Davi Ferreira de Souza

Prof. Douglas Rodrigues

Métodos Numéricos Computacionais

2 de maio de 2025

Relatório Trabalho Prático 2 - Métodos Numéricos Computacionais

Implementação

A implementação do programa foi realizada em Python, utilizando as bibliotecas matplotlib, os e tabulate, escolhidas por sua praticidade na geração de gráficos, interação via terminal e organização de dados em tabelas. O objetivo do sistema é aplicar os métodos de diferenciação numérica — progressiva, regressiva e central — a um conjunto de pontos experimentais, permitindo ao usuário analisar e comparar os valores das derivadas obtidas por cada abordagem. O programa conta com um menu interativo que facilita a navegação entre os resultados numéricos, gráficos e erros de aproximação, oferecendo uma ferramenta didática para o estudo de métodos numéricos.

Implementação dos métodos

```
def diferenciacao_regressiva(x, y):
     derivadas = [None]

     for i in range(1, len(x)):
         dx = x[i] - x[i-1]
         dy = y[i] - y[i-1]
```

```
        derivadas.append(dy / dx)

    return derivadas

def diferenciacao_central(x, y):
    derivadas = [None];

    for i in range(1, len(x) - 1):
        dx = x[i+1] - x[i-1]
        dy = y[i+1] - y[i-1]

        derivadas.append(dy / dx)

        derivadas.append(None)

    return derivadas


def diferenciacao_progressiva(x, y):
    derivadas = []

    for i in range(0, len(x) - 1):
        dx = x[i+1] - x[i]
        dy = y[i+1] - y[i]

        derivadas.append(dy / dx)

        derivadas.append(None)

    return derivadas
```

Esse trecho do código é responsável pela implementação dos métodos de diferenciação numérica: regressiva, central e progressiva. Cada função recebe dois vetores, x e y, que representam, respectivamente, os valores da variável independente e os correspondentes valores

da função. A ideia central é aproximar a derivada em pontos discretos utilizando fórmulas baseadas em diferenças finitas, comuns em análises numéricas quando os dados disponíveis são tabelados.

- Diferenciação Regressiva:

  Calcula a derivada em cada ponto usando a diferença entre o valor atual e o anterior

$$f'(x_i) \approx \frac{(y_i - y_{i-1})}{(x_i - x_{i-1})}$$

Como não existe ponto anterior para $i = 0$, o primeiro elemento do vetor de derivadas é definido como None.

A partir de $i = 1$, todos os valores têm derivada válida, pois sempre há um ponto anterior.

- Diferenciação Central:

  Combina a diferença regressiva e progressiva ao usar pontos anterior e seguinte:

$$f'(x_i) \approx \frac{(y_{i+1} - y_{i-1})}{(x_{i+1} - x_{i-1})}$$

Nos extremos da amostra ($i = 0$ e $i = n - 1$) não é possível aplicar, então ambos são marcados como None.

- Diferenciação Progressiva:

  Usa a diferença entre o próximo ponto e o atual para estimar a derivada:

$$f'(x_i) \approx \frac{(y_{i+1} - y_i)}{(x_{i+1} - x_i)}$$

Para $i = 0$, há derivada válida (pois existe $i + 1$), mas no último índice $i = n - 1$ não há ponto seguinte, então o último valor é None.

Essa estrutura torna claro onde cada método não pode ser aplicado (marcado com `None`) e preserva o alinhamento entre os vetores de entrada e saída. A implementação baseada em listas e laços `for` garante simplicidade e eficiência, facilitando a integração desses cálculos em tabelas e gráficos para análise comparativa.

Geração e salvamento dos gráficos:

```
def gerar_graficos_comparacao(valores_x, derivadas_regressivas,
derivadas_centrais, derivadas_progressivas,
derivadas_aproximadas_regressao):
      xs_reg, ys_reg = [], []
      xs_cent, ys_cent = [], []
      xs_prog, ys_prog = [], []
      xs_regressao, ys_regressao = [],[]

      for i in range(len(valores_x)):

          if derivadas_regressivas[i] != None:
              xs_reg.append(valores_x[i])
              ys_reg.append(derivadas_regressivas[i])

      if derivadas_centrais[i] != None:
            xs_cent.append(valores_x[i])
            ys_cent.append(derivadas_centrais[i])
```

```
      if derivadas_progressivas[i] != None:
            xs_prog.append(valores_x[i])
            ys_prog.append(derivadas_progressivas[i])

      xs_regressao.append(valores_x[i])
      ys_regressao.append(derivadas_aproximadas_regressao[i])


      mpl.figure(figsize=(8, 5))
      mpl.plot(xs_prog, ys_prog, marker='o', linestyle='-',
label='Progressiva')
      mpl.plot(xs_reg, ys_reg,  marker='s', linestyle='--',
label='Regressiva')
      mpl.plot(xs_cent, ys_cent, marker='^', linestyle='-.',
label='Central')
      mpl.plot(xs_regressao, ys_regressao, marker='8', linestyle =
':', label = 'Diferenciação usando função regressão')

      mpl.title("Comparação de Métodos de Diferenciação Numérica")
      mpl.xlabel("Tempo (s)")
      mpl.ylabel("Velocidade (m/s)")
      mpl.legend()
      mpl.tight_layout()
      mpl.savefig("grafico_comparacao.png")


def gerar_graficos_erro(valores_x, erros_derivadas_regressivas,
erros_derivadas_centrais, erros_derivadas_progressivas):
      xs_reg, ys_reg = [], []
      xs_cent, ys_cent = [], []
      xs_prog, ys_prog = [], []

      for i in range(len(valores_x)):
          if(erros_derivadas_regressivas[i] != None):
              xs_reg.append(valores_x[i])
              ys_reg.append(erros_derivadas_regressivas[i])

      if(erros_derivadas_centrais[i] != None):
```

```
            xs_cent.append(valores_x[i])
            ys_cent.append(erros_derivadas_centrais[i])

      if(erros_derivadas_progressivas[i] != None):
            xs_prog.append(valores_x[i])
            ys_prog.append(erros_derivadas_progressivas[i])

      mpl.figure(figsize=(8, 5))
      mpl.plot(xs_prog, ys_prog, marker='o', linestyle='-',
label='Progressiva')
      mpl.plot(xs_reg, ys_reg,  marker='s', linestyle='--',
label='Regressiva')
      mpl.plot(xs_cent, ys_cent, marker='^', linestyle='-.',
label='Central')

      mpl.title("Erros em relação à regressão quadrática")
      mpl.xlabel("Tempo (s)")
      mpl.ylabel("|Aproximação - Derivada Função Regressiva|")
      mpl.legend()
      mpl.tight_layout()
      mpl.savefig("grafico_erro.png")
```

As funções gerar_graficos_comparacao e gerar_graficos_erro encapsulam toda a lógica necessária para transformar vetores de dados em gráficos prontos para análise. Cada função recebe como parâmetros listas de valores de tempo e estimativas (ou erros) de derivadas, filtra automaticamente os pontos inválidos (marcados como None) e gera figuras Matplotlib de tamanho 8×5 polegadas, aplicando estilos consistentes de marcadores e linhas antes de salvar as imagens em arquivos PNG.

Função gerar_graficos_comparacao

A função aceita cinco parâmetros:

- valores_x: lista de abscissas (por exemplo, tempos em segundos).

- derivadas_regressivas, derivadas_centrais e derivadas_progressivas: listas de derivadas numéricas, cada uma contendo None nos pontos onde o método não se aplica. derivadas_aproximadas_regressao: lista com valores de referência (aqui obtidos por ajuste quadrático).

Internamente, percorre todo o vetor valores_x e, para cada índice, verifica se a entrada correspondente em cada lista de derivadas não é None; se for válida, adiciona o par (x, y) às listas específicas de cada método. Já as derivadas de referência são todas incluídas sem filtragem. Em seguida, abre uma figura Matplotlib com plt.figure(figsize=[8,5]) para definir o tamanho em polegadas, plota cada série usando plt.plot(...) com diferentes marcadores e estilos de linha para facilitar a distinção visual, adiciona título, rótulos e legenda, executa plt.tight_layout() para ajustar automaticamente os espaços e finaliza salvando em "grafico_comparacao.png" via plt.savefig(...)

Função gerar_graficos_erro

Essa função recebe quatro listas:

- valores_x: lista de abscissas.
- erros_derivadas_regressivas, erros_derivadas_centrais e erros_derivadas_progressivas: listas de erros absolutos de cada método, com None nos pontos inválidos.

O processamento é análogo ao gráfico de comparação: filtra os None, agrupa os pares (x, erro) em listas específicas e gera uma nova figura 8×5 polegadas, empregando os mesmos padrões de marcadores e estilos para consistência. Após inserir título, eixos e legenda, aplica

tight_layout() e salva em "grafico_erro.png" usando plt.savefig. Dessa forma, ambas as funções automatizam desde a filtragem dos dados até a geração e exportação dos gráficos, garantindo reproducibilidade e clareza visual.

Função Main

```
def main():
      valores_x = [0,3,5,8,10,13]
      valores_y = [0, 225, 383, 623, 742, 993]

      derivadas_obtidas_regressao = [76.3073788546256,
76.1573788546256, 76.0573788546256, 75.9073788546256,
75.8073788546256, 75.6573788546256]

      derivadas_regressivas =
diferenciacao_regressiva(valores_x,valores_y)
      derivadas_centrais = diferenciacao_central(valores_x,
valores_y)
      derivadas_progressivas =
diferenciacao_progressiva(valores_x,valores_y)

      rodando = True

      while(rodando):
      os.system("chcp 65001")
      os.system("cls" if os.name == "nt" else "clear")

      print("--- Diferenciação Numérica ---")
      print("\nPontos analisados:")
      print(tabulate(zip(valores_x, valores_y), headers=["x", "y"],
tablefmt="rounded_grid"))

      print("\nMenu:")
      print("1 - Resultados das derivadas")
      print("2 - Gráficos de comparação")
```

```
      print("3 - Erros de aproximação")
      print("4 - Gráficos de erro")
      print("5 - Sair\n")

      try:
            escolha = int(input("Insira sua escolha: "))
      except ValueError:
            print("Entrada inválida! Digite um número entre 1 e 5.")
            input("Pressione Enter para continuar...")
            continue

      if escolha < 1 or escolha > 5:
            print("Opção inválida! Digite um número entre 1 e 5.")
            input("Pressione Enter para continuar...")
            continue

      match escolha:
            case 1:
                  os.system("cls" if os.name == "nt" else "clear")
                  dados_tabela = []
                  for i in range(len(valores_x)):
                  linha = [
                        valores_x[i],
                        f"{derivadas_regressivas[i]:.6f}" if
derivadas_regressivas[i] is not None else "N/A",
                        f"{derivadas_centrais[i]:.6f}" if
derivadas_centrais[i] is not None else "N/A",
                        f"{derivadas_progressivas[i]:.6f}" if
derivadas_progressivas[i] is not None else "N/A",
                        f"{derivadas_obtidas_regressao[i]:.6f}"
                  ]
                  dados_tabela.append(linha)
                  headers = ["Tempo (s)", "Regressiva (m/s)", "Central
(m/s)", "Progressiva (m/s)", "Função Regressão (m/s)"]

                  print("\nRESULTADOS DAS DERIVADAS NUMÉRICAS")
                  print(tabulate(
                  dados_tabela,
```

```
                headers=headers,
                tablefmt="rounded_grid",
                floatfmt=".6f",
                stralign="center",
                numalign="center"
                ))

                print("\nLegenda:")
                print("N/A = Método não aplicável neste ponto")
                print("Valores mostrados com 6 casas decimais")

                input("Aperte enter para continuar: ")

           case 2:

                gerar_graficos_comparacao(valores_x,
derivadas_regressivas, derivadas_centrais, derivadas_progressivas,
derivadas_obtidas_regressao)

                print("Gráfico gerado com sucesso!")
                input("Aperte enter para continuar: ")

           case 3:
                dados_tabela = []
                for i in range(len(valores_x)):
                linha = [
                     valores_x[i],
                     f"{abs(derivadas_regressivas[i] -
derivadas_obtidas_regressao[i]):.6f}"
                          if derivadas_regressivas[i] is not None
else "N/A",
                     f"{abs(derivadas_centrais[i] -
derivadas_obtidas_regressao[i]):.6f}"
                          if derivadas_centrais[i] is not None else
"N/A",
                     f"{abs(derivadas_progressivas[i] -
derivadas_obtidas_regressao[i]):.6f}"
                          if derivadas_progressivas[i] is not None
```

```
else "N/A"
                ]
                dados_tabela.append(linha)

                headers = ["Tempo (s)", "Erro Regressiva", "Erro
Central", "Erro Progressiva"]

                print("\nERROS DE APROXIMAÇÃO EM RELAÇÃO À
REGRESSÃO")
                print(tabulate(
                dados_tabela,
                headers=headers,
                tablefmt="rounded_grid",
                floatfmt=".6f",
                stralign="center",
                numalign="center"
                ))

                input("Aperte enter para continuar: ")

           case 4:
                erros_regressiva = []
                erros_central = []
                erros_progressiva = []

                for i in range(0,len(valores_x)):
                erros_regressiva.append(abs(derivadas_regressivas[i]
- derivadas_obtidas_regressao[i]) if derivadas_regressivas[i] is not
None else None)
                erros_central.append(abs(derivadas_centrais[i] -
derivadas_obtidas_regressao[i]) if derivadas_centrais[i] is not None
else None)

erros_progressiva.append(abs(derivadas_progressivas[i] -
derivadas_obtidas_regressao[i]) if derivadas_progressivas[i] is not
None else None)
```

```
                gerar_graficos_erro(valores_x, erros_regressiva,
erros_central, erros_progressiva)

                print("Gráfico gerado com sucesso!")
                input("Aperte enter para continuar: ")

            case 5:
                rodando = False
```

A função main orquestra todo o fluxo do programa de diferenciação numérica de forma interativa. Primeiro, inicializa os dados de entrada (valores_x, valores_y) e calcula as derivadas pelos três métodos (regressiva, central e progressiva), bem como as derivadas de referência obtidas por regressão. Em seguida, entra num laço while que se mantém ativo até o usuário escolher sair.

Em cada iteração do loop, o console é limpo e são apresentadas ao usuário:

1. Uma tabela com os pontos analisados (usando tabulate).
2. Um menu de opções que permite visualizar os resultados das derivadas, gerar os gráficos de comparação, exibir os erros de aproximação em tabela ou gerar os gráficos de erro.

Dependendo da escolha (case), a função chama as rotinas correspondentes — impressão de tabelas ou geração de gráficos — e aguarda o usuário pressionar Enter para retornar ao menu. Quando a opção “5 – Sair” é selecionada, a variável rodando é definida como False e o loop é encerrado, finalizando o programa.

Resultados Obtidos:

Nesta seção apresentam-se os valores de derivadas numéricas e seus respectivos erros em relação à referência obtida por regressão quadrática, bem como os gráficos que ilustram o comportamento dos métodos regressivo, progressivo e central.

Para estimar os erros relativos aos métodos de diferenciação numérica, utilizou-se um ajuste por regressão polinomial aos pontos experimentais fornecidos. A partir do GeoGebra, obteve-se o polinômio de segundo grau:

$$f(x) = -0.025x^2 + 76.3073788546256x - 0.1354625559661$$

que serviu como referência contínua para o cálculo das derivadas analíticas. Em seguida, calculou-se o valor exato de $f'(x)$ a partir de:

$$f'(x) = -0.050x + 76.3073788546256$$

e comparou-se com as estimativas numéricas (regressiva, central e progressiva), definindo o erro absoluto em cada ponto como:

$$|f'(x_i) - derivada\_numérica(x_i)|$$

Dessa forma, foi possível quantificar e analisar o desempenho de cada método em relação à solução exata fornecida pelo polinômio de regressão.

1. Tabela de Derivadas Numéricas

Tempo(s)	Regressiva (m/s)	Central (m/s)	Progressiva (m/s)	Função Regressão (m/s)
0	N/A	N/A	75	76,307379
3	75	76,6	79	76,157379
5	79	79,6	80	76,057379
8	80	71,8	59,5	75,907379
10	59,5	74	83,666667	75,807379
13	83,666667	N/A	N/A	75.657379

Legenda:

- N/A: Não é possível aplicar o método nesse ponto.
- Valores mostrados com 6 casas decimais

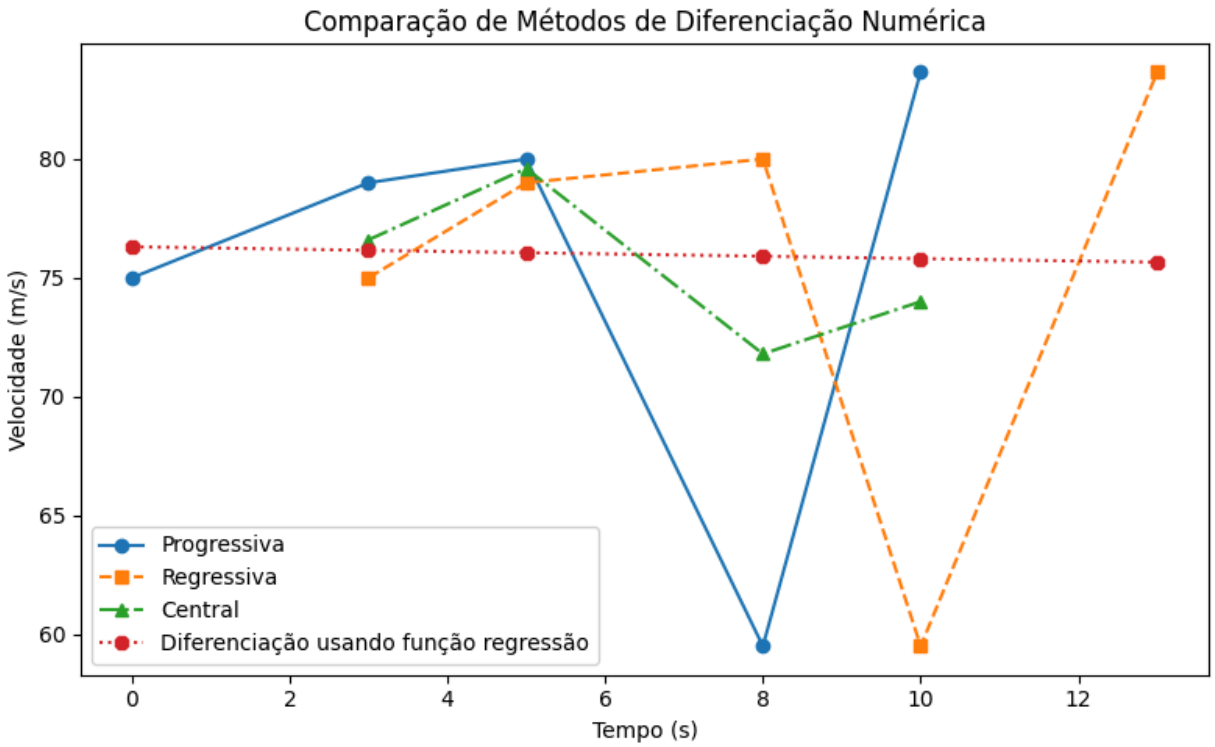
2. Tabela de Erros de Aproximação

Tempo (s)	Erro Regressiva	Erro Central	Erro Progressiva
0	N/A	N/A	1,307379
3	1,157379	0,442621	2,842621
5	2,942621	3,542621	3,942621
8	4,092621	4,107379	16,407379
10	16,307379	1,807379	7,859288
13	8,009288	N/A	N/A

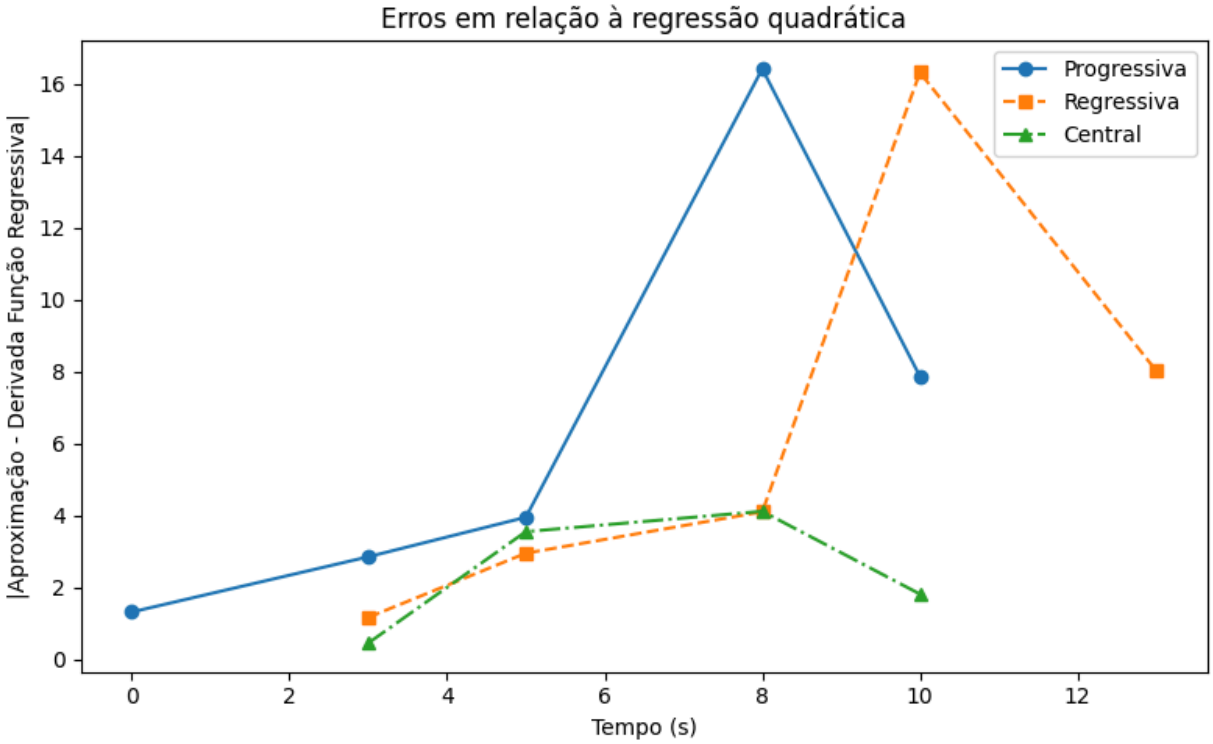
Legenda:

- N/A: Não é possível aplicar o método nesse ponto.
- Valores mostrados com 6 casas decimais

3. Gráfico de Comparação dos métodos



4. Gráfico de Erros em Relação à Regressão Quadrática



CONCLUSÃO

O estudo evidenciou que os métodos de diferenciação numérica apresentam comportamentos distintos em precisão e aplicabilidade, conforme previsto pela teoria. O método das diferenças centrais destacou-se pela maior acurácia, com erros significativamente menores que os métodos unidirecionais, corroborando sua ordem de erro quadrática $O(h^2)$ No entanto, sua restrição a pontos internos do domínio exigiu a combinação com abordagens regressivas ou progressivas nas extremidades, onde estas, embora aplicáveis, introduziram erros sistemáticos mais elevados, da ordem $O(h)$. A análise comparativa revelou ainda discrepâncias entre os erros teóricos e os

observados, sugerindo influência de fatores externos, como possíveis ruídos nos dados ou limitações inerentes ao modelo de regressão adotado como referência. Em regiões de alta curvatura, como em $t = 8s$, a degradação na precisão dos métodos unidirecionais reforçou a importância da escolha criteriosa de técnicas adequadas à variação local da função. Os resultados reforçam que, em aplicações práticas, a seleção do método deve equilibrar densidade de dados, complexidade do fenômeno analisado e custo computacional, priorizando estratégias híbridas que combinem a precisão do método central com a flexibilidade dos métodos de extremidade quando necessário. Assim, o trabalho não apenas validou os fundamentos teóricos, mas também destacou desafios críticos na derivação numérica de dados experimentais, sublinhando a necessidade de análise contextualizada para garantir resultados confiáveis.