Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 12, n. 1, 2026.

Trabalho apresentado no XLIV CNMAC, Fundação Getulio Vargas - Rio de Janeiro - RJ, 2025.

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Criação de uma Apostila sobre Introdução aos Métodos Numéricos e Programação para o Ensino Médio

Luis A. D'Afonseca[1] Edson B. L. Santos[2] Carlos M. M. Cosme[3]
CEFET-MG, Belo Horizonte, MG

Resumo. Este trabalho descreve a criação de uma apostila voltada para professores do Ensino Médio, com o intuito de apresentar conceitos de Pensamento Computacional, a linguagem de programação Python e a aplicação de métodos numéricos para estudantes dessa etapa educacional. A apostila foi desenvolvida como o produto educacional de uma dissertação defendida no âmbito do PROFMAT no CEFET-MG. O material aborda métodos numéricos clássicos, como técnicas para encontrar zeros de funções de uma variável e soluções para sistemas de equações lineares e não lineares. O principal objetivo desta iniciativa é contribuir para a melhoria da qualidade do processo de ensino-aprendizagem na educação básica, oferecendo um recurso didático que pode ser utilizado como apoio para promover discussões sobre Pensamento Computacional e métodos numéricos iterativos, integrando ferramentas computacionais no contexto da sala de aula.

Palavras-chave. Ensino, Pensamento Computacional, Métodos Numéricos, Programação

1 Introdução

Atualmente, a tecnologia está profundamente integrada à nossa realidade, influenciando tanto o modo como vivemos quanto o processo de aprendizagem. Nesse cenário, o Pensamento Computacional surge como uma competência essencial, cuja inserção no ambiente escolar pode preparar as novas gerações para um futuro cada vez mais moldado por avanços tecnológicos. A Base Nacional Comum Curricular (BNCC) [2] reconhece a relevância dessa habilidade ao enfatizar sua importância no contexto educacional, destacando seu papel fundamental na formação dos estudantes e no avanço da aprendizagem no mundo contemporâneo.

Com o objetivo de desenvolver o Pensamento Computacional e, simultaneamente, explorar aspectos aplicados da Matemática, propomos a elaboração de uma apostila que introduz, de maneira simplificada, métodos numéricos em um nível acessível para alunos do Ensino Médio, especialmente aqueles com maior interesse pela disciplina.

Na Seção 2, discutimos o conceito e a relevância do Pensamento Computacional no contexto educacional e na Seção 3 apresentamos como a BNCC sugere que ele seja trabalhado. A Seção 4 é dedicada à apresentação da apostila desenvolvida, detalhando sua estrutura e conteúdo. Por fim, na Seção 5, são apresentadas as considerações finais.

2 Pensamento Computacional no Ensino

A introdução do Pensamento Computacional no âmbito educacional não é uma proposta recente. Desde a década de 1980, Papert [6] já defendia a ideia de que:

[1]luis.dafonseca@cefetmg.br
[2]profedsonbertosa@gmail.com
[3]cmagnomc@cefetmg.br

> Aprender a se comunicar com um computador pode mudar a forma como as pessoas pensam, favorecendo um outro tipo de aprendizado (PAPERT, 1980, p. 6).

Em 2013, Grover e Pea [5] argumentam que o Pensamento Computacional deve integrar o conjunto de conhecimentos que buscamos desenvolver nas crianças por meio da resolução de problemas cotidianos. Os autores afirmam que, assim como reconhecemos a importância da alfabetização científica para a compreensão do mundo, é essencial incluir o entendimento do funcionamento interno de aplicativos e computadores.

Mais recentemente, em 2006, Wing [9] define o Pensamento Computacional como um conjunto de habilidades e competências associadas à Ciência da Computação, que devem ser cultivadas desde os primeiros anos escolares. Wing enfatiza que o Pensamento Computacional é uma forma de pensar inerente aos seres humanos, envolvendo diversas habilidades que exigem diferentes níveis de abstração. Essa forma de pensar difere do funcionamento das máquinas, e, portanto, não se restringe apenas a profissionais da área de computação. Trata-se de uma habilidade intelectual fundamental para todos, assim como ler, escrever ou realizar operações matemáticas.

Outra perspectiva sobre o tema é apresentada por Bordini (2016) [1]. Segundo o autor, o Pensamento Computacional engloba um conjunto de habilidades específicas, tais como:

1. Formular problemas de maneira que possam ser resolvidos com o auxílio de computadores e outras ferramentas;
2. Organizar e analisar dados de forma lógica;
3. Representar dados por meio de abstrações, modelos ou simulações;
4. Automatizar soluções através de algoritmos;
5. Identificar, analisar e implementar soluções de maneira eficiente e eficaz;
6. Generalizar e transferir a abordagem de resolução para outros problemas.

3 Pensamento Computacional na BNCC

Na BNCC, diversos trechos destacam a importância do Pensamento Computacional e da utilização de tecnologias digitais no contexto do ensino de Matemática. Um exemplo disso é a menção de que os conhecimentos matemáticos desempenham um papel fundamental no desenvolvimento do Pensamento Computacional. Além disso, a BNCC enfatiza que os algoritmos podem ser integrados como parte do conteúdo a ser trabalhado no ensino de Matemática, reforçando a relação entre essas duas áreas [2].

> (...) a aprendizagem de Álgebra, como também aquelas relacionadas a Números, Geometria e Probabilidade e estatística, podem contribuir para o desenvolvimento do Pensamento Computacional dos alunos, tendo em vista que eles precisam ser capazes de traduzir uma situação dada em outras linguagens, como transformar situações-problema, apresentadas em língua materna, em fórmulas, tabelas e gráficos e vice-versa. (BNCC, 2018, p. 271)

O trecho a seguir, destaca as conexões entre a Álgebra e o Pensamento Computacional, destacando como essas duas áreas se interligam e se complementam no processo de aprendizagem [2].

> Associado ao Pensamento Computacional, cumpre salientar a importância dos algoritmos e de seus fluxogramas, que podem ser objetos de estudo nas aulas de Matemática. (...) Outra habilidade relativa à álgebra que mantém estreita relação com o Pensamento Computacional é a identificação de padrões para se estabelecer generalizações, propriedades e algoritmos. (BNCC, 2018, p. 271)

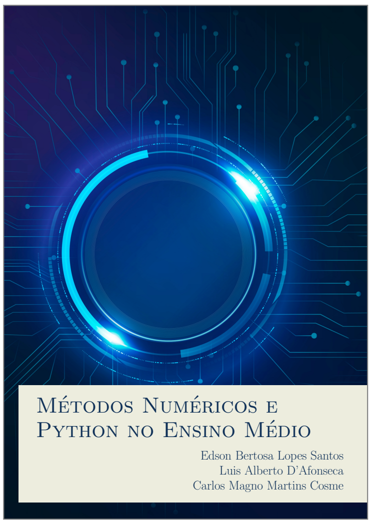
O próximo trecho ressalta a importância de introduzir o uso de tecnologias já nos anos iniciais da educação, como uma estratégia para preparar os estudantes para as demandas e desafios que encontrarão nos anos finais [2].

> Além disso, a BNCC propõe que os estudantes utilizem tecnologias, como calculadoras e planilhas eletrônicas, desde os anos iniciais do Ensino Fundamental. Tal valorização possibilita que, ao chegarem aos anos finais, eles possam ser estimulados a desenvolver o Pensamento Computacional, por meio da interpretação e da elaboração de algoritmos, incluindo aqueles que podem ser representados por fluxogramas. (BNCC, 2018, p. 528)

Apesar de o Pensamento Computacional ser explicitamente mencionado na BNCC, ele ainda é um tema relativamente novo, o que gera diversas incertezas em relação à sua compreensão e implementação nos currículos escolares. Diante desse cenário, a apostila apresentada neste trabalho foi desenvolvida com o objetivo de oferecer um material acessível e prático, que contribua para a aplicação das diretrizes propostas pela BNCC, facilitando a integração do Pensamento Computacional no ambiente educacional. Nossa abordagem consistiu em utilizar os métodos numéricos como um exemplo concreto da aplicação do Pensamento Computacional na resolução de problemas matemáticos. Dessa forma, buscamos ilustrar como problemas complexos podem ser solucionados com o auxílio de algoritmos e o uso de ferramentas computacionais.

4 A Apostila

Elaboramos a apostila "Métodos Numéricos e Python no Ensino Médio" com o objetivo de auxiliar alunos e professores do Ensino Médio na introdução ao Pensamento Computacional e no uso de métodos numéricos aplicados à resolução de problemas matemáticos, utilizando a linguagem de programação Python. Nosso propósito foi criar um material didático e acessível, que não apenas apresentasse conceitos fundamentais, mas também oferecesse uma experiência prática por meio da construção de algoritmos e programas. A apostila está disponível sob uma licença *Creative Commons* na página pessoal do autor deste trabalho [3]. Sua capa, sumário e QR code para download podem ser vistos na Figura 1.



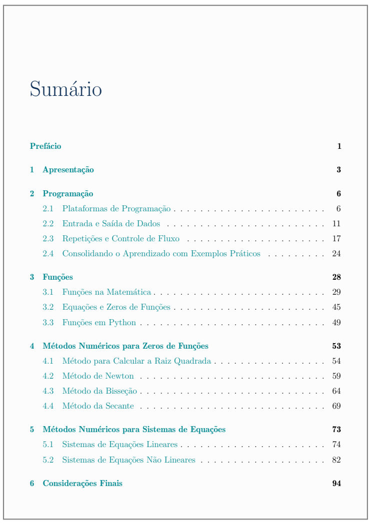
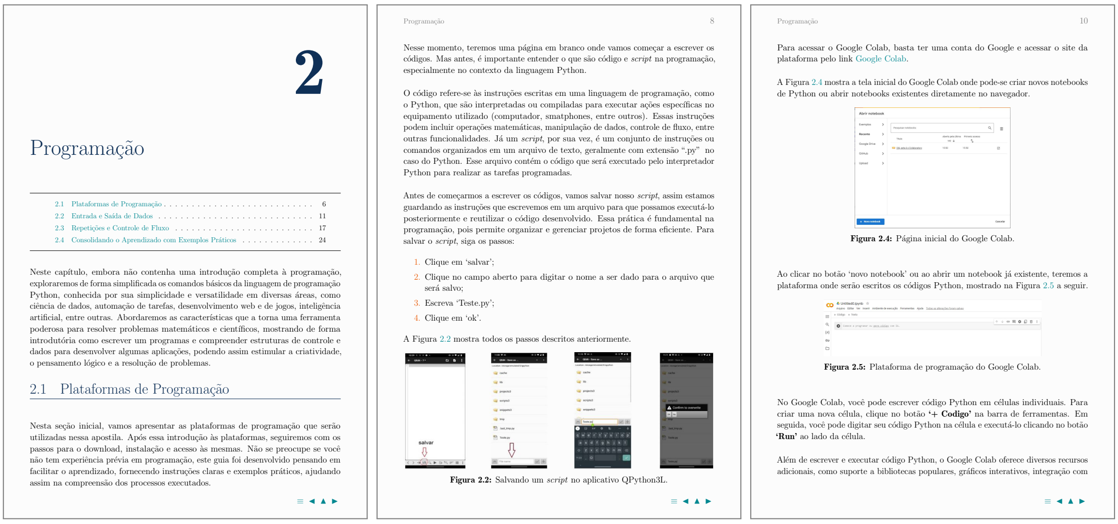

Sumário

Figura 1: Capa, sumário e QR code para baixar a apostila. Fonte: dissertação de E. B. L. Santos [8].

Neste trabalho, propomos apresentar as ideias fundamentais para a elaboração da Apostila “Métodos Numéricos e Python no Ensino Médio”. Não caberia nesta exposição, adentrarmos nos detalhes dos métodos matemáticos discutidos, pois são assuntos muito bem estabelecidos na literatura. A apostila está organizada em cinco capítulos, cada um dedicado a aspectos específicos do aprendizado de programação e métodos numéricos, sempre com o suporte de recursos computacionais de fácil acesso. Por meio de *links* disponíveis na apostila, os códigos Python dos exemplos apresentados podem ser acessados diretamente no Google Colaboratory [4], assim como os códigos utilizados para a geração das imagens.

O primeiro capítulo, “Apresentação”, introduz o propósito da apostila e a metodologia adotada, destacando a importância de integrar conceitos matemáticos com ferramentas de programação para potencializar o aprendizado.

O Capítulo 2, “Programação”, é dedicado a uma breve introdução à linguagem de programação Python. Ele apresenta as plataformas de programação que serão utilizadas ao longo da apostila, com destaque para o Google Colaboratory, uma ferramenta online que permite a execução de códigos Python diretamente no navegador, e o aplicativo QPython3L [7], que pode ser usado em smartphones sem necessidade de conexão à internet, permitindo que os alunos programem em seus dispositivos móveis. Em seguida, são apresentados os comandos básicos de entrada e saída de dados, fundamentais para a interação com o usuário e a manipulação de informações, assim como as estruturas de repetição e controle de fluxo, essenciais para a criação de algoritmos. Para consolidar o aprendizado, é proposta a criação de uma Tabuada Eletrônica Interativa e de um programa que simula uma Venda Online, atividades que permitem aplicar os conceitos estudados e desenvolver a habilidade de construir programas úteis e funcionais. A Figura 2 mostra algumas páginas da apostila que apresentam os ambientes de desenvolvimento utilizados. O objetivo das imagens é ilustrar como a apostila está organizada e não seus conteúdos.


2

Programação

2.1 Plataformas de Programação . . . 6
2.2 Entrada e Saída de Dados . . . 11
2.3 Repetições e Controle de Fluxo . . . 17
2.4 Consolidando o Aprendizado com Exemplos Práticos . . . 24

Neste capítulo, embora não contenha uma introdução completa à programação, exploraremos de forma simplificada os comandos básicos da linguagem de programação Python, conhecida por sua simplicidade e versatilidade em diversas áreas, como ciência de dados, automação de tarefas, desenvolvimento web e de jogos, inteligência artificial, entre outras. Abordaremos as características que a torna uma ferramenta poderosa para resolver problemas matemáticos e científicos, mostrando de forma introdutória como escrever um programas e compreender estruturas de controle e dados para desenvolver algumas aplicações, podendo assim estimular a criatividade, o pensamento lógico e a resolução de problemas.

2.1 Plataformas de Programação

Nesta seção inicial, vamos apresentar as plataformas de programação que serão utilizadas nessa apostila. Após essa introdução às plataformas, seguiremos com os passos para o download, instalação e acesso às mesmas. Não se preocupe se você não tem experiência prévia em programação, este guia foi desenvolvido pensando em facilitar o aprendizado, fornecendo instruções claras e exemplos práticos, ajudando assim na compreensão dos processos executados.

Programação 8

Nesse momento, teremos uma página em branco onde vamos começar a escrever os códigos. Mas antes, é importante entender o que são código e *script* na programação, especialmente no contexto da linguagem Python.

O código refere-se às instruções escritas em uma linguagem de programação, como o Python, que são interpretadas ou compiladas para executar ações específicas no equipamento utilizado (computador, smatphones, entre outros). Essas instruções podem incluir operações matemáticas, manipulação de dados, controle de fluxo, entre outras funcionalidades. Já um *script*, por sua vez, é um conjunto de instruções ou comandos organizados em um arquivo de texto, geralmente com extensão “.py” no caso do Python. Esse arquivo contém o código que será executado pelo interpretador Python para realizar as tarefas programadas.

Antes de começarmos a escrever os códigos, vamos salvar nosso *script*, assim estamos guardando as instruções que escrevemos em um arquivo para que possamos executá-lo posteriormente e reutilizar o código desenvolvido. Essa prática é fundamental na programação, pois permite organizar e gerenciar projetos de forma eficiente. Para salvar o *script*, siga os passos:

1. Clique em ‘salvar’;
2. Clique no campo aberto para digitar o nome a ser dado para o arquivo que será salvo;
3. Escreva ‘Teste.py’;
4. Clique em ‘ok’.

A Figura 2.2 mostra todos os passos descritos anteriormente.

salvar

Figura 2.2: Salvando um *script* no aplicativo QPython3L.

Programação 10

Para acessar o Google Colab, basta ter uma conta do Google e acessar o site da plataforma pelo link Google Colab.

A Figura 2.4 mostra a tela inicial do Google Colab onde pode-se criar novos notebooks de Python ou abrir notebooks existentes diretamente no navegador.

Figura 2.4: Página inicial do Google Colab.

Ao clicar no botão ‘novo notebook’ ou ao abrir um notebook já existente, teremos a plataforma onde serão escritos os códigos Python, mostrado na Figura 2.5 a seguir.

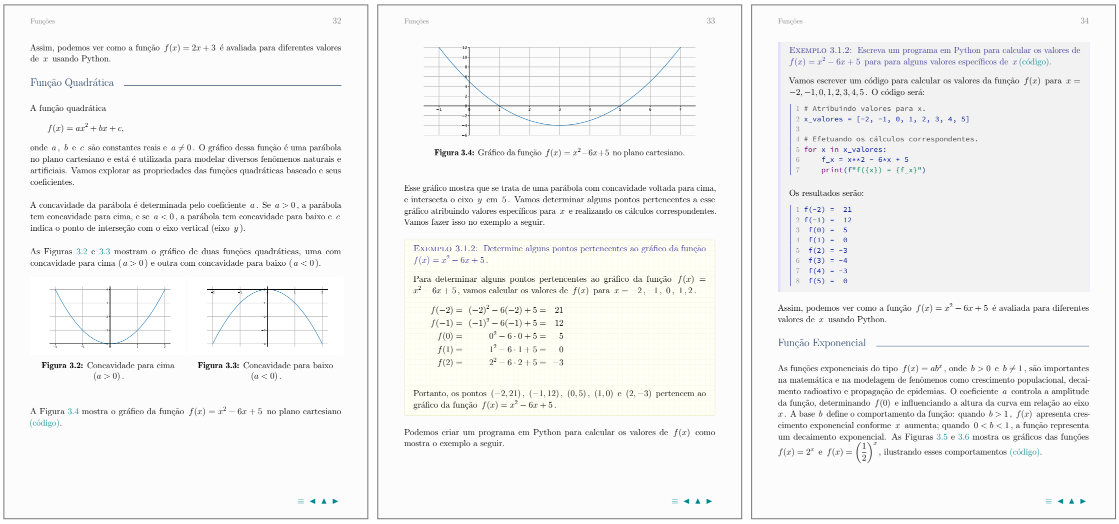
Figura 2.5: Plataforma de programação do Google Colab.

No Google Colab, você pode escrever código Python em células individuais. Para criar uma nova célula, clique no botão **‘+ Codigo’** na barra de ferramentas. Em seguida, você pode digitar seu código Python na célula e executá-lo clicando no botão **‘Run’** ao lado da célula.

Além de escrever e executar código Python, o Google Colab oferece diversos recursos adicionais, como suporte a bibliotecas populares, gráficos interativos, integração com

Figura 2: Páginas da apostila que apresentam os ambientes de desenvolvimento para o Python. Fonte: dissertação de E. B. L. Santos [8].

O Capítulo 3, “Funções”, concentra-se no estudo de funções, tanto no contexto matemático quanto em Python. Primeiramente, revisamos conceitos matemáticos essenciais relacionados a funções e zeros de funções. Em seguida, o capítulo avança para a criação e utilização de funções em Python, demonstrando como essa prática permite modularizar e organizar códigos de maneira eficiente. Este capítulo é fundamental para compreender a importância das funções na progra-



Funções 32

Assim, podemos ver como a função $f(x) = 2x + 3$ é avaliada para diferentes valores de x usando Python.

Função Quadrática

A função quadrática

$$f(x) = ax^2 + bx + c,$$

onde a, b e c são constantes reais e $a \neq 0$. O gráfico dessa função é uma parábola no plano cartesiano e está é utilizada para modelar diversos fenômenos naturais e artificiais. Vamos explorar as propriedades das funções quadráticas baseado e seus coeficientes.

A concavidade da parábola é determinada pelo coeficiente a. Se $a > 0$, a parábola tem concavidade para cima, e se $a < 0$, a parábola tem concavidade para baixo e c indica o ponto de interseção com o eixo vertical (eixo y).

As Figuras 3.2 e 3.3 mostram o gráfico de duas funções quadráticas, uma com concavidade para cima ($a > 0$) e outra com concavidade para baixo ($a < 0$).

Figura 3.2: Concavidade para cima ($a > 0$).

Figura 3.3: Concavidade para baixo ($a < 0$).

A Figura 3.4 mostra o gráfico da função $f(x) = x^2 - 6x + 5$ no plano cartesiano (código).

Funções 33

Figura 3.4: Gráfico da função $f(x) = x^2 - 6x + 5$ no plano cartesiano.

Esse gráfico mostra que se trata de uma parábola com concavidade voltada para cima, e intersecta o eixo y em 5. Vamos determinar alguns pontos pertencentes a esse gráfico atribuindo valores específicos para x e realizando os cálculos correspondentes. Vamos fazer isso no exemplo a seguir.

Exemplo 3.1.2: Determine alguns pontos pertencentes ao gráfico da função $f(x) = x^2 - 6x + 5$.

Para determinar alguns pontos pertencentes ao gráfico da função $f(x) = x^2 - 6x + 5$, vamos calcular os valores de $f(x)$ para $x = -2, -1, 0, 1, 2$.

$$\begin{aligned} f(-2) &= (-2)^2 - 6(-2) + 5 = 21 \\ f(-1) &= (-1)^2 - 6(-1) + 5 = 12 \\ f(0) &= 0^2 - 6 \cdot 0 + 5 = 5 \\ f(1) &= 1^2 - 6 \cdot 1 + 5 = 0 \\ f(2) &= 2^2 - 6 \cdot 2 + 5 = -3 \end{aligned}$$

Portanto, os pontos $(-2, 21)$, $(-1, 12)$, $(0, 5)$, $(1, 0)$ e $(2, -3)$ pertencem ao gráfico da função $f(x) = x^2 - 6x + 5$.

Podemos criar um programa em Python para calcular os valores de $f(x)$ como mostra o exemplo a seguir.

Funções 34

Exemplo 3.1.2: Escreva um programa em Python para calcular os valores de $f(x) = x^2 - 6x + 5$ para para alguns valores específicos de x (código).

Vamos escrever um código para calcular os valores da função $f(x)$ para $x = -2, -1, 0, 1, 2, 3, 4, 5$. O código será:

```
# Atribuindo valores para x.
x_valores = [-2, -1, 0, 1, 2, 3, 4, 5]

# Efetuando os cálculos correspondentes.
for x in x_valores:
    f_x = x**2 - 6*x + 5
    print(f"f({x}) = {f_x}")
```

Os resultados serão:

```
f(-2) =  21
f(-1) =  12
 f(0) =  5
 f(1) =  0
 f(2) = -3
 f(3) = -4
 f(4) = -3
 f(5) =  0
```

Assim, podemos ver como a função $f(x) = x^2 - 6x + 5$ é avaliada para diferentes valores de x usando Python.

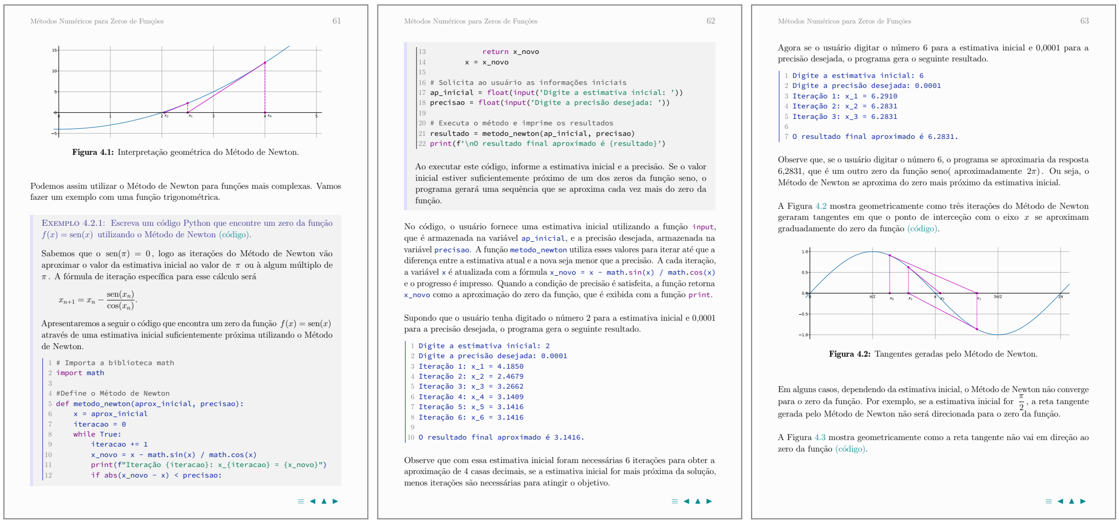
Função Exponencial

As funções exponenciais do tipo $f(x) = ab^x$, onde $b > 0$ e $b \neq 1$, são importantes na matemática e na modelagem de fenômenos como crescimento populacional, decaimento radioativo e propagação de epidemias. O coeficiente a controla a amplitude da função, determinando $f(0)$ e influenciando a altura da curva em relação ao eixo x. A base b define o comportamento da função: quando $b > 1$, $f(x)$ apresenta crescimento exponencial conforme x aumenta; quando $0 < b < 1$, a função representa um decaimento exponencial. As Figuras 3.5 e 3.6 mostra os gráficos das funções $f(x) = 2^x$ e $f(x) = \left(\frac{1}{2}\right)^x$, ilustrando esses comportamentos (código).

Figura 3: Páginas da apostila que apresentam a função polinomial de segundo grau. Fonte: dissertação de E. B. L. Santos [8].

mação e sua aplicação na resolução de problemas matemáticos. Na Figura 3 está o trecho que apresenta a função polinomial de segundo grau, seu gráfico e atividades para serem realizadas tanto manualmente quanto computacionalmente.

O Capítulo 4, "Métodos Numéricos para Zeros de Funções", inicia apresentando um método numérico para calcular raízes quadradas. Em seguida, aborda métodos clássicos utilizados para calcular zeros de funções, como o Método de Newton, o Método da Bisseção e o Método da Secante. Cada método é explicado em detalhes, seguido de exemplos práticos implementados em Python. O uso desses métodos permite resolver equações complexas de maneira eficiente e precisa. Além



Métodos Numéricos para Zeros de Funções 61

Figura 4.1: Interpretação geométrica do Método de Newton.

Podemos assim utilizar o Método de Newton para funções mais complexas. Vamos fazer um exemplo com uma função trigonométrica.

Exemplo 4.2.1: Escreva um código Python que encontre um zero da função $f(x) = \text{sen}(x)$ utilizando o Método de Newton (código).

Sabemos que o $\text{sen}(\pi) = 0$, logo as iterações do Método de Newton vão aproximar o valor da estimativa inicial ao valor de π ou à algum múltiplo de π. A fórmula de iteração específica para esse cálculo será

$$x_{n+1} = x_n - \frac{\text{sen}(x_n)}{\cos(x_n)}.$$

Apresentaremos a seguir o código que encontra um zero da função $f(x) = \text{sen}(x)$ através de uma estimativa inicial suficientemente próxima utilizando o Método de Newton.

```
# Importa a biblioteca math
import math

#Define o Método de Newton
def metodo_newton(aprox_inicial, precisao):
    x = aprox_inicial
    iteracao = 0
    while True:
        iteracao += 1
        x_novo = x - math.sin(x) / math.cos(x)
        print(f"Iteração {iteracao}: x_{iteracao} = {x_novo}")
        if abs(x_novo - x) < precisao:
```

Métodos Numéricos para Zeros de Funções 62

```
            return x_novo
        x = x_novo

# Solicita ao usuário as informações iniciais
ap_inicial = float(input('Digite a estimativa inicial: '))
precisao = float(input('Digite a precisão desejada: '))

# Executa o método e imprime os resultados
resultado = metodo_newton(ap_inicial, precisao)
print(f'\nO resultado final aproximado é {resultado}')
```

Ao executar este código, informe a estimativa inicial e a precisão. Se o valor inicial estiver suficientemente próximo de um dos zeros da função seno, o programa gerará uma sequência que se aproxima cada vez mais do zero da função.

No código, o usuário fornece uma estimativa inicial utilizando a função `input`, que é armazenada na variável `ap_inicial`, e a precisão desejada, armazenada na variável `precisao`. A função `metodo_newton` utiliza esses valores para iterar até que a diferença entre a estimativa atual e a nova seja menor que a precisão. A cada iteração, a variável `x` é atualizada com a fórmula `x_novo = x - math.sin(x) / math.cos(x)` e o progresso é impresso. Quando a condição de precisão é satisfeita, a função retorna `x_novo` como a aproximação do zero da função, que é exibida com a função `print`.

Supondo que o usuário tenha digitado o número 2 para a estimativa inicial e 0,0001 para a precisão desejada, o programa gera o seguinte resultado.

```
Digite a estimativa inicial: 2
Digite a precisão desejada: 0.0001
Iteração 1: x_1 = 4.1850
Iteração 2: x_2 = 2.4679
Iteração 3: x_3 = 3.2662
Iteração 4: x_4 = 3.1409
Iteração 5: x_5 = 3.1416
Iteração 6: x_6 = 3.1416

O resultado final aproximado é 3.1416.
```

Observe que com essa estimativa inicial foram necessárias 6 iterações para obter a aproximação de 4 casas decimais, se a estimativa inicial for mais próxima da solução, menos iterações são necessárias para atingir o objetivo.

Métodos Numéricos para Zeros de Funções 63

Agora se o usuário digitar o número 6 para a estimativa inicial e 0,0001 para a precisão desejada, o programa gera o seguinte resultado.

```
Digite a estimativa inicial: 6
Digite a precisão desejada: 0.0001
Iteração 1: x_1 = 6.2910
Iteração 2: x_2 = 6.2831
Iteração 3: x_3 = 6.2831

O resultado final aproximado é 6.2831.
```

Observe que, se o usuário digitar o número 6, o programa se aproximaria da resposta 6,2831, que é um outro zero da função seno(aproximadamente 2π). Ou seja, o Método de Newton se aproxima do zero mais próximo da estimativa inicial.

A Figura 4.2 mostra geometricamente como três iterações do Método de Newton geraram tangentes em que o ponto de interceção com o eixo x se aproximam graduadamente do zero da função (código).

Figura 4.2: Tangentes geradas pelo Método de Newton.

Em alguns casos, dependendo da estimativa inicial, o Método de Newton não converge para o zero da função. Por exemplo, se a estimativa inicial for $\frac{\pi}{2}$, a reta tangente gerada pelo Método de Newton não será direcionada para o zero da função.

A Figura 4.3 mostra geometricamente como a reta tangente não vai em direção ao zero da função (código).

Figura 4: Páginas da apostila que apresentam o método de Newton. Fonte: dissertação de E. B. L. Santos [8].

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics. v. 12, n. 1, 2026.

disso, os códigos podem ser acessados diretamente no Google Colaboratory, facilitando o acesso dos leitores aos códigos e permitindo a execução e experimentação dos exemplos apresentados. O trecho que apresenta o Método de Newton pode ser visto na Figura 4.

O Capítulo 5, "Métodos Numéricos para Sistemas de Equações", foca na resolução de sistemas de equações, tanto lineares quanto não lineares. Ele inicia discutindo o Método de Jacobi para resolver sistemas de equações lineares, detalhando seu funcionamento e aplicabilidade. Em seguida, aborda o Método de Newton para sistemas de equações não lineares, mostrando como essa técnica pode ser adaptada para resolver sistemas mais complexos. A Figura 5 mostra um exemplo que encontra os pontos de intersecção de duas curvas utilizando o método de Newton.

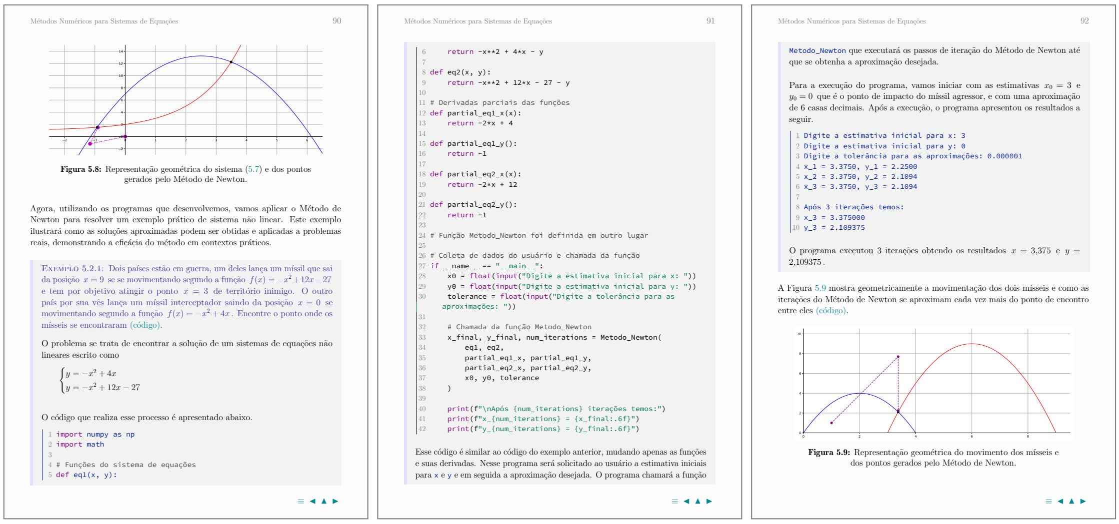

Métodos Numéricos para Sistemas de Equações 90

Figura 5.8: Representação geométrica do sistema (5.7) e dos pontos gerados pelo Método de Newton.

Agora, utilizando os programas que desenvolvemos, vamos aplicar o Método de Newton para resolver um exemplo prático de sistema não linear. Este exemplo ilustrará como as soluções aproximadas podem ser obtidas e aplicadas a problemas reais, demonstrando a eficácia do método em contextos práticos.

Exemplo 5.2.1: Dois países estão em guerra, um deles lança um míssil que sai da posição $x = 9$ se se movimentando segundo a função $f(x) = -x^2 + 12x - 27$ e tem por objetivo atingir o ponto $x = 3$ de território inimigo. O outro país por sua vês lança um míssil interceptador saindo da posição $x = 0$ se movimentando segundo a função $f(x) = -x^2 + 4x$. Encontre o ponto onde os mísseis se encontraram (código).

O problema se trata de encontrar a solução de um sistemas de equações não lineares escrito como

$$\begin{cases} y = -x^2 + 4x \\ y = -x^2 + 12x - 27 \end{cases}$$

O código que realiza esse processo é apresentado abaixo.

```
import numpy as np
import math

# Funções do sistema de equações
def eq1(x, y):
```

Métodos Numéricos para Sistemas de Equações 91

```
    return -x**2 + 4*x - y

def eq2(x, y):
    return -x**2 + 12*x - 27 - y

# Derivadas parciais das funções
def partial_eq1_x(x):
    return -2*x + 4

def partial_eq1_y():
    return -1

def partial_eq2_x(x):
    return -2*x + 12

def partial_eq2_y():
    return -1

# Função Metodo_Newton foi definida em outro lugar

# Coleta de dados do usuário e chamada da função
if __name__ == "__main__":
    x0 = float(input("Digite a estimativa inicial para x: "))
    y0 = float(input("Digite a estimativa inicial para y: "))
    tolerance = float(input("Digite a tolerância para as
  aproximações: "))

    # Chamada da função Metodo_Newton
    x_final, y_final, num_iterations = Metodo_Newton(
        eq1, eq2,
        partial_eq1_x, partial_eq1_y,
        partial_eq2_x, partial_eq2_y,
        x0, y0, tolerance
    )

    print(f"\nApós {num_iterations} iterações temos:")
    print(f"x_{num_iterations} = {x_final:.6f}")
    print(f"y_{num_iterations} = {y_final:.6f}")
```

Esse código é similar ao código do exemplo anterior, mudando apenas as funções e suas derivadas. Nesse programa será solicitado ao usuário a estimativa iniciais para x e y e em seguida a aproximação desejada. O programa chamará a função

Métodos Numéricos para Sistemas de Equações 92

Metodo_Newton que executará os passos de iteração do Método de Newton até que se obtenha a aproximação desejada.

Para a execução do programa, vamos iniciar com as estimativas $x_0 = 3$ e $y_0 = 0$ que é o ponto de impacto do míssil agressor, e com uma aproximação de 6 casas decimais. Após a execução, o programa apresentou os resultados a seguir.

```
Digite a estimativa inicial para x: 3
Digite a estimativa inicial para y: 0
Digite a tolerância para as aproximações: 0.000001
x_1 = 3.3750, y_1 = 2.2500
x_2 = 3.3750, y_2 = 2.1094
x_3 = 3.3750, y_3 = 2.1094

Após 3 iterações temos:
x_3 = 3.375000
y_3 = 2.109375
```

O programa executou 3 iterações obtendo os resultados $x = 3{,}375$ e $y = 2{,}109375$.

A Figura 5.9 mostra geometricamente a movimentação dos dois mísseis e como as iterações do Método de Newton se aproximam cada vez mais do ponto de encontro entre eles (código).

Figura 5.9: Representação geométrica do movimento dos mísseis e dos pontos gerados pelo Método de Newton.

Figura 5: Páginas da apostila que mostram como resolver um sistema não-linear. Fonte: dissertação de E. B. L. Santos [8].

Os temas abordados nesta apostila estão alinhados com o contexto educacional e tecnológico atual. A programação, por exemplo, tornou-se uma habilidade fundamental, não apenas para quem deseja seguir carreiras na área de tecnologia, mas também como uma ferramenta poderosa para resolver problemas em diversas áreas. O conhecimento sobre métodos numéricos, por sua vez, permite que os alunos compreendam e apliquem técnicas amplamente utilizadas na ciência, engenharia, economia e muitas outras disciplinas, para solucionar problemas complexos que não podem ser resolvidos de forma analítica. Ao capacitar os alunos com essas habilidades, estamos não apenas preparando-os para desafios acadêmicos e profissionais, mas também despertando o interesse e a curiosidade por explorar e inovar em um mundo cada vez mais orientado pela tecnologia.

5 Considerações Finais

A apostila desenvolvida tem como objetivo auxiliar o professor de matemática interessado em explorar conceitos matemáticos mais avançados no ensino do Pensamento Computacional, com ênfase na utilização de algoritmos, para o Ensino Médio. Ela está alinhada com o papel fundamental do Pensamento Computacional, conforme descrito na Base Nacional Comum Curricular (BNCC). Além disso, a apostila contribui para a inserção dos estudantes no universo da tecnologia digital, desmistificando o funcionamento de computadores e calculadoras, ao introduzir a programação e os métodos utilizados para resolver problemas complexos.

DOI: 10.5540/03.2026.012.01.0269

Agradecimentos

Os autores agradecem ao apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Código de Financiamento 001 – e ao CEFET-MG e à FAPEMIG pelo apoio financeiro pela Chamada FAPEMIG 16/2024 – Processo PCE-00114-25.

Referências

[1] A. Bordini. "Computação na educação básica no Brasil: o estado da arte". Em: **Revista de Informática Teórica e Aplicada** 23.2 (2016), pp. 210–238.

[2] Brasil. **Base Nacional Comum Curricular**. Brasília: Ministério da Educação, 2018.

[3] L. A. D'Afonseca. **Página pessoal**. Online. Acessado em 27/04/2025, `https://sites.google.com/view/prof-luis-dafonseca/ensino-b\%C3\%A1sico/Calculo-Numerico`.

[4] Google. **Google Colaboratory**. Online. Acessado em 27/04/2025, `https://colab.research.google.com`.

[5] S. Grover e R. Pea. "Computational Thinking in K–12: A Review of the State of the Field". Em: **Educational Researcher** 42.1 (2013), pp. 38–43.

[6] S. Papert. **Mindstorms: Children, Computers, and Powerful Ideas**. New York, NY: Basic Books, Inc., 1980.

[7] QPythonLab. **QPython3 - Python for Android**. Online. Acessado em 27/04/2025, `https://play.google.com/store/apps/details?id=org.qpython.qpy3&pcampaignid=web_share`.

[8] E. B. L. Santos. "Elaboração de uma apostila sobre programação e métodos numéricos para o ensino médio". PROFMAT. Dissertação de mestrado. CEFET-MG, 2024.

[9] J. M. Wing. "Computational Thinking". Em: **Communications of the ACM** 49.3 (2006), pp. 33–35.

