UNIVERSIDADE FEDERAL DE RORAIMA CENTRO DE CIÊNCIA E

TECNOLOGIA BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**DCC703 – COMPUTAÇÃO GRÁFICA** (2024.2)

Data de entrega: 19/03/2025

**DISCENTES:**

**FELIPE RUBENS DE SOUSA BORGES (2020020120)**

# COMPUTAÇÃO GRÁFICA

**Relatório Curvas de Bézier**

**Boa Vista-RR**

**2025.2**

## Relatório de Curvas de Bézier

Relatório apresentado para o projeto de construção das Curvas de Bézier por meio do algoritmo dos algoritmos: Equação Paramétrica e de Casteljau, da disciplina de Computação Gráfica, ofertada pelo curso de Ciência da Computação da Universidade Federal de Roraima.

Prof. Luciano Ferreira

Boa Vista-RR

2025.2

Sumário

[COMPUTAÇÃO GRÁFICA 1](#_Toc193249244)

[Relatório de Curvas de Bézier 2](#_Toc193249245)

[Resumo 4](#_Toc193249246)

[Algoritmos e Fundamentação 4](#_Toc193249247)

[Algoritmo Equação Paramétrica 4](#_Toc193249248)

[Algoritmo de Casteljau 4](#_Toc193249249)

[Resultado 7](#_Toc193249250)

[Conclusão 8](#_Toc193249251)

## Resumo

Este relatório descreve a implementação de uma interface gráfica para visualização das Curvas de Bézier, sua construção e os resultados obtidos utilizando as bibliotecas Matplotlib, Numpy e Scipy, em Python. O relatório aborda dois tipos de implementação da Curva de Bézier, os algoritmos: Equação Paramétrica que utiliza os polinômios de Bernstein para calcular a curva diretamente. Já o algoritmo de Casteljau utiliza uma abordagem recursiva para subdividir a curva até que ela seja precisa. O programa tem como objetivo realizar a amostragem da Curva através de diferentes abordagens na implementação, mas resultados idênticos.

## Algoritmos e Fundamentação

## Algoritmo Equação Paramétrica

A Curva de Bézier é definida por uma fórmula matemática que combina dois pontos de controle com os Polinômios de Beirstein:

Texto

O conteúdo gerado por IA pode estar incorreto.

onde Bi,n(t)=(ni)ti(1−t)n−iB\_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}B\_{i,n}(t) = {n i} t^i (1-t)^{n-i} é o polinômio de Bernstein, PiP\_iP\_i são os pontos de controle, e ( n ) é o grau da curva (número de pontos de controle menos 1).

Assim calculando a curva diretamente para vários valores de ( t ), gerando pontos ao longo da trajetória.

## Algoritmo de Casteljau

* É um método geométrico e recursivo que subdivide os pontos de controle iterativamente até alcançar uma aproximação suave da curva.
* Utiliza interpolação linear repetida entre os pontos de controle para encontrar pontos na curva, controlando a precisão com um critério de distância.
* O processo pode ser visto como uma construção passo a passo, onde cada subdivisão refina a curva até que ela seja menor que um limiar.

Os dois métodos produzem a mesma curva de Bézier, mas diferem na abordagem: o paramétrico é analítico e contínuo, enquanto o Casteljau é iterativo e adaptável.

**No código:**

* Definição da Função **bernstein**:
* Calcula o polinômio de Bernstein Bi,n(t)=(ni)ti(1−t)n−iB\_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}B\_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}.
* Parâmetros:
  + n: Grau da curva (número de pontos de controle - 1).
  + i: Índice do ponto de controle atual.
  + t: Valor do parâmetro (array de 0 a 1).
* Retorna o valor do polinômio para cada ( t ), usado na equação paramétrica.
* Definição da Função **subdivisao\_curva**:
* Implementa a subdivisão geométrica dos pontos de controle, baseada no algoritmo de Casteljau.
* Para quatro pontosP0,P1,P2,P3P\_0, P\_1, P\_2, P\_3 P\_0, P\_1, P\_2, P\_3:
  + Calcula pontos intermediários pela média:
    - M01=(P0+P1)/2M\_{01} = (P\_0 + P\_1)/2M\_{01} = (P\_0 + P\_1)/2
    - M12=(P1+P2)/2M\_{12} = (P\_1 + P\_2)/2M\_{12} = (P\_1 + P\_2)/2
    - M23=(P2+P3)/2M\_{23} = (P\_2 + P\_3)/2M\_{23} = (P\_2 + P\_3)/2
  + Calcula pontos de segunda ordem:
    - M012=(M01+M12)/2M\_{012} = (M\_{01} + M\_{12})/2M\_{012} = (M\_{01} + M\_{12})/2
    - M123=(M12+M23)/2M\_{123} = (M\_{12} + M\_{23})/2M\_{123} = (M\_{12} + M\_{23})/2
  + Calcula o ponto médio da curva:
    - M0123=(M012+M123)/2M\_{0123} = (M\_{012} + M\_{123})/2M\_{0123} = (M\_{012} + M\_{123})/2
* Retorna duas novas curvas (primeira e segunda metades) e o ponto médioM0123M\_{0123} M\_{0123}

.

* Definição da Função **parametrica**:
* Calcula a curva de Bézier usando a **equação paramétrica**.
* Passos:
  + Define n=len(points)−1n = len(points) - 1n = len(points) - 1(grau da curva).
  + Gera 100 valores de ( t ) entre 0 e 1 com np.linspace.
  + Para cada ponto de controlePiP\_i P\_i, calcula Bi,n(t)⋅PiB\_{i,n}(t) \cdot P\_iB\_{i,n}(t) . P\_i e soma os resultados.
* Retorna um array de pontos (x, y) que formam a curva.
* Definição da Função **casteljau**:
* Implementa o algoritmo de Casteljau com subdivisão recursiva.
* Parâmetros:
  + pontos: Lista de pontos de controle.
  + u: Tolerância para decidir se a curva é "reta" o suficiente.
* Função interna **subdivisão\_recursiva**:
  + Calcula a distância entre o primeiro e o último ponto de controle.
  + Se a distância for maior que ( u ):
    - Subdivide a curva em duas metades com subdivisao\_curva.
    - Chama a si mesma recursivamente para cada metade.
  + Se menor ou igual a ( u ):
    - Adiciona o último ponto à lista de pontos da curva.
* Retorna os pontos calculados como um array NumPy.
* Definição dos **Pontos de Controle**:
* **pontos\_controle**: Array NumPy com quatro pontos ([[0, 0], [1, 3], [3, 3], [4, 0]]), definindo uma curva cúbica de Bézier.
* **pontos\_controle\_2**: Cópia idêntica para o método Casteljau.
* Geração das Curvas:
* **curva\_parametrica** = bezier\_curva(pontos\_controle): Calcula a curva com a equação paramétrica.
* **curva\_casteljau** = casteljau(pontos\_controle\_2): Calcula a curva com o algoritmo de Casteljau.

#### Implementação



Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Tela preta com letras brancas

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

## Resultado

**Gráfico, Gráfico de linhas

O conteúdo gerado por IA pode estar incorreto.**

Como resultado, foi obtida uma representação dentro do esperado, onde a Equação Paramétrica é calculada usando a equação matemática direta a partir dos polinômios de Bernstein gerando uma curva suave pois o número de pontos é definido, é uniforme e rápido e de simples implementação. Já o Casteljau é calculado recursivamente, não possui o número de pontos definidos pois eles são subdivididos, tornando-se um algoritmo adaptativo servindo para curvas mais complexas sendo mais complexo.

Em termos de comparação, para curvas simples e numpointsnum\_pointsnúmero de pontos moderado, o método paramétrico é mais eficiente, enquanto para curvas complexas ou alta precisão, Casteljau pode ser mais eficiente ao evitar pontos desnecessários, mas a recursão aumenta o custo computacional.

## Conclusão

O código apresentado implementa os algoritmos Equação Paramétrica e Casteljau, sendo ambos eficientes para gerar curvas de Bézier. No contexto do código, ambos demonstram a versatilidade das curvas de Bézier, com a visualização destacando sua equivalência prática apesar das abordagens distintas. Conclui-se que o Paramétrico é ideal para renderização rápida e uniforme (ex.: gráficos 2D pré-calculados) e o Casteljau é melhor para animações ou sistemas adaptativos, onde a curva pode ser refinada dinamicamente.