

# Curva de Casteljau e Bézier

Oziel Ramos de Lima Junior  
Modelagem Geométrica

UNIVERSIDADE FEDERAL DO AMAZONAS  
INSTITUTO DE CIÊNCIAS EXATAS E DA TERRA  
DEPARTAMENTO DE MATEMÁTICA  
CURSO DE VERÃO

2023



# Sumário

## 1 Introdução

## 2 Curva de Casteljau e Bézier

- Teste
- Elaboração do Código
  - Bibliotecas Padrões
  - Código
  - Polígono de controle com os pontos no parâmetro  $t$
  - Curva de Bézier

## 3 Classe Final

## 4 Referências

- Links



# Introdução

Este trabalho tem o objetivo de realizar a implementação da curva de casteljau e bézier, utilizando linguagem de programação python 3.10.9. Foram utilizados as bibliotecas:

- Matplotlib 3.6.3
- Numpy 1.24.1

Todos os códigos utilizados e histórico de construção deles está contido no link do GitHub



# GitHub

The screenshot shows the GitHub repository page for 'ozilejunior / GM'. The repository is public and has 11 commits. The file list includes 'examples', 'pictures', 'python\_project', '.gitattributes', 'python-version', 'LICENSE', 'README.md', and 'req.txt'. The 'README.md' file is selected, showing the title 'GM' and a description: 'Anotações e desenvolvimentos de códigos do curso de verão em modelagem geométrica. Aqui estão alguns exemplos de códigos que estudamos durante a aula.' The repository also has a 'About' section with 'Geometric models' and a 'Languages' section showing 'Python 100.0%'.

ozilejunior / GM (Public)

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file + Code +

ozilejunior 2 hours ago 11 commits

File	Commit	Time
examples	Alt	3 days ago
pictures	Alt	2 hours ago
python_project	Alt	2 hours ago
.gitattributes	Initial commit	2 weeks ago
python-version	atualizações iniciais	2 weeks ago
LICENSE	Initial commit	2 weeks ago
README.md	atualizações iniciais	2 weeks ago
req.txt	Alt	2 days ago

README.md

## GM

Anotações e desenvolvimentos de códigos do curso de verão em modelagem geométrica. Aqui estão alguns exemplos de códigos que estudamos durante a aula.

About

Geometric models

python python3 matplotlib

Readme

MIT license

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

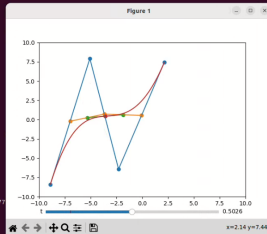
Python 100.0%

© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About



# Teste da Curva

```
new
[[[-0.99504091 -0.24498939]
 [-3.68804732 0.74871743]]]
[[[-0.99504091 -0.24498939]
 [-3.68804732 0.74871743]]]
[[[-0.99504091 -0.24498939]
 [-3.68804732 0.74871743]]]
[[[-0.99504091 -0.24498939]
 [-3.68804732 0.74871743]]]
new
[[[-5.34115236 0.25210289]
 [-5.34115236 0.25210289]]]
Valor atualizado
3
[[[-0.91129032 -0.41991342]
 [-5.08064516 7.92207792]]]
[[[-2.2983871 -6.41774892]
 [-0.91129032 -0.41991342]]]
[[[-5.08064516 7.92207792]
 [-2.2983871 -6.41774892]]]
new
[[[-0.98593871 -0.20609859]
 [-3.68222608 0.7145914]]]
[[[-0.98593871 -0.20609859]
 [-3.68222608 0.7145914]]]
[[[-0.98593871 -0.20609859]
 [-3.68222608 0.7145914]]]
[[[-0.98593871 -0.20609859]
 [-3.68222608 0.7145914]]]
new
[[[-5.32542205 0.25665879]
 [-5.32542205 0.25665879]]]
MouseButton_RIGHT
click x 2.137096774193548
click y 7.435864935064936
Color curva
[[[-0.91129032 -0.41991342]
 [-5.08064516 7.92207792]]]
a
[[[-0.91129032 -0.41991342]
 [-5.08064516 7.92207792]]]
[[[-2.2983871 -6.41774892]
 [-0.91129032 -0.41991342]]]
[[[-5.08064516 7.92207792]
 [-2.2983871 -6.41774892]]]
new
[[[-0.98593871 -0.20609859]
 [-3.68222608 0.7145914]]]
[[[-0.98593871 -0.20609859]
 [-3.68222608 0.7145914]]]
[[[-0.98593871 -0.20609859]
 [-3.68222608 0.7145914]]]
[[[-0.98593871 -0.20609859]
 [-3.68222608 0.7145914]]]
new
[[[-5.32542205 0.25665879]
 [-5.32542205 0.25665879]]]
[[[-5.32542205 0.25665879]
 [-5.32542205 0.25665879]]]
[[[-1.80615742 0.62932874]
 [-1.80615742 0.62932874]]]
[[[-5.32542205 0.25665879]
 [-1.80615742 0.62932874]]]
new
[[[-3.5867258 0.44397823]]]
[[[-3.5867258 0.44397823]]]
```



# Bibliotecas Padrões

É possível encontrar na web várias exemplos de códigos da curva de bézier

The screenshot displays a GitHub repository for a Python library named 'Bezier.py'. The repository interface includes a file list at the top with 'README.md' (updated 'last year') and 'examples.py' (updated '3 years ago'). The main content area shows the 'README.md' file, which contains the title 'Bezier.py', a subtitle 'Create Bezier curves in Python', and a list of details: 'Version: 1.0', 'Project page: <https://src.ht/~torresjrjr/Bezier.py>', and 'Author: @torresjrjr'. Below this, a section titled 'Preview plots with matplotlib' features a grid of six plots. These plots include: a 2D curve with control points, a heart-shaped curve, a 2D curve with a different set of control points, a 3D curve, a 3D plot of a 14-point Bézier curve, and another 2D curve. The right sidebar shows repository statistics: 'GPL-3.0 license', '40 stars', '2 watching', and '21 forks'. A 'Languages' section indicates 'Python 100.0%'. At the bottom left of the repository view, the text '14-point 3D Bezier curve:' is visible.



# Bibliotecas Padrões

Tal como exercícios de faculdade

The screenshot shows a GitHub repository interface. At the top, there are tabs for 'master' (selected), '1 branch', and '0 tags'. Below this is a table of file history with columns for file name, commit message, and time ago. The files listed are: 'imagens', '.gitignore', 'LICENSE.md', 'README.md', 'bezier.ipynb', 'bezier.py', 'interpolated.ipynb', and 'links.txt'. The 'README.md' file is selected, showing its content. The README is in Portuguese and describes a project for a course on Computer Graphics, focusing on Bézier curves and interpolation. It lists the tools used: Python 3.6, NumPy (1.13.3), Matplotlib (2.1.0), and Jupyter notebooks. The repository also has an 'About' section on the right, stating it's for 'Development of Bezier and Linear Interpolation curve fitting in python', and a 'Releases' section with no releases published. The 'Languages' section shows 'Jupyter Notebook' at 99.3% and 'Python' at 0.7%.

File	Commit	Time Ago
imagens	Final commit	5 years ago
.gitignore	Final commit	5 years ago
LICENSE.md	Initial commit	5 years ago
README.md	Merge remote-tracking branch 'origin/master'	5 years ago
bezier.ipynb	Updating before local deletion	5 years ago
bezier.py	Added bezier curve	5 years ago
interpolated.ipynb	Updating before local deletion	5 years ago
links.txt	Final commit	5 years ago

**Curvas Paramétricas**

Trabalho proposto na disciplina de Computação Gráfica. Propõe a implementação dos algoritmos de curva de Bézier e Interpolação.

**Ferramentas utilizadas**

- Python 3.6
- NumPy (1.13.3)
- Matplotlib (2.1.0)
- Jupyter notebooks

**Implementação**



# Bibliotecas Padrões

Existe um comando para a curva de bézier na biblioteca matplotlib. Na sua documentação, é possível ver que o algoritmo elaborado utiliza o algoritmo de Casteljau

The screenshot shows the official documentation for the `matplotlib.bezier` module. The page is titled "matplotlib.bezier" and describes it as a module providing utility functions for Bézier path manipulation. It details the `BezierSegment` class, which takes control points as input and provides methods to find intersections, extrema, and other geometric properties. The documentation also lists various properties of the segment, such as its degree and dimension.

**matplotlib.bezier**

A module providing some utility functions regarding Bézier path manipulation.

```
class matplotlib.bezier.BezierSegment(control_points) [source]
```

Bases: `object`

A d-dimensional Bézier segment.

**Parameters:**

- `control_points` : `(N, d)` array  
Location of the  $N$  control points.

**axis\_aligned\_extrema()** [source]

Return the dimension and location of the curve's interior extrema.

The extrema are the points along the curve where one of its partial derivatives is zero.

**Returns:**

- `dims` : array of int  
Index  $i$  of the partial derivative which is zero at each interior extrema.
- `dzeros` : array of float  
Of same size as `dims`. The  $t$  such that  $d/dx_i B(t) = 0$

**property control\_points**

The control points of the curve.

**property degree**

Degree of the polynomial. One less the number of control points.

**property dimension**

The dimension of the curve.

**On this page**

- `BezierSegment`
- `NonIntersectingPathException`
- `check_if_parallel()`
- `find_bezier_t_intersecting_wit`
- `find_control_points()`
- `get_cos_sin()`
- `get_intersection()`
- `get_normal_points()`
- `get_parallel()`
- `inside_circle()`
- `make_wedged_bezier2()`
- `split_bezier_intersecting_wit`
- `split_de_casteljau()`
- `split_path_inout()`



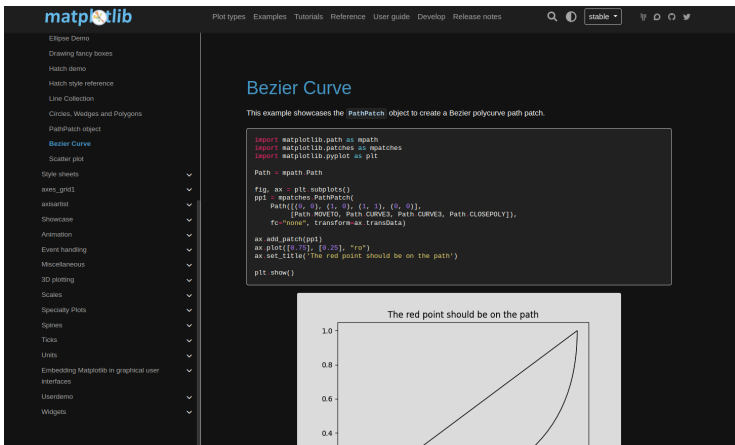
# Bibliotecas Padrões

Existe um comando para a curva de bézier na biblioteca matplotlib. Na sua documentação, é possível ver que o algoritmo elaborado utiliza o algoritmo de Casteljau

```
87 # http://www.fh-muenchen.de/~daniel.homonka/teaching/graphics/graphics/graphics.html
88
89
90 def _de_casteljau(beta, t):
91     next_beta = beta[:-1] * (1 - t) + beta[1:] * t
92     return next_beta
93
94
95 def split_de_casteljau(beta, t):
96     """
97     Split a Bézier segment defined by its control points "beta" into two
98     separate segments divided at "t" and return their control points.
99     """
100     beta = np.asarray(beta)
101     beta_list = [beta]
102     while True:
103         beta = _de_casteljau(beta, t)
104         beta_list.append(beta)
105         if len(beta) == 1:
106             break
107     left_beta = [beta[0] for beta in beta_list]
108     right_beta = [beta[-1] for beta in reversed(beta_list)]
109
110     return left_beta, right_beta
111
112
113 def find_bezier_t_intersecting_with_closedpath(
114     bezier_point_at_t, inside_closedpath, t0=0., t1=1., tolerance=0.01):
115     """
116     Find the intersection of the Bézier curve with a closed path.
117
118     The intersection point "ti" is approximated by two parameters "t0", "t1"
119     such that "t0" <= "t" <= "t1".
120
121     Search starts from "t0" and "t1" and uses a simple bisection algorithm
122     therefore one of the end points must be inside the path while the other
123     doesn't. The search stops when the distance of the points parametrized by
124     "t0" and "t1" gets smaller than the given "tolerance".
125
126     Parameters
127     ----------
128     bezier_point_at_t : callable
129         A function returning x, y coordinates of the Bézier at parameter "t".
130         It must have the signature:
```

# Bibliotecas Padrões

E até mesmo montar a curva apenas com comandos python e biblioteca matplotlib



The screenshot shows the matplotlib website's documentation for the Bezier Curve. The left sidebar lists various topics, with 'Bezier Curve' highlighted. The main content area is titled 'Bezier Curve' and includes a description: 'This example showcases the `PathPatch` object to create a Bezier polycurve path patch.' Below this is a code block containing the following Python code:

```
import matplotlib.path as mpath
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt

Path = mpath.Path

fig, ax = plt.subplots()
ppi = mpatches.PathPatch(
    Path([(0, 0), (1, 0), (1, 1), (0, 0)],
        [Path.MOVETO, Path.CURVE3, Path.CURVE3, Path.CLOSEPOLY]),
    fc="none", transform=ax.transData)

ax.add_patch(ppi)
ax.plot([0.75], [0.25], "ro")
ax.set_title('The red point should be on the path')
plt.show()
```

Below the code is a plot titled 'The red point should be on the path'. The plot shows a square with a curved path from (0,0) to (1,1) and a red dot at (0.75, 0.25). The y-axis is labeled from 0.4 to 1.0.

# Ideia

Como conversado em aula, a construção do código foi organiza-lo de maneira separada. A partir de uma construção. Dessa maneira:

- 1 Foi criado uma janela de pontos com o matplotlib;



# Ideia

- 1 Foi criado uma janela de pontos com o matplotlib;



# Ideia

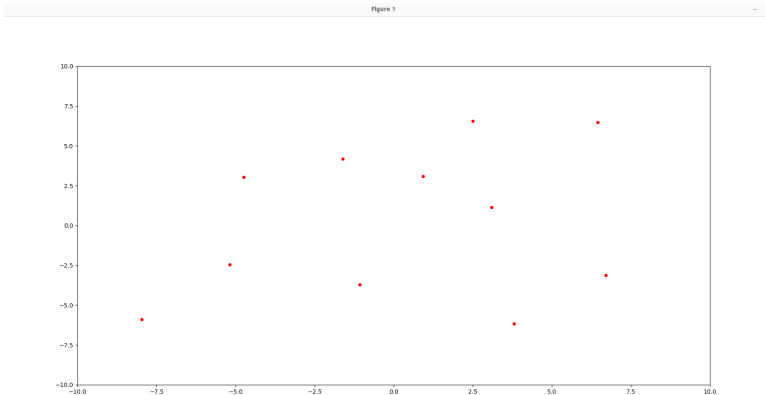
- 1 Foi criado uma janela de pontos com o matplotlib;

```
1 import matplotlib.pyplot as plt
2
3 # Create a figure
4 fig = plt.figure()
5
6 # Create an empty Cartesian plane
7 ax = fig.add_subplot(111)
8
9 # Set the limits of the plane
10 ax.set_xlim([-10, 10])
11 ax.set_ylim([-10, 10])
12
13 # Plot the plane
14 ax.plot()
15
16 # Function that creates a point when a mouse click is detected
17 def onclick(event):
18     print('click', event)
19     plt.scatter(event.xdata, event.ydata, color = 'red', s = 20)
20     plt.show()
21
22 # Connect the function to the plot
23 cid = fig.canvas.mpl_connect('button_press_event', onclick)
24
25 # Show the plot
26 plt.show()
```



# Ideia

- 1 Foi criado uma janela de pontos com o matplotlib;



# Ideia

- 1 Foi criado uma janela de pontos com o matplotlib;

Arquivo encontrado na pasta examples, arquivo examples3.py



# Ideia

- 2 Em seguida, a janela foi ajustada para receber entradas de pontos. E com isso, com os pontos criados, cria-se segmentos. Portanto, cria-se um polígono de controle.





# Ideia

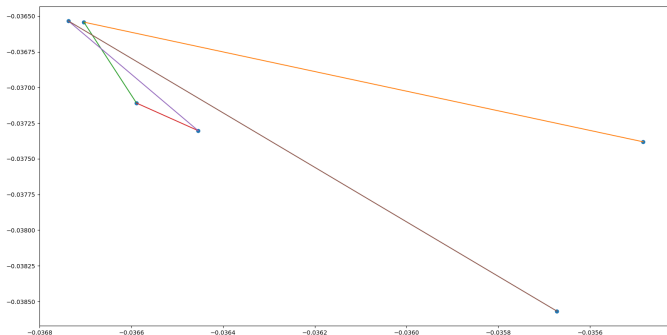
- 2 Em seguida, a janela foi ajustada para receber entradas de pontos. E com isso, com os pontos criados, cria-se segmentos. Portanto, cria-se um polígono de controle.

```
1 import matplotlib.pyplot as plt
2
3 class Point:
4     def __init__(self, x, y):
5         self.x = x
6         self.y = y
7
8 class Line:
9     def __init__(self, point1, point2):
10         self.point1 = point1
11         self.point2 = point2
12
13 points = []
14 lines = []
15
16 def on_click(event):
17     point = Point(event.xdata, event.ydata)
18     points.append(point)
19     if len(points) > 1:
20         line = Line(points[-2], points[-1])
21         lines.append(line)
22     plt.cla()
23     for line in lines:
24         plt.plot([line.point1.x, line.point2.x], [line.point1.y, line.point2.y])
25     plt.scatter([point.x for point in points], [point.y for point in points])
26     plt.show()
27
28 plt.connect("button_press_event", on_click)
29 plt.show()
```



# Ideia

- 2 Em seguida, a janela foi ajustada para receber entradas de pontos. E com isso, com os pontos criados, cria-se segmentos. Portanto, cria-se um polígono de controle.



# Ideia

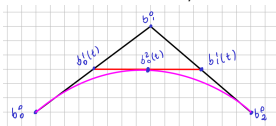
- 2 Em seguida, a janela foi ajustada para receber entradas de pontos. E com isso, com os pontos criados, cria-se segmentos. Portanto, cria-se um polígono de controle.

Arquivo encontrada na pasta examples, arquivo example8.py



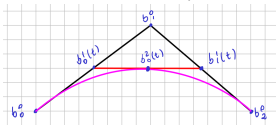
# Ideia

- 3 Agora, é necessário trabalhar um pouco com a definição da curva de bézier. Para isto, olhemos a definição a partir da imagem.



# Ideia

- 3 Agora, é necessário trabalhar um pouco com a definição da curva de bézier. Para isto, olhemos a definição a partir da imagem.

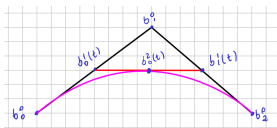


$$b_0^1(t) = (1-t)b_0^0 + tb_1^0 \quad (1)$$

$$b_1^1(t) = (1-t)b_1^0 + tb_2^0 \quad (2)$$

# Ideia

- 3 Agora, é necessário trabalhar um pouco com a definição da curva de bézier. Para isto, olhemos a definição a partir da imagem.



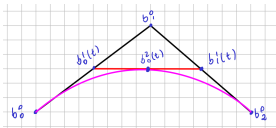
$$b_0^1(t) = (1-t)b_0^0 + tb_1^0 \quad (1)$$

$$b_1^1(t) = (1-t)b_1^0 + tb_2^0 \quad (2)$$

$$b_0^2(t) = (1-t)b_0^1(t) + tb_1^1(t) \Rightarrow$$

# Ideia

- 3 Agora, é necessário trabalhar um pouco com a definição da curva de bézier. Para isto, olhemos a definição a partir da imagem.



$$b_0^1(t) = (1-t)b_0^0 + tb_1^0 \quad (1)$$

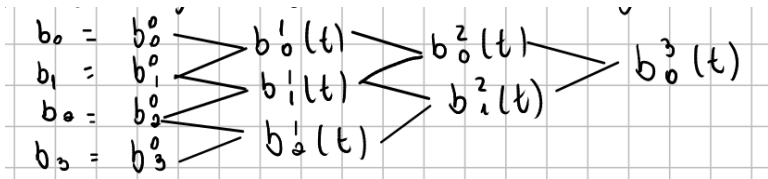
$$b_1^1(t) = (1-t)b_1^0 + tb_2^0 \quad (2)$$

$$b_0^2(t) = (1-t)b_0^1(t) + tb_1^1(t) \Rightarrow$$

$$b_0^2(t) = (1-t)^2 b_0^0 + 2t(1-t)b_1^0 + t^2 b_2^0$$

# Ideia

- 3 Para o cálculo de  $b_0^n(t)$ ,  $n = 3$ . Ou seja, curva de bézier de grau 3. Temos





# Ideia

- 3** Portanto, o algoritmo que cria o polígono de construção deve realizar os mesmos processos, ou algo parecido com os cálculos. Assim, foi construído um algoritmo que depende da entrada  $t$  e os pontos dados.



# Curva de Casteljau e Bézier

## Curva de Casteljau e Bézier

### Elaboração do Código

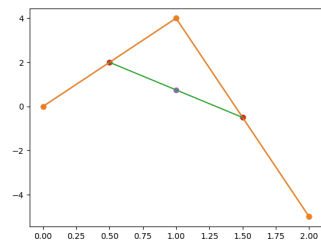
example13.py X

init.py

example14.py

trash.py U

examples > example13.py > ...  
1 import numpy as np  
2 import matplotlib.pyplot as plt  
3  
4 t = float(input('Insira t: '))  
5 control\_points = eval(input("Insira os pontos: "))  
6 control\_points = np.array(control\_points)  
7 tam = len(control\_points)  
8 plt.scatter(control\_points[:,0], control\_points[:,1])  
9 plt.plot(control\_points[:,0], control\_points[:,1])  
10 n = 1  
11  
12 while tam > n:  
13 if tam > n:  
14 name = "control\_points" + str(n)  
15 name = control\_points  
16 print(n)  
17 print(name)  
18 list = []  
19 for i in range(0, len(name) - 1):  
20 simple\_array = name[i] + t\*(name[i+1]-name[i])  
21 list.append(simple\_array)  
22 control\_points = np.array(list)  
23 print("new")  
24 print(control\_points)  
25 plt.scatter(name[:,0], name[:,1])  
26 plt.plot(name[:,0], name[:,1])  
27 n = n + 1  
28 plt.scatter(control\_points[:,0], control\_points[:,1])  
29 plt.show()  
30

Figure 1  
  
Figure 1 shows a plot of the Bézier curve construction process. The x-axis ranges from 0.00 to 2.00, and the y-axis ranges from -4 to 4. The plot displays four control points (blue dots) at (0,0), (0.5,2), (1,4), and (1.5,-1). An orange line connects the first and last control points. A green line segment connects the first and second control points. A red line segment connects the second and third control points. A blue line segment connects the third and fourth control points. The plot illustrates the iterative process of the de Casteljau algorithm.

PROBLEMAS

SAÍDA

CONSOLE DE DEBURAÇÃO

TERMINAL

(venv) oziel@DarkCover:~/Documentos/Mestrado/UFAM/Verao/Modelagem\_2023/GM\$ cd /home/oziel/Documentos/Mestrado/UFAM/Verao/Modelagem\_2023/GM ; /usr/bin/env /home/oziel/Documentos/Mestrado/UFAM/Verao/Modelagem\_2023/GM/venv/bin/python /home/oziel/.vscode/extensions/ms-python.python-2022.20.2/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 42279 -- /home/oziel/Documentos/Mestrado/UFAM/Verao/Modelagem\_2023/GM/examples/example13.py  
Insira t: 0.5  
Insira os pontos: [0,0],[1,4],[2,-5]  
1  
[[ 0 0]  
[ 1 4]  
[ 2 -5]]  
new  
[[ 0.5 2. ]  
[ 1.5 -0.5]]  
2  
[[ 0.5 2. ]  
[ 1.5 -0.5]]  
new  
[[ 1. 0.75]]  
[]

bash

Python Debug Console



Este código pode ser encontrado na pasta example, arquivo example13.py



Para elaborar a curva de bézier, basta pega o último algoritmo criado e realizar o cálculo dos pontos com o  $t$  variando de 0 a 1. Nesse exemplo, foram criados 50 valores de  $t$ .



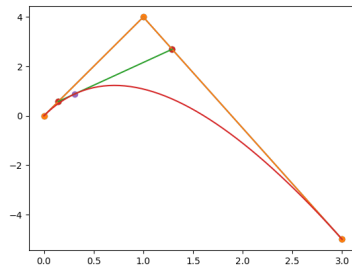
```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  t = float(input('Insira t: '))
5  control_points = eval(input("Insira os pontos: "))
6  control_points = np.array(control_points)
7  tam = len(control_points)
8  plt.scatter(control_points[:,0], control_points[:,1])
9  plt.plot(control_points[:,0], control_points[:,1])
10 n = 1
11 controle = control_points
12
13 while tam > n:
14     if tam > n:
15         name = "control_points" + str(n)
16         name = control_points
17         print(n)
18         print(name)
19         list = []
20         for i in range(0, len(name) - 1):
21             simple_array = name[i] + (t*(name[i+1]-name[i]))
22             list.append(simple_array)
23         control_points = np.array(list)
24         print("new")
25         print(control_points)
26         plt.scatter(name[:,0], name[:,1])
27         plt.plot(name[:,0], name[:,1])
28         n = n + 1
29     plt.scatter(control_points[:,0], control_points[:,1])
```

```

31 t1 = np.linspace(0,1, 50)
32 #print(t1)
33 final = []
34 for t in t1:
35     control_points = controle
36     n = 1
37     while tam > n:
38         if tam > n:
39             name = "control_points" + str(n)
40             name = control_points
41             #print(n)
42             #print(name)
43             #print("t: ", t)
44             list = []
45             for i in range(0, len(name) - 1):
46                 simple_array = name[i] + (t*(name[i+1]-name[i]))
47                 list.append(simple_array)
48             control_points = np.array(list)
49             #print("new")
50             n = n + 1
51         final.append(control_points)
52     final1 = []
53     for j in range(0, len(final)):
54         for k in range(0, len(final[j])):
55             final1.append(final[j][k])
56     final = np.array(final1)
57     plt.plot(final[:,0], final[:,1])
58
59 plt.show()

```

Figure 1



PROBLEMAS SAÍDA CONSOLE DE DEBURAÇÃO **TERMINAL**

```

(venv) oziel@DarkCover:~/Documentos/Mestrado/UFAM/Verão/Modelagem_2023/GM$ cd /home/oziel/Documentos/Mestrado/UFAM/Verão/Modelagem_20
23/GM ; /usr/bin/env /home/oziel/Documentos/Mestrado/UFAM/Verão/Modelagem_2023/GM/venv/bin/python /home/oziel/.vscode/extensions/ms-py
thon.python-2022.20.2/pythonFiles/lib/python/debugpy/adapter/.../debugpy/launcher 50909 -- /home/oziel/Documentos/Mestrado/UFAM/Verã
o/Modelagem_2023/GM/examples/example14.py
Insira t: 0.145
Insira os pontos: [0,0],[1,4],[3,-5]
1
[[ 0  0]
 [ 1  4]
 [ 3 -5]]
new
[[0.145 0.58 ]
 [1.29 2.695]]
2
[[0.145 0.58 ]
 [1.29 2.695]]
new
[[0.311025 0.886675]]

```

bash

Python Debug C



# Classe Final

Portanto, o trabalho final consiste na junção de todos os códigos citados em um único arquivo `.py`. Foi elaborado um arquivo `classep.py` que consiste no projeto final e pode ser encontrado na pasta `python_project`, arquivo `classep.py`. No perfil Projeto da Curva.



# Links

- Projeto da Curva
- Código da documentação da Curva de Bézier do Matplotlib
- Documentação da Curva de Bézier do Matplotlib
- Exemplo de Curva de Bézier com matplotlib
- Biblioteca da Curva de Bézier
- Exercício da Curva de Bézier

