Bezier curve. Therefore, have to repeat the above processes a few times with different location control points. The output 3D bezier curve is shown as below

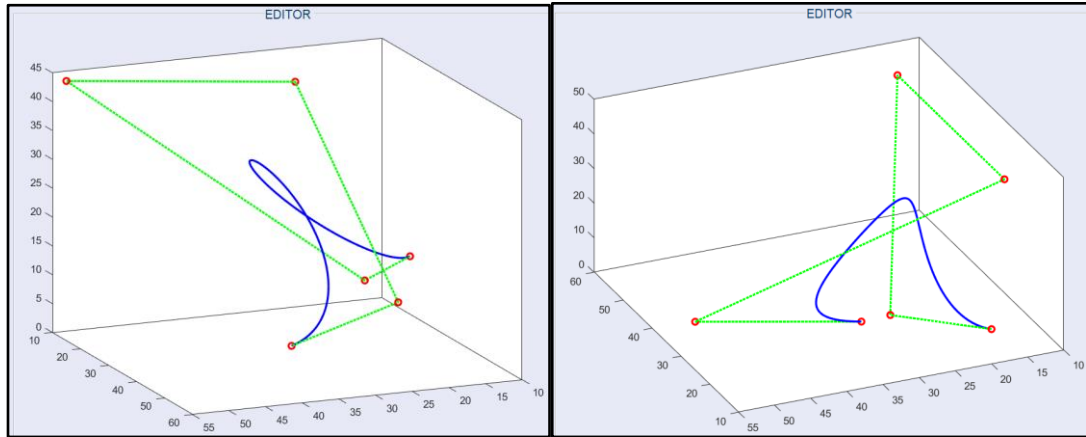


## B. Rotate 360 degrees



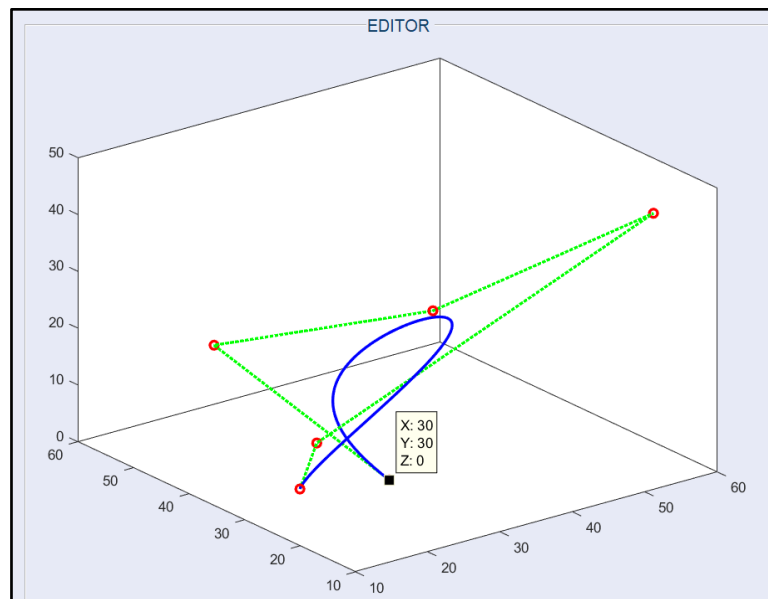
Besides, the **Rotate 3D** tool can be used to rotate the model in 360 degrees to have different perspective. Therefore, this application providing a convenient way for the user to have a better and clear understanding about the model. The below figures shown the different perspectives of the 3D Bezier curve model.



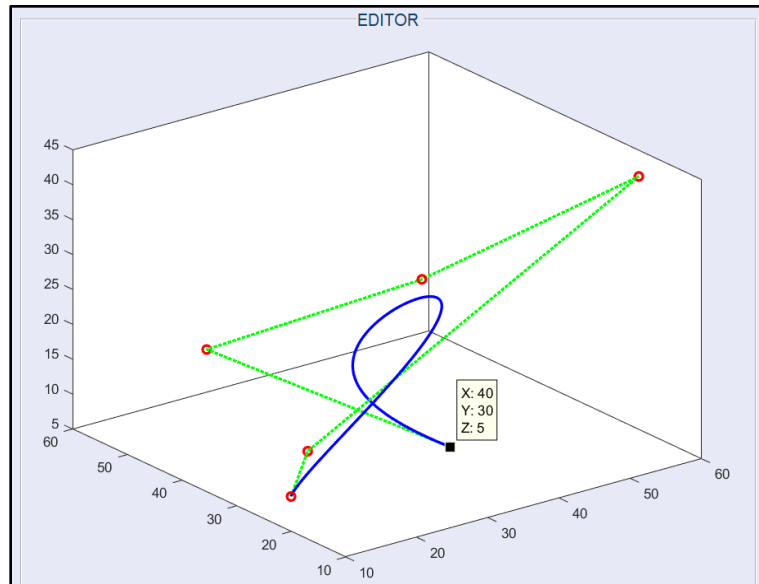


### C. Adjust control point location

In this case, the 6th control point had been chosen. The initial location of the control point is (30, 30, 0). **Data Cursor** tool had been used to show the location of the control point. Then, a X-coordinate adjustment button and Z-coordinate adjustment button had been used to make a few adjustments on X coordinate and Z coordinate. The location of the control point after adjustment become (40, 30, 5).

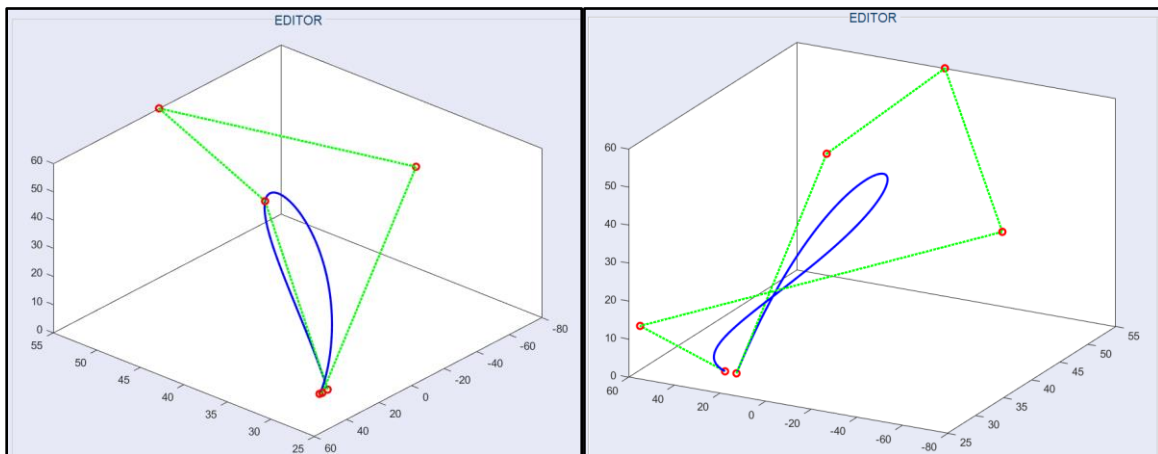


Initial location of 6th control point is (30, 30, 0).



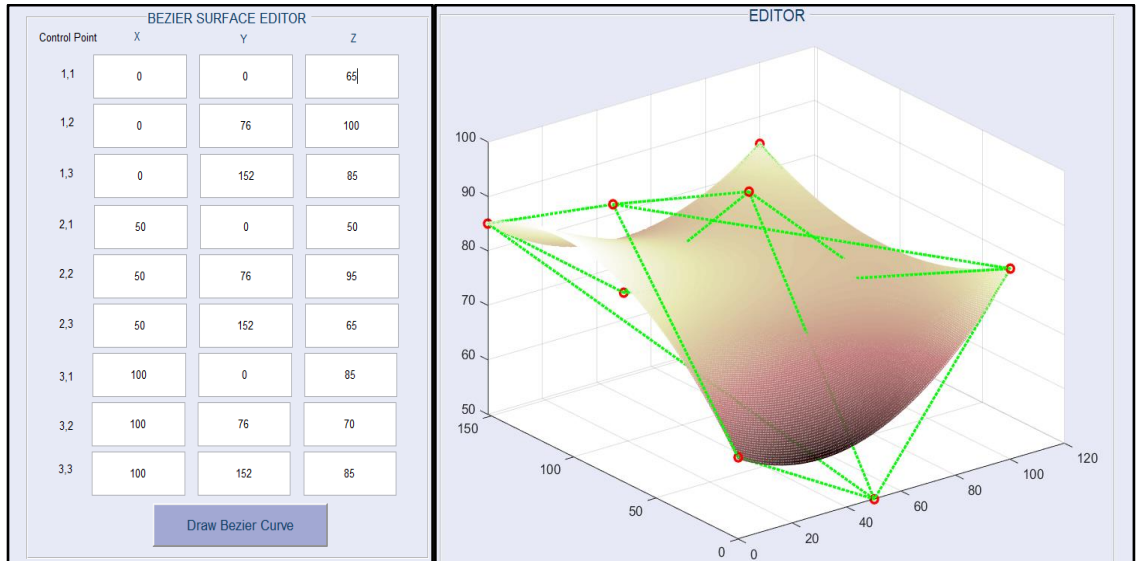
Location of 6th control point after adjusting is (40, 30, 5).

After that, a few adjustments made on others control points as well and had shown below.

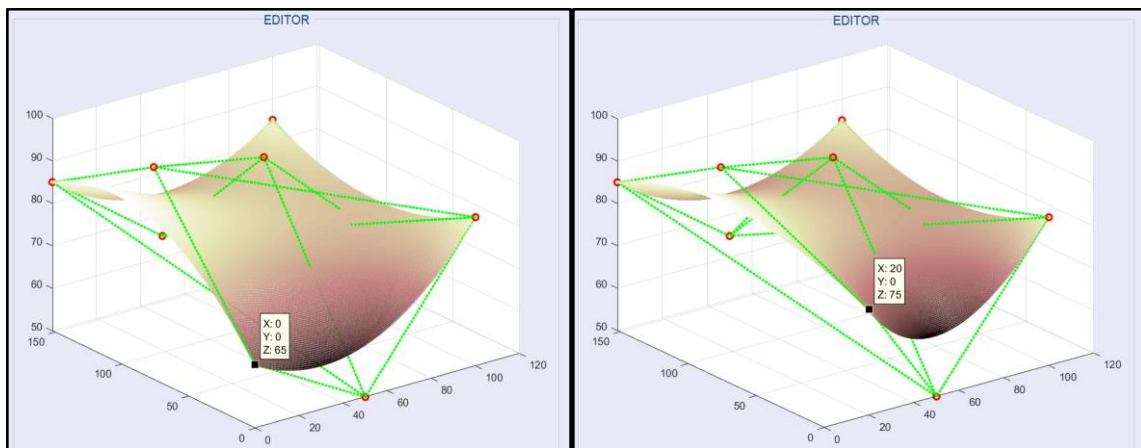


## 7.3 Bezier Surface

First, have to insert all the control points location and click Draw Bezier Curve button. One clicked, a Bezier surface will be shown in Editor panel.

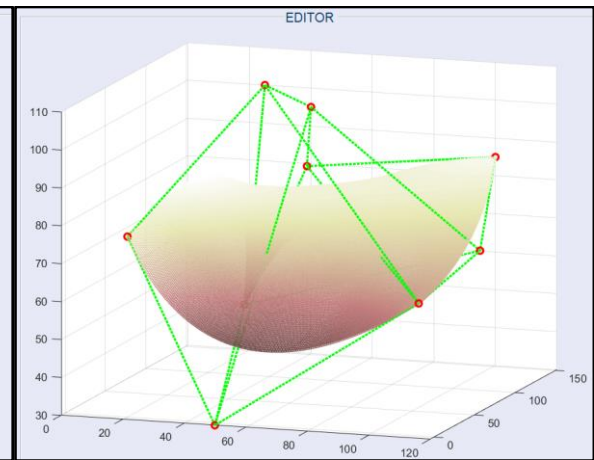
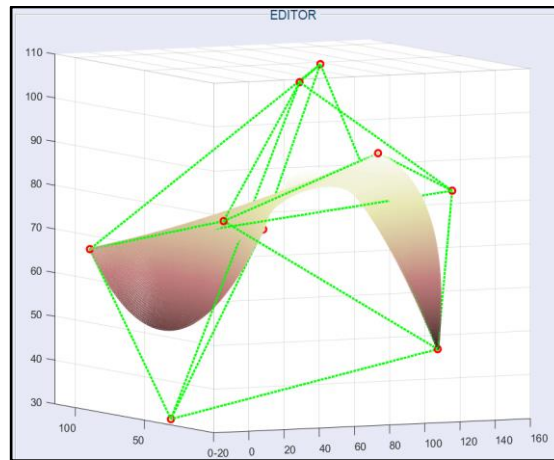
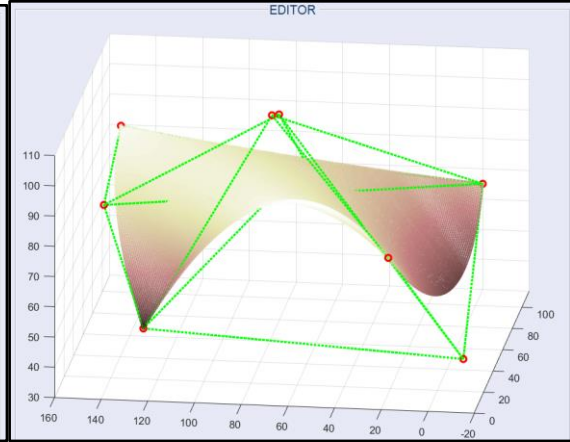
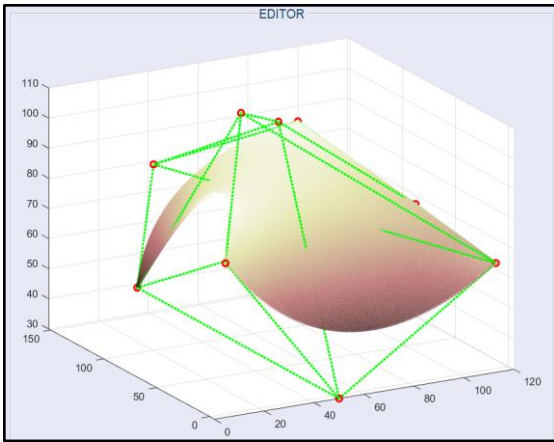


### A. Adjust Control Point Location



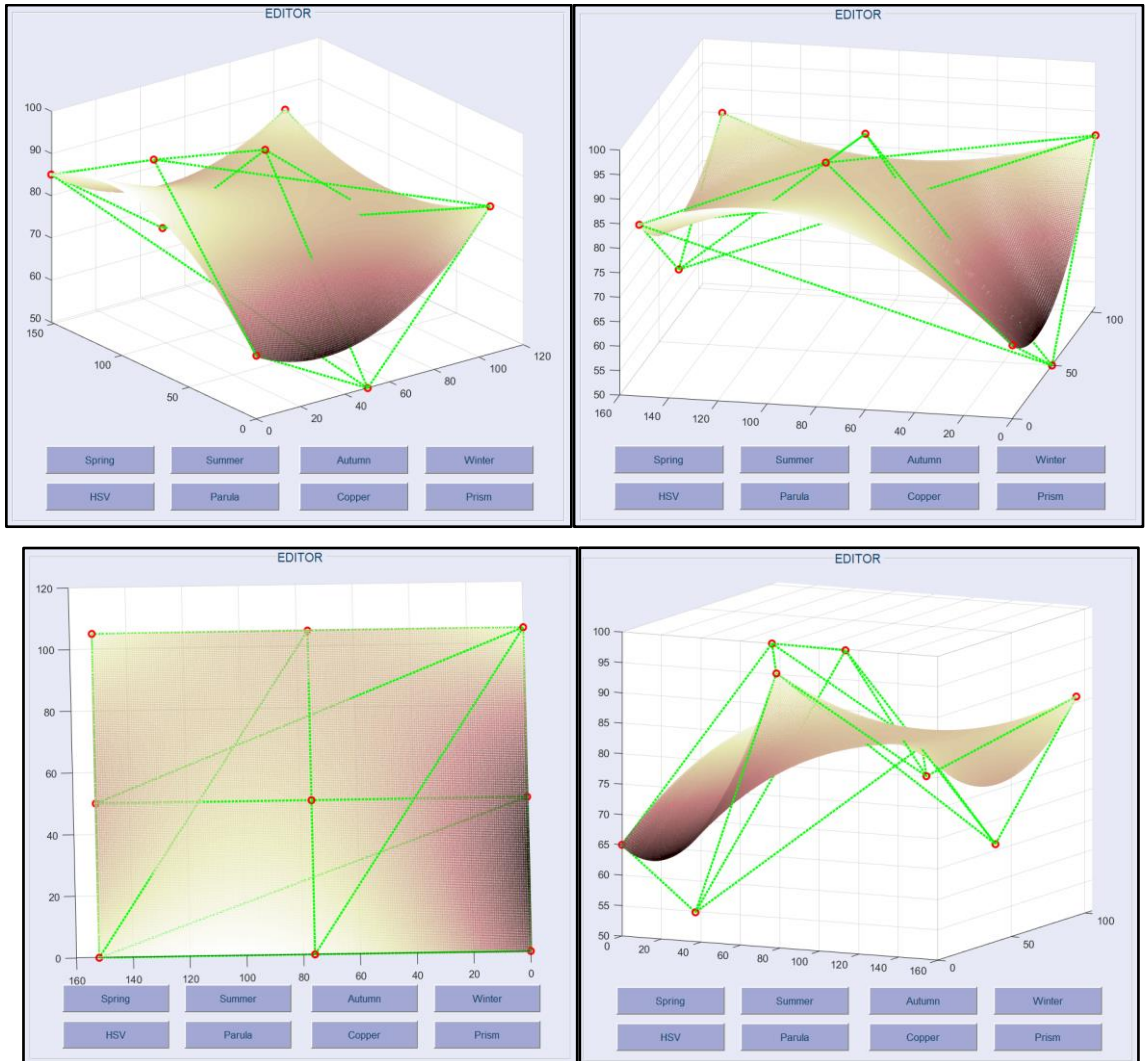
In this case, control point (1,1) has been chosen. **Data Cursor** tool had been used to show the location of the control point. The Change Difference had been set to 5, which is the default value. Then, X coordinate had been adjusted from 0 to 20, Y coordinate remain unchanged and Z coordinate adjusted from 66 to 75. It is clear that the point had been adjusted compared to the original location.

After that, a few adjustments had been made and the final output is shown below (from different perspectives).



## B. Rotate 360 degrees

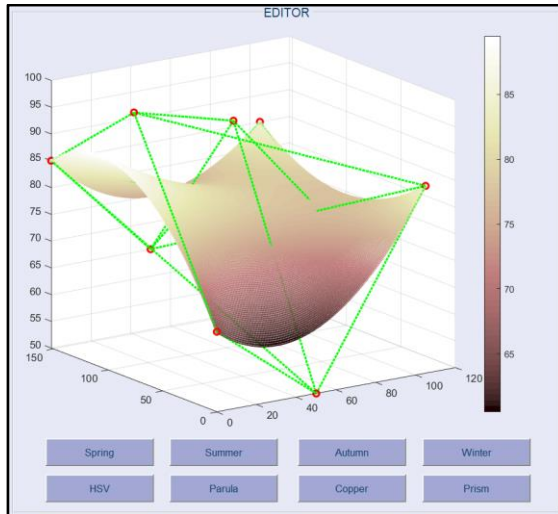
Besides, the **Rotate 3D** tool can be used to rotate the model in 360 degrees to have different perspective. Therefore, this application providing a convenient way for the user to have a better and clear understanding about the model.



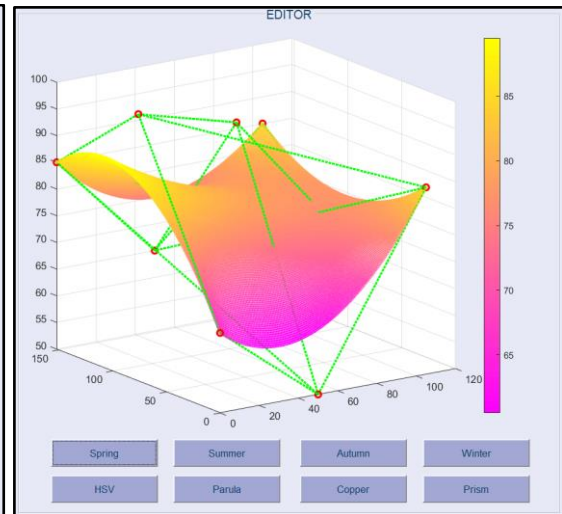


### C. Changing Color

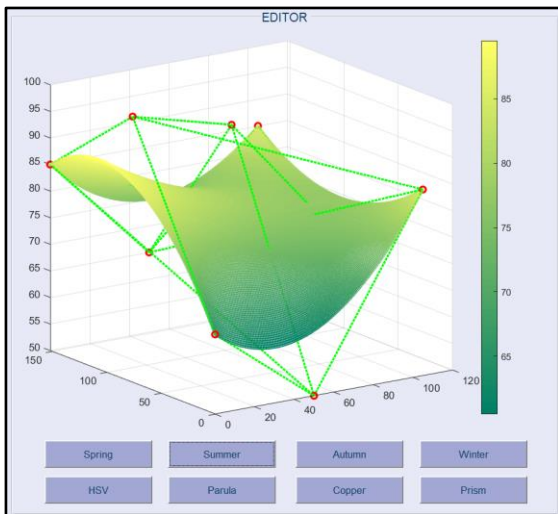
To have a better illustration, this application provided several colors for the user to choose. The default color for the model is pink. Besides, by clicking the **Color Bar** tool, a color bar will be shown.



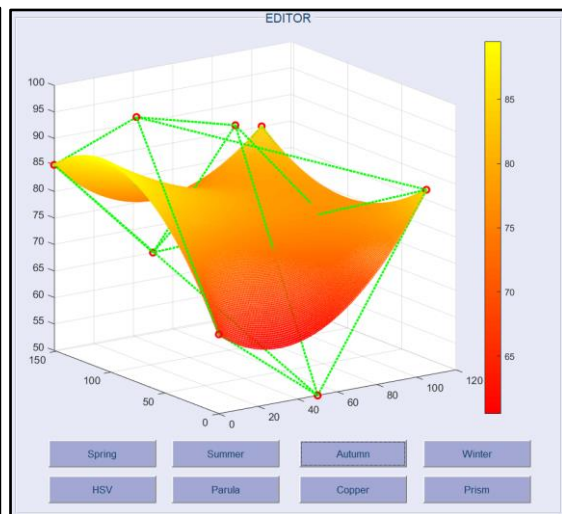
Default colour: pink



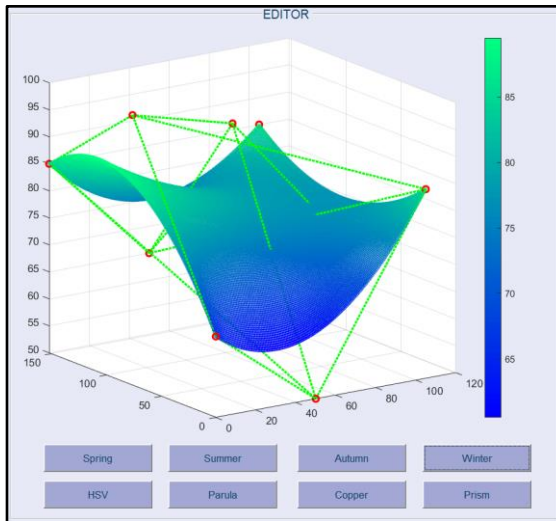
Colour: Spring



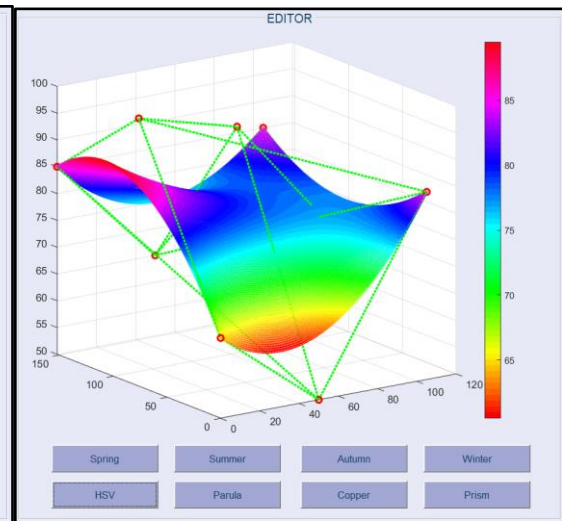
Colour: Summer



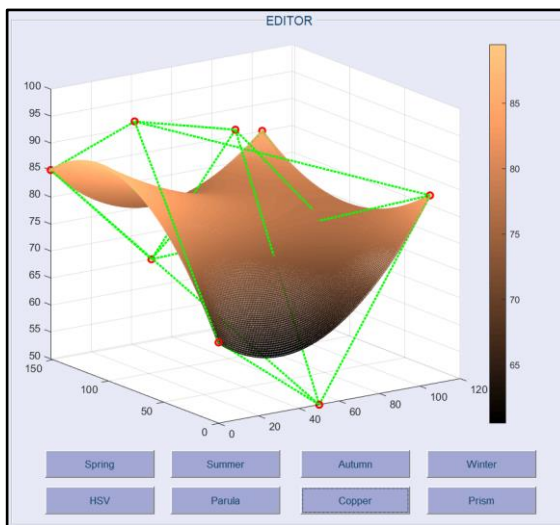
Colour: Autumn



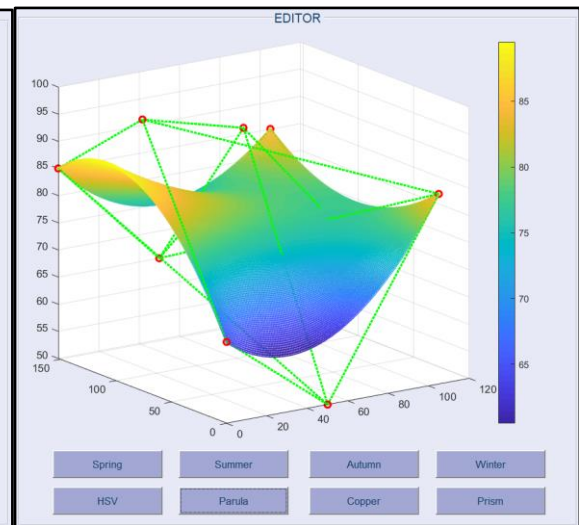
Colour: Winter



Colour: HSV

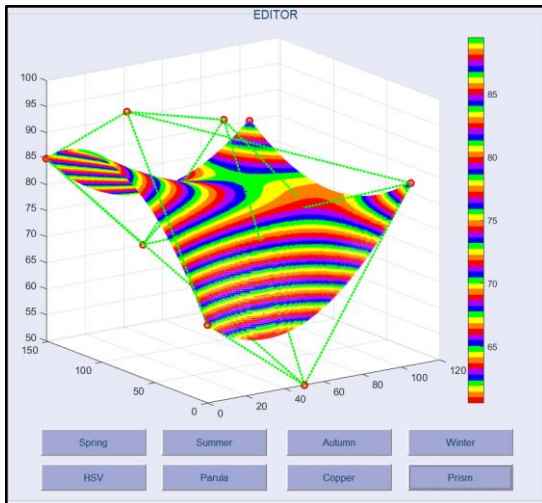


Colour: Copper



Colour: Parula





Colour: Prism

## 8. CONCLUSION

In conclusion, the project really helped us to discover a lot of things. Matlab is a software that are capable of doing mathematical computing, we took the first step in learning it and applying the bezier curve function. Despite the lack of knowledge towards graphic implementation in Matlab, we managed to create an application that is able to modify the 3D bezier curve and 3D bezier surface. However, there are still a lot of things that still need improvement. In the process, we are glad our teammate did a well done in committing to the project.

Our project concluded that it is possible to make 3D bezier curve even in the Matlab too using the equation. However, it proved to be challenging to implement it. Using other software like OpenGL or WebGL may be much easier compared to using Matlab due to the limitation on some function in Matlab.

The project's application weaknesses included that the user are not able to add new control points on the surfaces. Drag on the 3D bezier curve and surface may also not be possible. The GUI in Matlab are rather simple, but the implementation of the interactivity proves to be quite complicated compared to other software. Besides, when moving the control point of the surface or curve, once the model is being redrawn, the camera are then reset to default. We couldn't find a way to make the camera to stay on the current position as there was no camera input in the first place. We hope to find a way to improve the application in the future once we had discovered and learned even more about this software.

## 9. ACKNOWLEDGEMENT

In preparation for this project, we had to appreciate a few people who deserve our deepest gratitude. First, we would like to show our gratitude to our dearest lecturer, Madam Norhaida bte Mohd Suaib, for giving us a chance to explore the concept of 2D curve, 3D curve and also the extension from curve line to curve surface. Through this project, we had enhanced our understanding and knowledge especially in curve geometry. Besides, we would also like to thank those who have helped us directly and indirectly along with this project.

Finally, thanks to the classmates and team member who had made the greatest effort and high cooperation throughout the whole process, without the help, this project would not be completed perfectly on time.

## REFERENCE

- American Mathematical Society (2008) *From Bézier to Bernstein*, Available at: <http://www.ams.org/publicoutreach/feature-column/fcarc-bezier>(Accessed: ).
- Darren Meyer () *2-D Curve Generation*, Available at: <https://web.cs.wpi.edu/~matt/courses/cs563/talks/curves.html>(Accessed: ).
- Jonathon, A. & Jackson, M. (2014) 'On the Spline: A Brief History of the Computational Curve (Full)', *International Journal of Interior Architecture + Spatial Design, Applied Geometries* , (), pp. Retrieved from .<http://www.alatown.com/spline-history-architecture/>
- Nicholas (2015) *What is Matlab – Where and Why to Use It*, Available at: <https://www.matlabtips.com/what-is-matlab-where-how-and-when-to-use-it/>(Accessed: ).
- (2018) *Bezier Splines*, Available at: <https://docs.microsoft.com/en-us/windows/win32/gdiplus/-gdiplus-bezier-splines-about?redirectedfrom=MSDN>(Accessed: ).