Trabalho de Grupo - CB23 - Prog 02 - 2025

Emilio Vital Brazil

Entrega: 13 de novembro de 2025

Tema

Desenvolvimento de uma Biblioteca de Cálculo Numérico em Python com Recursos Gráficos.

Sumário

1 Objetivo Geral

O objetivo deste trabalho é que cada grupo desenvolva, em linguagem **Python**, uma **biblioteca de cálculo numérico própria**, contendo funcionalidades clássicas de análise numérica e recursos gráficos para visualização de resultados. O projeto deve ser versionado e desenvolvido colaborativamente através do **GitHub**, explorando boas práticas de programação, documentação e controle de versão.

2 Organização do Trabalho

- O trabalho deve ser realizado em **grupos**, conforme listas de grupos em na seção 8.
- Cada grupo deverá criar um **repositório público no GitHub** com o nome sugerido:

  `CB2325NumericaG<NumeroDoGrupo>`

- Todos os integrantes do grupo devem ser adicionados como **colaboradores** do repositório.
- Todos os professores da disciplina devem ser adicionados com permissão de leitura e escrita.

3 Entregas e Prazos

- A versão final do código deverá estar disponível no repositório até o dia **13 de novembro de 2025**, às 23h59.
- O **grupo** deverá enviar aos professores o **hash do commit** a ser considerado para avaliação final.
- Para a avaliação individual, serão analisados apenas os commits realizados com a conta associada ao e-mail institucional do IMPA TECH, e efetuados antes do commit definido como entrega final para avaliação.
- Commits posteriores não serão considerados para nota de grupo, mas poderão ser usados para fins de demonstração ou correção.

4 Avaliação

A avaliação será composta de duas partes complementares: a nota de grupo (**NG**) e a contribuição pessoal (**CP**).

4.1 Avaliação em Grupo (NG)

Avalia a qualidade técnica e estética da biblioteca produzida, considerando:

- Estrutura e organização do código;
- Clareza e completude da documentação;
- Implementação correta e eficiente dos algoritmos;
- Qualidade dos recursos gráficos;
- Clareza e profissionalismo do repositório (README, exemplos de uso, testes).

A nota **NG** varia entre 0 e 1.

4.2 Avaliação Individual (CP)

A contribuição individual de cada aluno será medida pela análise de seus commits, levando em conta a quantidade, relevância e qualidade das contribuições.

A nota **CP** varia entre 0 e 1.

4.3 Cálculo da Nota Parcial do Trabalho (NT)

A nota individual do trabalho será calculada como:

$$NT = 3{,}0 \times NG \times CP$$

Essa nota corresponderá a **3,0 pontos** da AV2. Os outros **3,0 pontos** da AV2 serão obtidos por meio de uma **prova presencial individual**, na qual o aluno deverá demonstrar capacidade de usar a biblioteca desenvolvida pelo grupo.

Exemplo: Se um grupo obtiver 80% ($NG = 0{,}8$) e um aluno tiver contribuição de 90% ($CP = 0{,}9$), sua nota será:

$$NT = 3{,}0 \times 0{,}8 \times 0{,}9 = 2{,}2$$

Se este aluno tirar na prova individual $PI = 2{,}3$ e suas notas de listas forem $NL = 9{,}0$ sua nota final da AV2 será:

$$AV2 = 9{,}0 \times 0{,}4 + 2{,}2 + 2{,}3 = 8{,}1$$

5 Requisitos Técnicos da Biblioteca

A biblioteca deverá ser desenvolvida em **Python 3.x** e conter, no mínimo, os seguintes módulos e funcionalidades. Cada módulo deve apresentar exemplos de uso e visualizações gráficas sempre que possível.

5.1 Erros Numéricos

- Cálculo de **erro absoluto** e **erro relativo**;

Exemplo de uso:

```python
from CB2325NumericaG0.erros import erro_absoluto, erro_relativo

valor_real = 3.141592
valor_aprox = 3.14

ea = erro_absoluto(valor_real, valor_aprox)
er = erro_relativo(valor_real, valor_aprox)

print(ea, er)
# Saída esperada:
# 0.001592 0.0005067
```

5.2 Raízes de Funções

- Implementar pelo menos 2 métodos numéricos para encontrar raízes de funções reais, por exemplo:
  - Método da Bisseção;
  - Método da secante;
  - Método de Newton-Raphson.
- Criar visualização gráfica do comportamento da função e das iterações.

Exemplo de uso:

```python
from CB2325NumericaG0.raizes import raiz

f = lambda x: x**3 - 9*x + 5

raiz_0 = raiz(f, a=0, b=2, tol=1e-6, method="secante")

print(f"{raiz_0:.3f}")
# Saída esperada:
# 0.551
```

Visualização esperada: gráfico da função $f(x)$ com marcações sucessivas das aproximações do método até a raiz.

5.3 Interpolação

- Interpolação de pontos na reta real usando:
  - Interpolação Linear por Partes;
  - Interpolação Polinomial.
  - Interpolação Polinomial de Hermite.
- Representação gráfica dos pontos e do polinômio interpolador.

Exemplo de uso:

```python
from CB2325NumericaG0.interpolacao import poly_interp

x = [0, 1, 2, 3]
y = [1, 2, 0, 4]

p = poly_interp(x, y)
print(p(1.5))
# Saída esperada:
# 0.8125
```

Visualização esperada: pontos (x_i, y_i) conectados pelo polinômio interpolador contínuo.

5.4 Aproximação

- Implementar ajuste de funções aproximadoras polinomiais (ex.: **regressão linear**, **mínimos quadrados**);
- Comparar visualmente os pontos reais e a função ajustada.

Exemplo de uso:

```python
from CB2325NumericaG0.aproximacao import ajuste_linear

x = [0, 1, 2, 3, 4]
y = [1.1, 1.9, 3.0, 3.9, 5.2]

a, b = ajuste_linear(x, y)
print(f"y = {a:.2f}x + {b:.2f}")
# Saída esperada:
# y = 1.02x + 0.98
```

Visualização esperada: gráfico de dispersão dos pontos com a reta ajustada sobreposta.

5.5 Integração Numérica

- Implementar pelo menos um método de integração numérica:
- Apresentar graficamente a função e a área sob a curva.

Exemplo de uso:

```
from CB2325NumericaG0.integracao import integral
import math

f = lambda x: math.sin(x)
area = integral(f, 0, math.pi, n=100)
print(area)
# Saída esperada:
# 1.998
```

Visualização esperada: gráfico da função $\sin(x)$ entre $[0, \pi]$ com as subdivisões trapezoidais e a área sombreada.

5.6 Outros Requisitos

- Módulo gráfico utilizando `matplotlib` ou `plotly`;
- Documentação completa e arquivo `README.md` com instruções e exemplos;
- Preferencialmente, incluir testes automatizados (ex.: `pytest`);
- Organização modular do código em pacotes e submódulos (por exemplo: `erros.py`, `raizes.py`, `interpolacao.py`, `aproximacao.py`, `integracao.py`);
- Utilização de `docstrings` e boas práticas de programação (PEP8).

6 Recomendações

- Utilize boas práticas de versionamento (commits claros e frequentes);
- Organize o código de forma limpa e comentada;
- Inclua exemplos práticos e notebooks de demonstração;
- Documente todas as funções com `docstrings`.

7 Entrega Final e Prova

- Entrega do hash final: até **13/11/2025, 23h59**;
- Prova presencial individual: **18/11/2025**;
- Entrega digital da prova: formato `.py`.