UNIVERSIDADE FEDERAL DO CARIRI
CENTRO DE CIÊNCIAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA EM REDE NACIONAL

MÁRCIO JOSÉ QUEIROZ CLAUDINO

UM MÉTODO COMPUTACIONAL PARA MINIMIZAR FUNÇÕES QUADRÁTICAS

JUAZEIRO DO NORTE
2024

MÁRCIO JOSÉ QUEIROZ CLAUDINO

UM MÉTODO COMPUTACIONAL PARA MINIMIZAR FUNÇÕES QUADRÁTICAS

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Matemática em Rede Nacional do Centro de Ciências e Tecnologia da Universidade Federal do Cariri, como parte dos requisitos necessários à obtenção do título de Mestre em Matemática.

Orientadores: Valdinês Leite de Sousa Júnior
Erica Boizan Batista

JUAZEIRO DO NORTE
2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Cariri
Sistema de Bibliotecas

C615m Claudino, Márcio José Queiroz.
Um método computacional para minimizar funções quadráticas / Márcio José Queiroz Claudino. - 2024.
45 f. il. color.; 30 cm.
(Inclui bibliografia, p. 34-35).

Dissertação (Mestrado) – Universidade Federal do Cariri, Centro de Ciências e Tecnologia, Programa de Pós-graduação em Matemática em Rede Nacional, Juazeiro do Norte, 2024.

Orientador: Prof. Dr. Valdinês Leite de Sousa Junior.
Co-orientador: Erica Boizan Batista.

1. Métodos Computacionais. 2. Funções Quadráticas. 3. Programação. I. Sousa Júnior, Valdinês Leite de - orientador. II. Batista, Erica Boizan - co- orientador. III, Título.

CDD 510

Bibliotecária: Maria Eliziana Pereira de Sousa – CRB 15/564

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO CARIRI
CENTRO DE CIÊNCIAS ETECNOLOGIA
MESTRADO PROFISSIONAL EM MATEMÁTICA EM REDE NACIONAL - PROFMAT

UM MÉTODO COMPUTACIONAL PARA MINIMIZAR FUNÇÕES QUADRÁTICAS

Márcio José Queiroz Claudino

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Matemática em Rede Nacional – PROFMAT do Centro de Ciências e Tecnologia da Universidade Federal do Cariri, como requisito parcial para obtenção do Título de Mestre em Matemática.

Área de concentração: Matemática.

Aprovada em 14 de junho de 2024.

Banca Examinadora

gov.br Documento assinado digitalmente
VALDINES LEITE DE SOUSA JUNIOR
Data: 30/07/2024 20:10:47-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Valdinês Leite de Sousa Júnior
Orientador - UFCA

gov.br Documento assinado digitalmente
EDVALDO ELIAS DE ALMEIDA BATISTA
Data: 30/07/2024 16:09:12-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Edvaldo Elias de Almeida Batista
Examinador externo - UFOB

gov.br Documento assinado digitalmente
DIEGO FRANKIN DE SOUZA VERAS SANT ANA
Data: 30/07/2024 08:18:30-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Diego Frankin de Souza Veras Sant'Ana
Examinador externo - UFCA

gov.br Documento assinado digitalmente
VICENTE HELANO FEITOSA BATISTA SOBRINHO
Data: 30/07/2024 19:58:23-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Vicente Helano Feitosa Batista Sobrinho
Examinador interno - UFCA

A todos os meus professores, que alimentaram minha paixão pela matemática e me ajudaram a chegar onde estou hoje.

Agradecimentos

Expresso minha profunda gratidão à minha esposa, Alice da Silva Amaral Claudino, pelo apoio incondicional, incentivo constante, paciência incansável e por estar ao meu lado em inúmeras viagens longas ao longo do curso. Agradeço também ao meu professor, Valdinês Leite de Sousa Júnior, e à minha coorientadora, Erica Boizan Batista, pela orientação valiosa e indispensável neste trabalho.

RESUMO

Neste trabalho, exploraremos um método computacional para a minimização de funções quadráticas. Utilizaremos a linguagem de programação Java para a implementação do algoritmo proposto. Uma aplicação do método utilizando o software GeoGebra também será apresentada. Nosso objetivo é introduzir princípios fundamentais de programação no Ensino Básico, ressaltando a relevância da matemática na era digital, oferecendo aos alunos uma introdução prática e tangível ao mundo da programação, preparando-os para desafios futuros e oportunidades na área de matemática aplicada e computação. Este trabalho é fundamentado integralmente em um artigo científico com o mesmo título, escrito por Márcio José Queiroz Claudino, Valdinês Leite de Sousa Júnior e Erica Boizan Batista, o qual foi submetido para publicação na revista do Professor de Matemática Online - PMO. O artigo apresenta um método computacional que integra o ensino de matemática e programação, demonstrando a aplicação prática dos conceitos matemáticos. O método será implementado utilizando a linguagem de programação Java e o software de matemática dinâmica GeoGebra.

Palavras-chave: Métodos Computacionais. Funções Quadráticas. Programação.

ABSTRACT

In this work, we will explore a computational method for minimizing quadratic functions. We will use the Java programming language to implement the proposed algorithm. An application of the method using the GeoGebra software will also be presented. Our goal is to introduce fundamental programming principles in Basic Education, emphasizing the relevance of mathematics in the digital age, offering students a practical and tangible introduction to the world of programming, preparing them for future challenges and opportunities in the field of applied mathematics and computing. This work is fully based on a scientific article with the same title, written by Márcio José Queiroz Claudino, Valdinês Leite de Sousa Júnior, and Erica Boizan Batista, which has been submitted for publication in the journal "Professor de Matemática Online - PMO". The article presents a computational method that integrates the teaching of mathematics and programming, demonstrating the practical application of mathematical concepts. The method will be implemented using the Java programming language and the dynamic mathematics software GeoGebra.

Keywords: Computational Methods. Quadratic Functions. Programming.

Sumário

Lista de Figuras **x**

Lista de Tabelas **xi**

1 Introdução **1**

2 Percurso Metodológico **4**

3 Minimizando funções quadráticas **6**

3.1 O algoritmo 7

4 Resultados de convergência **25**

5 Considerações finais **32**

Referências **34**

Lista de Figuras

3.1 Gráfico para $x_1 = 1$, $\lambda = 3$ e $\gamma = 1$ 22
3.2 Gráfico para $x_1 = -10$, $\lambda = 3$ e $\gamma = 1$ 23

Lista de Tabelas

3.1 Resultados para $x_1 = 1$ 17
3.2 Resultados para $x_1 = -10$ 18
3.3 Resultados da lista l_3 para $x_1 = 1$. 23
3.4 Resultados da lista l_3 para $x_1 = -10$. 24

4.1 Resultados para $x_1 = 1$, $\lambda = 3$ e $\gamma = 1$ 31
4.2 Resultados para $x_1 = 1$, $\lambda = 100$ e $\gamma = 49$. 31

Capítulo 1

Introdução

A inserção da programação matemática no ensino básico é uma mudança paradigmática que reflete a crescente relevância da tecnologia na sociedade contemporânea. O ensino de programação não se trata apenas de preparar os estudantes para carreiras em ciência da computação, mas também de desenvolver habilidades essenciais para o século XXI, como pensamento crítico, resolução de problemas e criatividade.

Em um contexto cada vez mais digitalizado, adquirir conhecimentos fundamentais em programação é essencial, independentemente da área de interesse dos estudantes. Vários países, como Alemanha, Argentina, Austrália, França, Inglaterra, Estados Unidos, e outros, têm reconhecido essa necessidade e estão implementando um currículo mínimo de Ciência da Computação em suas instituições educacionais.

Segundo Vasconcelos e Moraes Neto (2020, p.69), no Brasil, a exposição desse conteúdo tem sido restrita ao ensino médio profissionalizante, graduação e pós-graduação. A instrução em Ciência da Computação tem sido predominantemente oferecida por meio de cursos de Educação Profissional e Tecnológica, com uma ênfase considerável no aspecto técnico. No entanto, a Base Nacional Comum Curricular (BNCC) enfatiza a importância das Tecnologias Digitais de Informação e Comunicação (TDICs) como parte das habilidades essenciais para os alunos.

> Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva. (BRASIL, 2018, p. 9)

Nesse contexto, a programação desempenha um papel essencial, permitindo o desenvolvimento de soluções tecnológicas, promovendo o pensamento crítico e ético, e capacitando os indivíduos a exercerem protagonismo e autoria na sociedade digital atual.

A Sociedade Brasileira de Computação (SBC) também ressalta a importância de incluir a Computação na Educação Básica do Brasil. Em julho de 2017, a SBC lançou o documento "Referenciais de Formação em Computação: Educação Básica" (SBC, 2017), que delineia os fundamentos da área e as competências necessárias. Esse material aborda três conceitos centrais: Pensamento Computacional, Mundo Digital e Cultura Digital. A programação está intimamente ligada a esses conceitos, sendo utilizada para resolver problemas de forma eficiente, compreender o funcionamento dos computadores e permitir a expressão de soluções em diversos contextos culturais e disciplinares.

Além disso, a introdução de conceitos relacionados à computação desde cedo ajuda a democratizar o acesso à tecnologia. Conforme destacado por Alves, "é crucial que as crianças iniciem seu aprendizado em computação já no Ensino Fundamental" (ALVES, 2016, p. 32).

Os autores ressaltam a distinção vital entre a mera alfabetização digital e a proficiência digital no contexto educacional primário. Enquanto a alfabetização digital se concentra em ensinar habilidades básicas de uso de tecnologia, como processamento de texto e e-mails, a proficiência digital capacita os alunos a aprender, adaptar e aplicar novas tecnologias de maneira produtiva. Portanto, ao oferecer aos estudantes a oportunidade de aprender programação desde cedo, estamos diminuindo a disparidade digital e preparando uma geração para enfrentar os desafios de um mundo cada vez mais tecnológico.

Outro aspecto crucial é o estímulo à criatividade. A programação permite que os alunos criem projetos tangíveis, desde jogos simples até aplicativos e websites. Essa capacidade de criar algo novo e funcional é incrivelmente empoderadora e pode inspirar os alunos a explorar áreas como design de interfaces, desenvolvimento de jogos, robótica e muito mais. Segundo Maltempi e Valente:

> A criatividade é necessária e estimulada, uma vez que há sempre inúmeras maneiras de se resolver um problema por meio da programação, ou seja, não existem passos pré-definidos e diferentes soluções (programas) podem ser obtidas – essa característica torna a atividade de programação especialmente interessante para ser empregada em um ambiente de ensino-aprendizagem. (MALTEMPI e VALENTE, 2000, p. 1)

De acordo com Kafai e Burke (2013), a programação também tem o potencial para promover a colaboração e o trabalho em equipe. Muitos projetos de programação são desenvolvidos em equipe, exigindo que os alunos aprendam a comunicar suas ideias, resolver conflitos e contribuir de forma construtiva para alcançar um objetivo comum. Os autores destacam ainda a importância do aprendizado da programação,

mesmo para aqueles que não pretendem se tornar cientistas da computação. Embora apenas alguns se tornem programadores responsáveis por escrever códigos e projetar sistemas que sustentam grande parte de nossas vidas diárias, aprendemos que muitos de nós, em algum momento, precisarão de alguma forma de programação.

Todos nós somos e continuaremos sendo usuários de tecnologias digitais e, portanto, em certos momentos, precisaremos examinar criticamente e construtivamente os designs e as decisões por trás delas. Sendo assim, introduzir conceitos de computação, e em particular o estudo da programação, no ensino básico é uma medida essencial para preparar os alunos para um futuro cada vez mais digital, desenvolvendo habilidades-chave que serão valiosas em qualquer campo profissional e promovendo a inclusão digital e a criatividade.

Este trabalho é baseado em um artigo submetido á Revista do Professor de Matemática Online - PMO e está estruturado da seguinte forma: Na Seção 3, oferecemos uma introdução concisa às funções quadráticas, destacando propriedades fundamentais para este estudo. Apresentaremos também o método proposto, ilustrando seu funcionamento por meio de um exemplo prático implementado na linguagem de programação Java. Na Seção 4, abordaremos os resultados de convergência, essenciais para justificar o comportamento do algoritmo. Embora esses resultados recorram a conceitos matemáticos mais avançados, é importante ressaltar que o entendimento prático do método pode ser aplicado sem a necessidade de compreensão completa das técnicas de demonstração empregadas nesta seção. Adicionalmente, nesta mesma seção, será apresentado um segundo exemplo, enfatizando o desempenho do método. Finalmente, na Seção 5, serão expostas as observações finais, proporcionando uma síntese do trabalho e possíveis direções para futuras investigações.

Capítulo 2

Percurso Metodológico

Segundo Arimoto e Oliveira (2019) o processo de aprendizagem de programação não é uma tarefa simples, pois os conceitos são complexos e a carga de conhecimento necessária não é trivial, especialmente para alunos iniciantes. Esta constatação destaca a necessidade de novas metodologias e estratégias de ensino que possam tornar a aprendizagem da programação mais acessível e envolvente para os alunos.

A exploração de problemas computacionais, conforme delineado por Salvetti e Barbosa (1998), destaca a importância de definir claramente a entrada e as condições para a saída desejada. Essa perspectiva, ao ser integrada ao currículo escolar, possibilita o desenvolvimento de habilidades cruciais, como pensamento crítico, resolução de problemas e lógica computacional, essenciais para o sucesso dos alunos em um mundo cada vez mais tecnológico. Ao incorporar esses princípios no ensino, especialmente ao integrar o ensino da programação com o da matemática, com foco nas funções quadráticas, os alunos são expostos a uma abordagem interdisciplinar que une conceitos matemáticos fundamentais com habilidades práticas de programação.

Nesse contexto, abordar as funções quadráticas no ensino básico oferece vantagens significativas, especialmente quando consideramos sua relação com a programação. Estas funções desempenham um papel fundamental na introdução dos alunos ao pensamento algorítmico e à lógica de programação. Ao explorar as propriedades das funções quadráticas, como identificar raízes, determinar o vértice da parábola e traçar gráficos, os estudantes não só desenvolvem habilidades matemáticas essenciais, mas também adquirem uma base sólida para compreender conceitos de programação.

Além disso, as funções quadráticas encontram amplas aplicações no mundo da programação. Muitos algoritmos e estruturas de dados utilizam equações quadráticas e dependem de uma compreensão profunda de suas propriedades para implementações eficazes. Ao incorporar o estudo das funções quadráticas no contexto da programação, os alunos são capazes de visualizar e aplicar conceitos matemáticos de forma prática e tangível, o que fortalece sua compreensão e interesse na disciplina.

Essa abordagem interdisciplinar não apenas prepara os alunos para enfrentar os desafios futuros em campos tecnológicos e científicos, mas também os capacita a explorar conexões profundas entre o ensino da programação e o estudo das funções quadráticas. Optou-se por estudar um método que minimize essas funções, o que desempenha um papel fundamental na matemática e na otimização.

Os métodos de minimização de funções são essenciais em matemática e ciência de dados para encontrar pontos ótimos em um espaço de solução.

Ao estudar métodos de minimização em contextos matemáticos e de otimização, os aspirantes a programadores desenvolvem habilidades analíticas e de resolução de problemas fundamentais. A capacidade de identificar e resolver problemas de maneira eficiente é uma competência essencial para qualquer programador bem-sucedido. A compreensão da teoria das funções quadráticas e sua aplicação nos métodos de minimização contribui para uma formação sólida, preparando os estudantes para enfrentar os desafios complexos do mundo digital em constante evolução.

Nesse contexto, o algoritmo selecionado para atender ao objetivo deste artigo é uma versão de um método amplamente reconhecido na teoria da otimização: o método do ponto proximal. Especificamente, o algoritmo abordado aqui é uma instância adaptada para funções quadráticas, derivada do Algoritmo do Ponto Proximal apresentado por Iusem (1995). O autor foca na convergênciado algoritmo em cenários mais avançados da área de otimizaçao. Contudo, o presente artigo optou por uma abordagem mais acessível, visando sua aplicação no ensino básico.

A estrutura apresentada nesta versão do método é simplificada, composta apenas por elementos de matemática básica, tornando-a compreensível para alunos e professores do ensino fundamental e médio. Essa abordagem visa permitir que o método seja aplicado em uma variedade de problemas matemáticos sem grandes dificuldades, promovendo uma compreensão mais acessível e prática dos conceitos subjacentes à otimização e à resolução de problemas.

Para a implementação do algoritmo estudado neste trabalho, escolheu-se a linguagem de programação Java. A linguagem Java é uma escolha valiosa para estudantes do ensino básico devido à sua ampla utilidade em várias áreas da computação. Como código aberto, oferece acesso gratuito a ferramentas de programação, facilitando o aprendizado sem custos significativos. Sua portabilidade permite que programas sejam executados em diferentes dispositivos e sistemas operacionais, enriquecendo a experiência prática. Assim, Java proporciona uma plataforma acessível e versátil para aprender programação, preparando os estudantes para futuros estudos e carreiras em tecnologia da informação.

Capítulo 3

Minimizando funções quadráticas

Faremos aqui uma breve apresentação dos conceito de minimização e funções quadráticas. Para uma abordagem mais detalhada veja [8].

Minimizar uma função é um conceito fundamental em diversas áreas da matemática e da ciência, especialmente em otimização. O objetivo principal é encontrar o ponto no domínio da função onde ela assume seu valor mínimo. Esse conceito é amplamente utilizado para resolver problemas em engenharia, economia, ciência de dados, inteligência artificial, entre outros.

Uma função $f : \mathbb{R} \to \mathbb{R}$ é considerada quadrática se pode ser expressa como $f(x) = ax^2 + bx + c$, onde $a \neq 0$ e a, b e c são números reais. Uma observação importante é que, para todo real $a \neq 0$, a função quadrática pode ser reescrita na forma:

$$ax^2 + bx + c = a\left[\left(x + \frac{b}{2a}\right)^2 - \frac{\Delta}{4a^2}\right],$$

onde $\Delta = b^2 - 4ac$. Essa identidade nos permite concluir que, quando $a > 0$, $f(x) \geq -\Delta/4a$, para todo x real. Além disso, definindo $\overline{x} = -b/2a$, temos que $f(\overline{x}) = -\Delta/4a$, o que mostra que, para todo x real, $f(x) \geq f(\overline{x})$, ou seja, $\overline{x}$ é o ponto de mínimo global para a função f. De acordo com ZAHN (2022, p.281).

Definição 1. *Seja $f : I \to \mathbb{R}$ e $a \in \mathbb{R}$ um ponto interior. Dizemos que a é um ponto de mínimo local para f se existir $\delta > 0$ tal que $f(x) \geq f(a)$, para todo $x \in (a - \delta, a + \delta)$.*

Por outro lado, um ponto de mínimo global é um ponto x_0 onde $f(x_0) \leq f(x)$ para todos os x no domínio de f, ou seja, x_0 é o valor em que a função atinge seu valor mais baixo em todo o domínio. No contexto de funções quadráticas, $\overline{x} = -\dfrac{b}{2a}$ é não apenas um ponto de mínimo local, mas também o ponto mínimo global quando $a > 0$, pois a função quadrática tem um forma de parábola com uma abertura para cima, e assim, o valor de $f(\overline{x})$ é o menor valor que a função pode atingir. Em

uma função quadrática com $a > 0$ não há distinção prática entre mínimo local e global, já que a função possui uma única concavidade e o mínimo global é atingido em um único ponto. Assim, a determinação do ponto de mínimo de uma função quadrática, quando $a > 0$, se resume ao cálculo da constante $\overline{x} = -b/2a$. No entanto, para ilustrar a aplicação do estudo da programação matemática no ensino básico, propõe-se a utilização de um método computacional para calcular tal ponto de mínimo. Veremos a seguir como será essa abordagem.

3.1 O algoritmo

Durante todo este trabalho, iremos supor que $a > 0$. Destaque-se que quando $a < 0$, a função alcançará um ponto de máximo global, uma situação que não está dentro do escopo dos estudos deste artigo.

Inicialmente, definiremos a seguinte função auxiliar

$$F(\lambda, \gamma, x) = \left(\frac{2a\gamma + 1}{2a\lambda + 1}\right) x - \frac{(\lambda - \gamma)b}{2a\lambda + 1}. \tag{3.1}$$

onde λ e γ são constantes positivas satisfazendo o seguinte:

$$0 < 2\gamma < \lambda. \tag{3.2}$$

Esta função é derivada do Algoritmo do Ponto Proximal e consiste em uma versão que utiliza apenas matemática básica em sua formulação.

O algoritmo é definido da seguinte maneira:

1) PASSO INICIAL: Escolha qualquer $x_0 \in \mathbb{R}$ como ponto inicial.

2) PASSO ITERATIVO:

(a) Dado $x_n \in \mathbb{R}$, calcule $z_n \in \mathbb{R}$, satisfazendo

$$z_n = F(\lambda, \gamma, x_n). \tag{3.3}$$

(b) A iterada x_{n+1} é dada por

$$x_{n+1} = z_n - \frac{\gamma}{\lambda - \gamma}(x_n - z_n). \tag{3.4}$$

A proposta de implementação do algoritmo requer o uso de programação, uma disciplina essencial para a criação de instruções que um computador pode seguir para realizar tarefas específicas. A programação é um campo vasto que abrange desde a definição de algoritmos até a sua implementação em código executável por máquinas.

Neste contexto, vamos explorar alguns conceitos básicos que são fundamentais para iniciantes.

Um algoritmo é um conjunto de instruções com passos finitos que resolve problemas ou executa tarefas específicas. Ele é a base de toda a programação e é responsável por definir a lógica que será seguida pelo programa para atingir seu objetivo. A capacidade de elaborar algoritmos eficientes e corretos é uma das habilidades mais importantes para um programador. Por exemplo, um algoritmo simples para encontrar o maior número em uma lista de inteiros pode ser descrito em pseudocódigo da seguinte forma:

```
Algoritmo EncontrarMaiorNumero
Entrada: Lista de inteiros L
Saída: Maior número em L

Passos:
1. Inicializar o maior número como o primeiro elemento da lista:
  maior = L[0]
2. Para cada número n na lista L:
2.1. Se n for maior que maior, então:
2.1.1. Atualizar maior para n
3. Retornar maior
```

Linguagens de programação são notações formais com regras e sintaxes específicas que permitem a escrita de algoritmos. Elas são a ponte entre a lógica do algoritmo e a execução real pelo computador. Exemplos comuns de linguagens de programação incluem Python, JavaScript, Java e C++. Cada uma dessas linguagens possui características e aplicações particulares, adaptando-se a diferentes necessidades de desenvolvimento. Por exemplo, em Python, a implementação de um algoritmo para somar todos os elementos de uma lista seria:

```
# Python
def somar_elementos(lista):
soma = 0
for elemento in lista:
soma += elemento
return soma

lista = [1, 2, 3, 4, 5]
print(somar_elementos(lista))  # Saída: 15
```

A sintaxe de uma linguagem de programação é o conjunto de regras que define a estrutura das instruções que podem ser compreendidas pelo compilador ou interpretador. Um erro sintático pode impedir a execução do código, tornando-o não executável. Portanto, é crucial seguir rigorosamente as regras sintáticas de cada linguagem para garantir que o código funcione corretamente. Por exemplo, em Python, o uso correto da indentação é fundamental:

```
# Exemplo incorreto
numero = 10
if numero > 5:
print("O número é maior que 5")
else:
print("O número é 5 ou menor")

# Exemplo correto
numero = 10
if numero > 5:
    print("O número é maior que 5")
else:
    print("O número é 5 ou menor")
```

Variáveis são espaços de armazenamento que guardam dados que podem ser modificados durante a execução do programa. Cada variável possui um nome (identificador) e um valor associado, que pode ser alterado conforme necessário. O uso adequado de variáveis permite a manipulação eficiente dos dados no decorrer da execução de um programa. Em C, por exemplo, a declaração e o uso de variáveis podem ser feitos da seguinte forma:

```
// C
#include <stdio.h>

int main() {
     int a = 5;          // Declaração de variável inteira
     float b = 3.5;      // Declaração de variável float
     char c = 'A';       // Declaração de variável char

     printf("a: %d, b: %.2f, c: %c\n", a, b, c);
    // Saída: a: 5, b: 3.50, c: A
     return 0;
}
```

Os tipos de dados determinam os valores que podem ser armazenados e as operações que podem ser realizadas sobre esses valores. Tipos de dados comuns incluem inteiros (`int`), ponto flutuante (`float`), caracteres (`char`), strings (cadeias de caracteres) e booleanos (`bool`). A escolha adequada dos tipos de dados é essencial para garantir a eficiência e a precisão das operações realizadas pelo programa. Por exemplo, em Java:

```
// Java
public class TiposDeDados {
     public static void main(String[] args) {
         int inteiro = 10;              // Tipo inteiro
         double pontoFlutuante = 20.5;  // Tipo ponto flutuante
         char caractere = 'B';          // Tipo caractere
         String cadeia = "Olá, Mundo!"; // Tipo string
         boolean booleano = true;       // Tipo booleano

         System.out.println("Inteiro: " + inteiro);
        // Saída: Inteiro: 10
         System.out.println("Ponto flutuante: " + pontoFlutuante);
        // Saída: Ponto flutuante: 20.5
         System.out.println("Caractere: " + caractere);
        // Saída: Caractere: B
         System.out.println("String: " + cadeia);
        // Saída: String: Olá, Mundo!
         System.out.println("Booleano: " + booleano);
        // Saída: Booleano: true
     }
}
```

Operadores são símbolos que representam operações matemáticas, lógicas e de comparação. Operadores matemáticos como `+`, `-`, `*` e `/` são usados para realizar operações aritméticas. Operadores lógicos, como `&&` (E lógico), `||` (OU lógico) e `!` (não lógico), são utilizados para manipular valores booleanos. Já os operadores de comparação, como `==` (igual), `!=` (diferente), `<` (menor que) e `>` (maior que), são empregados para comparar valores e determinar relações entre eles. Por exemplo, em C++:

```
// C++
#include <iostream>
using namespace std;
```

```
int main() {
    int a = 10, b = 20;

    // Operadores matemáticos
    cout << "a + b = " << (a + b) << endl;  // Saída: a + b = 30
    cout << "a * b = " << (a * b) << endl;  // Saída: a * b = 200

    // Operadores de comparação
    cout << "a == b: " << (a == b) << endl;
    // Saída: a == b: 0 (falso)
    cout << "a < b: " << (a < b) << endl;
    // Saída: a < b: 1 (verdadeiro)

    // Operadores lógicos
    bool resultado = (a < b) && (a > 0);
    cout << "(a < b) && (a > 0): " << resultado << endl;
    // Saída: 1 (verdadeiro)

    return 0;
}
```

Os elementos de controle de fluxo direcionam a execução do programa, permitindo que diferentes blocos de código sejam executados com base em condições específicas. Condicionais como `if`, `else if` e `else` permitem a execução de blocos de código com base em condições. Laços (ou loops) como `for`, `while` e `do-while` permitem a repetição de blocos de código enquanto certas condições forem verdadeiras. Esses elementos são fundamentais para a construção de lógica complexa em programas. Por exemplo, em JavaScript:

```
// JavaScript
let x = 10;

// Condicional
if (x > 5) {
    console.log("x é maior que 5");
} else if (x < 5) {
    console.log("x é menor que 5");
} else {
    console.log("x é igual a 5");
```

```
}

// Laço for
for (let i = 0; i < 5; i++) {
    console.log("i = " + i);
}

// Laço while
let y = 0;
while (y < 5) {
    console.log("y = " + y);
    y++;
}
```

Funções são blocos de código reutilizáveis que executam uma tarefa específica. Elas podem receber entradas (parâmetros) e retornar um valor, o que permite a modularização e reutilização do código. A definição e o uso adequados de funções são cruciais para a criação de programas eficientes e fáceis de manter. Em Python, por exemplo:

```
# Python
def saudacao(nome):
    return f"Olá, {nome}!"

print(saudacao("Maria"))  # Saída: Olá, Maria!

def soma(a, b):
    return a + b

print(soma(3, 4))  # Saída: 7
```

Estruturas de dados são formas de organizar e armazenar dados de maneira eficiente, permitindo operações rápidas de acesso e modificação. Exemplos comuns incluem arrays (vetores), listas, filas, pilhas, árvores e dicionários. Em JavaScript, por exemplo:

```
// JavaScript
let array = [1, 2, 3, 4, 5];  // Array

array.push(6);  // Adiciona elemento ao final do array
console.log(array);  // Saída: [1, 2, 3, 4, 5, 6]
```

```
let fila = [];
fila.push(1);  // Enfileira elemento
fila.push(2);
console.log(fila.shift());  // Desenfileira elemento (Saída: 1)
console.log(fila);  // Saída: [2]
```

O processo de depuração envolve a identificação e correção de erros no código. Erros comuns incluem erros de sintaxe, erros de lógica e erros de tempo de execução. A depuração eficaz é essencial para garantir que o programa funcione conforme esperado e para evitar comportamentos indesejados. Em C++, por exemplo:

```
// C++
#include <iostream>
using namespace std;

int main() {
    int numero = 5;

    // Erro de lógica: a multiplicação deveria ser por 2
    int resultado = numero * 3;
    cout << "Resultado: " << resultado << endl; //Saída incorreta: 15

    // Correção: multiplicação por 2
    resultado = numero * 2;
    cout << "Resultado correto: " << resultado << endl;
    // Saída correta: 10

    return 0;
}
```

Comentários são textos dentro do código que não são executados e servem para documentar e explicar o funcionamento do programa. Eles são vitais para a manutenção e compreensão do código, especialmente em projetos grandes ou colaborativos. Por exemplo, em Java:

```
// Java
public class Comentarios {
    public static void main(String[] args) {
        // Este é um comentário de linha única
```

```
        /*
         * Este é um comentário de múltiplas linhas.
         * Pode ser utilizado para explicar partes mais complexas do
         *código.
         */

        // Exibindo mensagem
        System.out.println("Olá, Mundo!");  /* Comentário ao lado do
        código */
    }
}
```

A programação é uma ferramenta poderosa que, quando dominada, permite a criação de soluções eficientes para uma ampla variedade de problemas. A compreensão dos conceitos básicos apresentados aqui é um passo crucial para se tornar proficiente nesta área. Com a prática e o estudo contínuo, é possível desenvolver habilidades avançadas que abrirão muitas oportunidades no campo da computação.

Existem diversas abordagens e estilos de programação que podem ser utilizados de acordo com as necessidades de um projeto. Como, Programação Procedural, que se baseia na criação de procedimentos ou funções que são chamados sequencialmente para realizar tarefas específicas, enfatizando a execução de uma série de instruções ordenadas, facilitando assim o entendimento e a manutenção do fluxo lógico do programa; Programação Orientada a Objetos (POO), que organiza o código em objetos que encapsulam dados e comportamentos relacionados. Cada objeto é uma instância de uma classe e pode interagir com outros objetos, o que facilita a reutilização de código e a modelagem de sistemas complexos através da abstração, herança e polimorfismo; e Programação Funcional, que foca em funções puras, ou seja, funções sem efeitos colaterais e que retornam os mesmos resultados para os mesmos argumentos. Este estilo evita estados mutáveis e promove a composição de funções, facilitando a criação de programas mais previsíveis e fáceis de testar.

Para uma compreensão mais eficaz do processo iterativo descrito pelas equações (3.3) e (3.4), iremos empregar a linguagem de programação Java para demonstrar a implementação do algoritmo. Java é uma linguagem de programação de alto nível, orientada a objetos, lançada pela Sun Microsystems em 1995. É uma das plataformas mais utilizadas para o desenvolvimento de aplicações em todo o mundo. Segundo Sierra e Bates "o Java seduziu os programadores com sua sintexe amigavél, recursos orientados a objetos, gerenciamento de memória e, o melhor de tudo, a promessa de portabilidade"(SIERRA, K.; BATES, B.2010, p.1).

A programação Java é baseada no paradigma de orientação a objetos, permitindo a criação de classes e objetos com propriedades e comportamentos específicos.

Uma das principais vantagens do Java é sua portabilidade, garantida pela Máquina Virtual Java (JVM), que permite que o código seja executado em diferentes plataformas. Java possui uma extensa biblioteca de classes que facilitam o desenvolvimento, oferecendo soluções para situações comuns. Foi projetado com foco na segurança, incluindo recursos que ajudam a proteger contra erros comuns de programação. Embora o desempenho do Java tenha sido criticado no passado, melhorias significativas foram feitas ao longo dos anos, incluindo a implementação de um compilador Just-In-Time (JIT) na JVM. O Java Development Kit (JDK) é um conjunto de utilitários que suporta o desenvolvimento de aplicações em Java.

Sendo assim, empregaremos o método para desenvolver uma aplicação em Java. Esta aplicação servirá como uma aplicação prática, permitindo-nos observar e compreender o comportamento desse método durante cada interação do programa. Através desta abordagem, poderemos explorar gradativamente cada etapa do método em questão e avaliar seu desempenho em diferentes cenários dentro da nossa aplicação Java.

Exemplo 1. *Considere* $f(x) = x^2 + x + 1$ *e tome* $\lambda = 3$ *e* $\gamma = 1$. *Observe que as escolhas de* λ *e* γ *satisfazem a condição* (3.2). *Substituindo esses valores na expressão* (3.3), *obtém-se:*

$$\begin{aligned} z_n &= F(3, 1, x_n) \\ &= \frac{(2 \cdot 1 \cdot 1 + 1)}{(2 \cdot 1 \cdot 3 + 1)} x_n - \frac{(3-1) \cdot 1}{(2 \cdot 1 \cdot 3 + 1)} \\ &= \frac{3}{7} x_n - \frac{2}{7}. \end{aligned}$$

Consequentemente, usando a definição de x_n *em* (3.4) *pode-se deduzir que:*

$$\begin{aligned} x_{n+1} &= \frac{3}{7} x_n - \frac{2}{7} - \frac{1}{(3-1)} \left[x_n - \left(\frac{3}{7} x_n - \frac{2}{7} \right) \right] \\ &= \frac{3}{7} x_n - \frac{2}{7} - \frac{1}{2} \left(\frac{4}{7} x_n + \frac{2}{7} \right) \\ &= \frac{x_n}{7} - \frac{3}{7}. \end{aligned}$$

Supondo que a sequência $\{x_n\}$ *seja convergente, considere* $L = \lim_{n \to \infty} x_n$. *Assim,*

$$L = \lim_{n \to \infty} x_{n+1} = \lim_{n \to \infty} \left(\frac{x_n}{7} - \frac{3}{7} \right) = \frac{L}{7} - \frac{3}{7}.$$

Resolvendo essa equação, obtemos:

$$L = \frac{L}{7} - \frac{3}{7} \Leftrightarrow L = -\frac{1}{2}.$$

Portanto, caso a sequência $\{x_n\}$ convirja, necessáriamente ela deve convergir para $L = -1/2$. Por outro lado, note que o ponto de mínimo da função $f(x) = x^2 + x + 1$ é dado por $\overline{x} = -b/2a = -1/2$. Essa afirmação será demonstrada de forma geral pelo Teorema 1 na Seção 4. Podemos verificar este fato utilizando uma aplicação em Java. Para isso, começaremos com o ponto inicial $x_1 = 1$. Em seguida, testaremos o método com o ponto inicial $x_1 = -10$ para observar seu comportamento.

```
1: import java.util.Scanner;
2:
3: public class Algoritmo {
4:     public static void main(String[] args) {
5:         float xn, xn1, zn, lambda, gamma, a, b;
6:         int i = 0;
7:
8:         // Tolerância para o erro absoluto
9:         float tol = (float) (1.4 * Math.pow(10, -45));
10:
11:         try (Scanner ler = new Scanner(System.in)) {
12:             System.out.printf("Informe o valor de a: ");
13:             a = ler.nextFloat();
14:             System.out.printf("Informe o valor de b: ");
15:             b = ler.nextFloat();
16:             System.out.printf("Informe o valor de lambda: ");
17:             lambda = ler.nextFloat();
18:             System.out.printf("Informe o valor de gamma: ");
19:             gamma = ler.nextFloat();
20:             System.out.printf("Informe o valor de xn: ");
21:             xn = ler.nextFloat();
22:         }
23:
24:         while (true) {
25:             zn = ((2 * a * gamma + 1) / (2 * a * lambda + 1)) *
                    xn - ((lambda - gamma) * b)/(2 * a * lambda + 1);
26:
27:             xn1 = zn - (gamma / (lambda - gamma)) * (xn - zn);
28:             System.out.println("Para x" + i + " = " + xn +
                ", temos x" + (i + 1) + " = " + xn1 + "\n");
29:             i = i+1; //atualiza o índice dos termos da sequência
30:
31:             float abserr = Math.abs(xn1 - xn);
```

```
32:             float relerr = abserr / Math.abs(xn1);
33:
34:             /* Condição de parada: pelo menos uma das
                condições deve ser satisfeita */
35:             if (abserr <= tol || relerr <= tol) {
36:                 break;
37:             }
38:
39:             // Atualiza xn para a próxima iteração
40:             xn = xn1;
41:         }
42:     }
43: }
```

Disponibilizamos esse código-fonte em:
https://github.com/marciojqc/TCC/blob/main/AplicacaoJava
Observe que esta aplicação é uma versão simples com baixa precisão decimal e sem tratamento de erros. O foco principal é facilitar o entendimento na educação básica, garantindo simplicidade e clareza. Para uma versão mais precisa, com tratamento de erros, acesse:
https://github.com/marciojqc/TCC/blob/main/AlgortimoDePrecisao

A saída para os valores informados tomando $x_1 = 1$ são os seguintes:

Tabela 3.1: Resultados para $x_1 = 1$

n	x_n	x_{n+1}
1	$x_1 = 1$	$x_2 = -0,2857143$
2	$x_2 = -0,2857143$	$x_3 = -0,46938777$
3	$x_3 = -0,46938777$	$x_4 = -0,49562684$
4	$x_4 = -0,49562684$	$x_5 = -0,49937528$
5	$x_5 = -0,49937528$	$x_6 = -0,49991077$
6	$x_6 = -0,49991077$	$x_7 = -0,4999873$
7	$x_7 = -0,4999873$	$x_8 = -0,4999982$
8	$x_8 = -0,4999982$	$x_9 = -0,49999976$
9	$x_9 = -0,49999976$	$x_{10} = -0,5$
10	$x_{10} = -0,5$	$x_{11} = -0,5$

Fonte: Elaborada pelo autor.

Observe que no caso em que escolhemos $x_1 = -10$ tivemos um resultado de convergência aproximado, é importante entender que Java utiliza representação de ponto flutuante para números reais, como os tipos float e double. Essa representação é limitada em precisão e, portanto, sujeita a erros de arredondamento.

Tabela 3.2: Resultados para $x_1 = -10$

n	x_n	x_{n+1}
1	$x_1 = -10$	$x_2 = -1,8571424$
2	$x_2 = -1,8571424$	$x_3 = -0,6938775$
3	$x_3 = -0,6938775$	$x_4 = -0,52769685$
4	$x_4 = -0,52769685$	$x_5 = -0,5039567$
5	$x_5 = -0,5039567$	$x_6 = -0,5005653$
6	$x_6 = -0,5005653$	$x_7 = -0,5000808$
7	$x_7 = -0,5000808$	$x_8 = -0,5000116$
8	$x_8 = -0,5000116$	$x_9 = -0,50000167$
9	$x_9 = -0,50000167$	$x_{10} = -0,50000024$
10	$x_{10} = -0,50000024$	$x_{11} = -0,50000006$
11	$x_{11} = -0,50000006$	$x_{12} = -0,50000006$

Fonte: Elaborada pelo autor.

Durante cálculos matemáticos, os números são arredondados para se ajustarem à representação de ponto flutuante. Esses erros podem se acumular, especialmente em cálculos iterativos. Por exemplo, ao calcular raízes ou aproximações de funções, pequenas imprecisões podem afetar a convergência do resultado.

Além disso, a comparação direta de números de ponto flutuante usando operadores como == [1] *pode ser problemática. Devido à precisão limitada, números que deveriam ser iguais podem ser considerados diferentes. É recomendável usar uma tolerância ao comparar números de ponto flutuante, verificando se a diferença entre eles é menor que um valor pequeno.*

Outro fator é a implementação específica da JVM (Java Virtual Machine). Diferentes implementações da JVM (como Oracle HotSpot, OpenJ9 e GraalVM) podem ter estratégias de cálculo distintas, o que afeta os resultados numéricos.

Em resumo, a precisão limitada de ponto flutuante, erros de arredondamento e a forma como a JVM lida com cálculos podem contribuir para resultados aproximados em testes numéricos de convergência em Java. É essencial estar ciente dessas limitações e adotar práticas adequadas para minimizar esses efeitos.

Para um caso geral ou para escolha de outra linguagem de programação podemos utilizar o seguinte algoritmo (Pseudo-código) para a aplicação:

```
1: INÍCIO
2: // Declaração de variáveis
3: DECLARE xn, xn1, zn, lambda, gamma, a, b COMO NÚMEROS REAIS
4:  DECLARE i COMO INTEIRO
```

[1] Na linguagem de programação Java, o símbolo == é um operador de comparação que verifica se dois valores são iguais

```
5:  i = 0
6:  DECLARE tol COMO NÚMERO REAL
7:  tol = 1.4 * 10^-45 // Tolerância para o erro absoluto
8:
9:  // Criação do objeto para leitura de entrada do usuário
10:  CRIE um objeto Scanner chamado ler para ler a entrada do usuário
11:
12:  // Leitura de valores
13:  IMPRIMA "Informe o valor de a:"
14:  LEIA a
15:  IMPRIMA "Informe o valor de b:"
16:  LEIA b
17:  IMPRIMA "Informe o valor de lambda:"
18:  LEIA lambda
19:  IMPRIMA "Informe o valor de gamma:"
20:  LEIA gamma
21:  IMPRIMA "Informe o valor de xn:"
22:  LEIA xn
23:
24:  // Laço principal
25:  ENQUANTO VERDADEIRO FAÇA
26:    // Cálculo de zn e xn1
27:    zn = ((2 * a * gamma + 1) / (2 * a * lambda + 1)) * xn -
           ((lambda - gamma) * b) / (2 * a  * lambda + 1)
28:    xn1 = zn - (gamma / (lambda - gamma)) * (xn - zn)
29:
30:    // Impressão dos resultados
31:    IMPRIMA "Para x" + i + "= " + xn + ", temos x" + (i + 1) + "= "
                 + xn1
32:
33:    // Incremento do índice
34:    i = i + 1
35:
36:    // Cálculo do erro absoluto
37:    DECLARE abserr COMO NÚMERO REAL
38:    abserr = |xn1 - xn|
39:
40:    // Cálculo do erro relativo
41:    DECLARE relerr COMO NÚMERO REAL
```

```
42:     relerr = abserr / |xn1|
44:
45:     // Condição de parada
46:     SE abserr <= tol OU relerr <= tol ENTÃO
47:       PARE
48:     FIM SE
49:
50:     // Atualização do valor de xn
51:     xn = xn1
52:
53:   FIM ENQUANTO
54: FIM
```

Disponibilizamos esse pseudocódigo em:
https://github.com/marciojqc/TCC/blob/main/Pseudo-Codigo

Segundo Deitel, "O pseudocódigo é um liguagem artificial informal que ajuda os programadores a desenvolver algoritmos."(DEITEL2001, p.151)

O pseudocódigo pode auxiliar professores do Novo Ensino Médio, especialmente aqueles que estão na *Trilha de Tecnologias Digitais*, como os do estado de Pernambuco, por exemplo. Esses professores oferecem unidades curriculares como *Linguagem de Programação C*, *Programação com Python* e *Raciocínio Lógico e Pensamento Computacional*. Com base na ementas dessas disicplinas. Para uma abordagem mais detalhada veja [11].

Podemos observar que ensino de programação, como evidenciado pelas disciplinas de "Linguagem de Programação C" e "Programação com Python", é fundamental para equipar os alunos com as habilidades necessárias para resolver problemas complexos. Através da programação, os alunos aprendem a decompor problemas, identificar padrões e desenvolver soluções eficientes. Além disso, a programação ensina os alunos a pensar logicamente e a fazer inferências válidas, habilidades que são transferíveis para muitos outros campos. A disciplina de "Raciocínio Lógico e Pensamento Computacional" complementa isso, fornecendo uma compreensão mais ampla e conceitual da resolução de problemas. Juntas, essas disciplinas preparam os alunos para uma variedade de desafios, permitindo-lhes aplicar suas habilidades de programação e pensamento lógico para encontrar soluções inovadoras.

Desse modo, o pseudocódigo funciona como uma receita para o código-fonte de qualquer linguagem de programação. Além disso, é possível utilizar uma IA - Inteligência Artificial para criar o código a partir do pseudocódigo.

O critério de parada da convergência para o ponto fixo desempenha um papel fundamental na determinação do momento adequado para interromper o método iterativo. Esse critério irá avaliar se a aproximação atual está suficientemente pró-

xima da raiz exata, ou se é igual a mesma, no nosso caso do vértice da função, garantindo assim a precisão e a eficiência do processo de resolução. Além disso, é importante ressaltar que pelo Teorema da Convergência Monótona, que afirma que toda sequência monótona e limitada é convergente, ou seja, se a sequência de aproximações geradas pelo nosso método iterativo for monótona e limitada, como de fato(provaremos na Seção 4), podemos garantir que ela irá convergir para o valor desejado. Utilizaremos o seguinte critério de parada: $|x_{n+1} - x_n| < \varepsilon$, sendo $\varepsilon > 0$. Se a condição acima for satisfeita, o método para. Utilizaremos ε na aplicação como o menor valor possível para a variável de tipo float, nesse caso $\varepsilon = 1,4 \cdot 10^{-45}$.

Em contrapartida à linguagem de programação Java, o GeoGebra surge como uma alternativa viável para aqueles que não possuem familiaridade com linguagens de programação. Este software de matemática dinâmica é uma ferramenta acessível que permite a exploração interativa de conceitos matemáticos complexos, combinando elementos de geometria, álgebra, planilhas, gráficos, estatísticas e cálculos.

É importante ressaltar que o "Novo Ensino Médio", mais especificamente no estado de Pernambuco, oferece uma unidade curricular chamada *Softwares de Simulação Matemática* inserida na *Trilha de Tecnológias Digitais*. De acordo com a Secretaria Executiva de Desenvolvimento da Educação (2023), a ementa desta trilha é:

> Levantamento e testagem de hipóteses sobre variáveis que interferem na resolução de situações-problema e na modelagem, utilizando os conceitos matemáticos e da lógica. Seleção e mobilização de recursos criativos relacionados à matemática. Utilização de softwares e aplicativos como ferramenta de simulação que explorem a linguagem matemática na elaboração de modelos (Geogebra, Cabri Géomètre, Winplot etc.). (SED, 2023, p. X)

Embora a linguagem de programação Java seja conhecida por sua alta precisão matemática, ela pode ser desafiadora para aqueles sem experiência prévia em programação. Nesse contexto, o GeoGebra se destaca como uma ferramenta eficaz para visualização e exploração matemática.

Com base nos dados fornecidos no *Exemplo 1*, as seguintes entradas serão inseridas no software GeoGebra:

- **Entrada 1:** $f(x) = x^2 + x + 1$;
- **Entrada 2:** $x_1 = 1$;
- **Entrada 3:** $z_n = \frac{3x}{7} - \frac{2}{7}$;

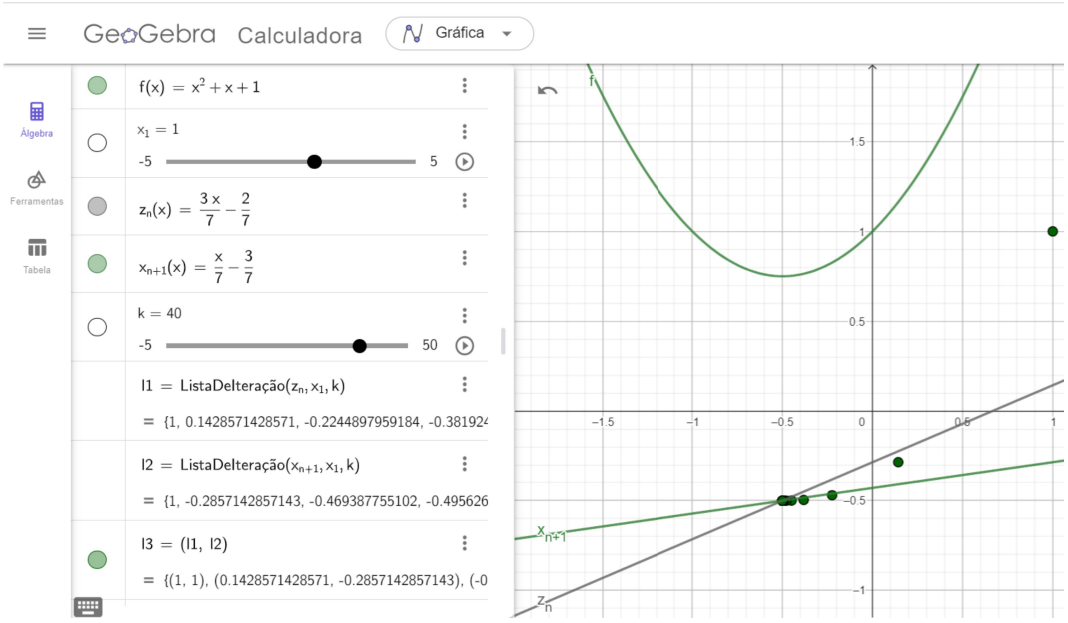
- **Entrada 4:** $x_{n+1} = \frac{x}{7} - \frac{3}{7}$;
- **Entrada 5:** $k = 40$;
- **Entrada 6:** $l_1 = \text{ListaDeIteração}(z_n, x_1, k)$;
- **Entrada 7:** $l_2 = \text{ListaDeIteração}(x_{n+1}, x_1, k)$;
- **Entrada 8:** $l_3 = (l_1, l_2)$;

As **Entradas 2** e **5** são designadas para representar os valores iniciais de x_1 e o número total de iterações k, respectivamente. Ajustes nesses valores podem ser realizados para otimizar a visualização dos pontos de acumulação.

As **Entradas 6** e **7** são responsáveis por gerar as listas de iterações, que são criadas a partir das funções z_n e x_{n+1}, utilizando o valor inicial x_1.

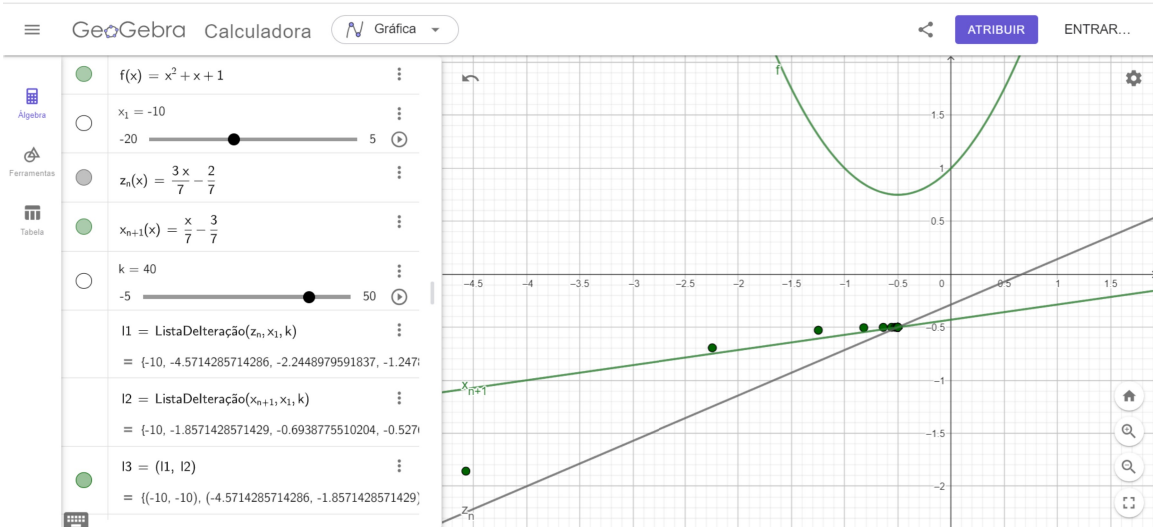
Por fim, a **Entrada 8** é uma lista composta pelas listas l_1 e l_2. Esta entrada é particularmente útil, pois permite a visualização dos pontos de interação entre as duas listas de iterações.

Figura 3.1: Gráfico para $x_1 = 1$, $\lambda = 3$ e $\gamma = 1$



Fonte: Elaborada pelo autor utilizando o GeoGebra.

Figura 3.2: Gráfico para $x_1 = -10$, $\lambda = 3$ e $\gamma = 1$



Fonte: Elaborada pelo autor utilizando o GeoGebra.

Com base nos gráficos da Figura 1 e da Figura 2, podemos observar a rápida taxa de acumulação dos pontos para os diferentes valores de x_1. Na lista l_3 teremos o seguinte conjunto de pontos para $x_1 = 1$ (Figura 1):

Tabela 3.3: Resultados da lista l_3 para $x_1 = 1$.

k	l_1	l_2
1	1	1
2	0.1428571428571	-0.2857142857143
3	-0.2244897959184	-0.469387755102
4	-0.3819241982507	-0.4956268221574
5	-0.4493960849646	-0.4993752603082
6	-0.478312607842	-0.4999107514726
7	-0.4907054033608	-0.4999872502104
8	-0.4960166014404	-0.4999981786015
9	-0.4982928291887	-0.4999997398002
10	-0.4992683553666	-0.4999999628286

Fonte: Elaborada pelo autor.

E para $x_1 = -10$ (Figura 2) teremos:

Tabela 3.4: Resultados da lista l_3 para $x_1 = -10$.

k	l_1	l_2
1	-10	-10
2	-4.5714285714286	-1.8571428571429
3	-2.2448979591837	-0.6938775510204
4	-1.2478134110787	-0.5276967930029
5	-0.8204914618909	-0.5039566847147
6	-0.6373534836675	-0.5005652406735
7	-0.5588657787147	-0.5000807486676
8	-0.5252281908777	-0.5000115355239
9	-0.5108120818047	-0.500001647932
10	-0.5046337493449	-0.5000002354189

Fonte: Elaborada pelo autor.

Observe que quanto mais próximo de $\bar{x}$ se escolhe x_1, mais próximo dele se iniciam os pontos de acumulação.

Capítulo 4

Resultados de convergência

Nesta seção, apresentaremos os resultados de convergência do método estudado. Iniciaremos com dois resultados auxiliares. Em seguida, demonstraremos que a sequência gerada pelo processo iterativo (3.3) e (3.4) converge para o ponto mínimo de uma função quadrática, desde que as constantes λ e γ satisfaçam a condição (3.2).

Lema 1. *Seja $\{z_n\}$ a sequência definida em (3.3). Então, valem os seguintes itens:*

(a) *Para todo n natural, z_n pode ser reescrito como segue:*

$$z_n = x_n - \frac{\lambda - \gamma}{\gamma}\left(z_n - \frac{z_n - b\gamma}{2a\gamma + 1}\right). \tag{4.1}$$

(b) *Se $\overline{x} = -\dfrac{b}{2a}$, tem-se:*

$$x_n - z_n = \left(\frac{\lambda - \gamma}{\gamma}\right)\left(\frac{2a\gamma}{2a\gamma + 1}\right)(z_n - \overline{x}). \tag{4.2}$$

Demonstração. Usando a definição da função F em (3.1), é possível escrever a expressão de z_n, definida em (3.3), da seguinte forma

$$z_n = \left(\frac{2a\gamma + 1}{2a\lambda + 1}\right) x_n - \frac{(\lambda - \gamma)b}{2a\lambda + 1}.$$

A partir de agora, efetuaremos algumas manipulações algébricas até obtermos o resultado desejado. Primeiramente, observe que

$$\left(\frac{2a\lambda + 1}{2a\gamma + 1}\right) z_n + \frac{(\lambda - \gamma)b}{2a\gamma + 1} = x_n. \tag{4.3}$$

Observe que,

$$\begin{aligned}\left(\frac{2a\lambda+1}{2a\gamma+1}\right)z_n &= \frac{(2a\lambda\gamma+\gamma)z_n}{(2a\gamma+1)\gamma}\\ &= \frac{(2a\lambda\gamma+\lambda-\lambda+\gamma)z_n}{(2a\gamma+1)\gamma}\\ &= \frac{(2a\lambda\gamma+\lambda)z_n}{(2a\gamma+1)\gamma} - \frac{(\lambda-\gamma)z_n}{(2a\gamma+1)\gamma}\\ &= \frac{\lambda}{\gamma}z_n - \frac{(\lambda-\gamma)}{\gamma(2a\gamma+1)}z_n\end{aligned}$$

Desse modo, obtemos

$$\left(\frac{2a\lambda+1}{2a\gamma+1}\right)z_n = \frac{\lambda}{\gamma}z_n - \frac{(\lambda-\gamma)}{\gamma(2a\gamma+1)}z_n \qquad \text{e} \qquad \frac{(\lambda-\gamma)b}{2a\gamma+1} = \frac{(\lambda-\gamma)\gamma b}{\gamma(2a\gamma+1)}. \tag{4.4}$$

Substituindo as equações em (4.4) na expressão (4.3), obtemos

$$\frac{\lambda}{\gamma}z_n - \frac{(\lambda-\gamma)}{\gamma(2a\gamma+1)}z_n + \frac{(\lambda-\gamma)\gamma b}{\gamma(2a\gamma+1)} = x_n. \tag{4.5}$$

Uma vez que,

$$\frac{\lambda}{\gamma}z_n = z_n + \left(\frac{\lambda-\gamma}{\gamma}\right)z_n,$$

pode-se deduzir de (4.5) que

$$z_n + \left(\frac{\lambda-\gamma}{\gamma}\right)z_n - \left(\frac{\lambda-\gamma}{\gamma}\right)\frac{z_n-\gamma b}{2a\gamma+1} = x_n.$$

Portanto,

$$z_n = x_n - \left(\frac{\lambda-\gamma}{\gamma}\right)\left(z_n - \frac{z_n-\gamma b}{2a\gamma+1}\right).$$

Isso prova o item (a).

Agora, provaremos o item (b). De (a) temos que,

$$x_n - z_n = \left(\frac{\lambda-\gamma}{\gamma}\right)\left(\frac{2az_n\gamma + z_n - z_n + \gamma b}{2a\gamma+1}\right).$$

O que implica em,

$$x_n - z_n = \left(\frac{\lambda-\gamma}{\gamma}\right)\frac{1}{(2a\gamma+1)}\left(2az_n\gamma + \frac{2a}{2a}\gamma b\right).$$

Assim,

$$x_n - z_n = \left(\frac{\lambda-\gamma}{\gamma}\right)\frac{2a\gamma}{(2a\gamma+1)}\left(z_n + \frac{b}{2a}\right).$$

Como $\overline{x} = -\dfrac{b}{2a}$, concluí-se que

$$x_n - z_n = \left(\frac{\lambda - \gamma}{\gamma}\right)\left(\frac{2a\gamma}{2a\gamma + 1}\right)(z_n - \overline{x})\,.$$

Isso prova o item (b). □

Lema 2. *Seja* $\overline{x} = -\dfrac{b}{2a}$. *Então, para todo* $n \in \mathbb{N}$,

$$\left(\frac{4a\lambda}{2a\gamma + 1}\right)|z_n - \overline{x}|^2 = |x_n - \overline{x}|^2 - |x_{n+1} - \overline{x}|^2 - \left(1 - \left(\frac{\gamma}{\lambda - \gamma}\right)^2\right)|x_n - z_n|^2\,. \tag{4.6}$$

Demonstração. Seja $z_n = F(\lambda, \gamma, x_n)$. Inicialmente, note que,

$$\begin{aligned}
|x_n - \overline{x}|^2 &= x_n^2 - 2x_n\overline{x} + \overline{x}^2 \\
&= x_n^2 - 2x_nz_n + z_n^2 + z_n^2 - 2\overline{x}z_n + \overline{x}^2 + 2x_nz_n - 2z_n^2 - 2x_n\overline{x} + 2\overline{x}z_n \\
&= (x_n - z_n)^2 + (z_n - \overline{x})^2 + 2\left(x_nz_n - z_n^2 - x_n\overline{x} + \overline{x}z_n\right) \\
&= (x_n - z_n)^2 + (z_n - \overline{x})^2 + 2\left[z_n(x_n - z_n) - \overline{x}(x_n - z_n)\right].
\end{aligned}$$

Consequentemente,

$$|x_n - \overline{x}|^2 = |x_n - z_n|^2 + |z_n - \overline{x}|^2 + 2\,(x_n - z_n)\,(z_n - \overline{x})\,. \tag{4.7}$$

Pelo Lema 1 item (b), tem-se

$$x_n - z_n = \left(\frac{\lambda - \gamma}{\gamma}\right)\left(\frac{2a\gamma}{2a\gamma + 1}\right)(z_n - \overline{x})\,. \tag{4.8}$$

Combinando as equações (4.7) e (4.8), ontém-se

$$|x_n - \overline{x}|^2 = |x_n - z_n|^2 + |z_n - \overline{x}|^2 + 2\left(\frac{\lambda - \gamma}{\gamma}\right)\left(\frac{2a\gamma}{2\mathrm{a}\gamma + 1}\right)|z_n - \overline{x}|^2.$$

Consequentemente,

$$|z_n - \overline{x}|^2 = |x_n - \overline{x}|^2 - |x_n - z_n|^2 - \left(\frac{\lambda - \gamma}{\gamma}\right)\left(\frac{4a\gamma}{2a\gamma + 1}\right)|z_n - \overline{x}|^2. \tag{4.9}$$

Por outro lado, observe que

$$\begin{aligned}|x_{n+1}-\overline{x}|^2 &= x_{n+1}^2 - 2x_{n+1}\overline{x} + \overline{x}^2 \\ &= x_{n+1}^2 - 2x_{n+1}z_n + z_n^2 + z_n^2 - 2z_n\overline{x} + \overline{x}^2 + 2x_{n+1}z_n - 2z_n^2 - 2x_{n+1}\overline{x} + 2z_n\overline{x} \\ &= (x_{n+1}-z_n)^2 + (z_n-\overline{x})^2 + 2\left(x_{n+1}z_n - z_n^2 - x_{n+1}\overline{x} + z_n\overline{x}\right) \\ &= (x_{n+1}-z_n)^2 + (z_n-\overline{x})^2 + 2\left[z_n(x_{n+1}-z_n) - \overline{x}(x_{n+1}-z_n)\right].\end{aligned}$$

Isso implica que

$$|x_{n+1}-\overline{x}|^2 = |x_{n+1}-z_n|^2 + |z_n-\overline{x}|^2 + 2(x_{n+1}-z_n)(z_n-\overline{x}) \tag{4.10}$$

Combinando as equações (3.4) e (4.10), obtém-se

$$|x_{n+1}-\overline{x}|^2 = \left(\frac{\gamma}{\lambda-\gamma}\right)^2 |x_n-z_n|^2 + |z_n-\overline{x}|^2 - \frac{2\gamma}{\lambda-\gamma}(x_n-z_n)(z_n-\overline{x}). \tag{4.11}$$

Agora, substituindo (4.8) em (4.11), obtemos

$$|x_{n+1}-\overline{x}|^2 = \left(\frac{\gamma}{\lambda-\gamma}\right)^2 |x_n-z_n|^2 + |z_n-\overline{x}|^2 - \frac{2\gamma}{(\lambda-\gamma)}\left(\frac{\lambda-\gamma}{\gamma}\right)\left(\frac{2a\gamma}{2a\gamma+1}\right)|z_n-\overline{x}|^2.$$

Assim,

$$|x_{n+1}-\overline{x}|^2 = \left(\frac{\gamma}{\lambda-\gamma}\right)^2 |x_n-z_n|^2 + |z_n-\overline{x}|^2 - \left(\frac{4a\gamma}{2a\gamma+1}\right)|z_n-\overline{x}|^2. \tag{4.12}$$

Substituindo (4.9) em (4.12),

$$\begin{aligned}|x_{n+1}-\overline{x}|^2 &= \left(\frac{\gamma}{\lambda-\gamma}\right)^2 |x_n-z_n|^2 + |x_n-\overline{x}|^2 - |x_n-z_n|^2 \\ &\quad - \left(\frac{\lambda-\gamma}{\gamma}\right)\left(\frac{4a\gamma}{2a\gamma+1}\right)|z_n-\overline{x}|^2 - \left(\frac{4a\gamma}{2a\gamma+1}\right)|z_n-\overline{x}|^2 \\ &= |x_n-\overline{x}|^2 - \left(1-\left(\frac{\gamma}{\lambda-\gamma}\right)^2\right)|x_n-z_n|^2 \\ &\quad - \left(1+\left(\frac{\lambda-\gamma}{\gamma}\right)\right)\left(\frac{4a\gamma}{2a\gamma+1}\right)|z_n-\overline{x}|^2 \\ &= |x_n-\overline{x}|^2 - \left(1-\left(\frac{\gamma}{\lambda-\gamma}\right)^2\right)|x_n-z_n|^2 - \frac{\lambda}{\gamma}\left(\frac{4a\gamma}{2a\gamma+1}\right)|z_n-\overline{x}|^2.\end{aligned}$$

Portanto,

$$\left(\frac{4a\lambda}{2a\gamma+1}\right)|z_n-\overline{x}|^2 = |x_n-\overline{x}|^2 - |x_{n+1}-\overline{x}|^2 - \left(1-\left(\frac{\gamma}{\lambda-\gamma}\right)^2\right)|x_n-z_n|^2.$$

□

Teorema 1. *Seja* $\{x_n\}$ *a sequência gerada pelo processo iterativo* (3.3) *e* (3.4). *Então, para todo* n,

$$|x_n - \overline{x}| \leq \theta^n |x_0 - \overline{x}|, \tag{4.13}$$

onde

$$\theta = \sqrt{1 - \left(\frac{4a\lambda}{2a\gamma + 1}\right)\left(\frac{2a\gamma + 1}{2a\lambda + 1}\right)^2}. \tag{4.14}$$

Além disso, a sequência $\{x_n\}$ *converge para* $\overline{x} = -b/2a$.

Demonstração. Inicialmente, vale ressaltar que pela expressão (4.14), tem-se que $\theta < 1$. Além disso, θ é uma constante positiva. Com efeito, note que

$$\frac{(2a\lambda + 1)^2}{4a\lambda} = \frac{4a^2\lambda^2 + 4a\lambda + 1}{4a\lambda} > a\lambda + 1.$$

Como $\lambda > 2\gamma$, temos $\dfrac{(2a\lambda + 1)^2}{4a\lambda} > 2a\gamma + 1$. Assim,

$$\begin{aligned}
\frac{(2a\lambda + 1)^2}{4a\lambda} > 2a\gamma + 1 &\Leftrightarrow \left(\frac{2a\lambda + 1}{2a\gamma + 1}\right)^2 > \frac{4a\lambda}{2a\gamma + 1} \\
&\Leftrightarrow \left(\frac{2a\gamma + 1}{2a\lambda + 1}\right)^2 \left(\frac{4a\lambda}{2a\gamma + 1}\right) < 1 \\
&\Leftrightarrow \theta > 0.
\end{aligned}$$

Agora provaremos (4.13). Pela hipótese (3.2), pode-se deduzir que

$$\lambda > 2\gamma \Rightarrow \left(1 - \left(\frac{\gamma}{\lambda - \gamma}\right)^2\right) > 0.$$

Daí, a partir de (4.6), obtemos

$$\left(\frac{4a\lambda}{2a\gamma + 1}\right) |z_n - \overline{x}|^2 < |x_n - \overline{x}|^2 - |x_{n+1} - \overline{x}|^2 . \tag{4.15}$$

Usando o item (b) do Lema 1, obtém-se

$$\begin{aligned}
x_n - \overline{x} &= (x_n - z_n) + (z_n - \overline{x}) \\
&= \left(\frac{\lambda - \gamma}{\gamma}\right)\left(\frac{2a\gamma}{2a\gamma + 1}\right)(z_n - \overline{x}) + (z_n - \overline{x}) \\
&= \left(\frac{2a\lambda + 1}{2a\gamma + 1}\right)(z_n - \overline{x}) .
\end{aligned}$$

Logo,

$$(z_n - \overline{x}) = \left(\frac{2a\gamma + 1}{2a\lambda + 1}\right)(x_n - \overline{x}). \tag{4.16}$$

Substituindo (4.16) em (4.15), obtemos

$$\left(\frac{4a\lambda}{2a\gamma + 1}\right)\left(\frac{2a\gamma + 1}{2a\lambda + 1}\right)^2 |x_n - \overline{x}|^2 < |x_n - \overline{x}|^2 - |x_{n+1} - \overline{x}|^2 .$$

Assim,

$$|x_{n+1} - \overline{x}| < \sqrt{1 - \left(\frac{4a\lambda}{2a\gamma + 1}\right)\left(\frac{2a\gamma + 1}{2a\lambda + 1}\right)^2}\,|x_n - \overline{x}|. \tag{4.17}$$

O que implica em $|x_{n+1} - \overline{x}| < \theta|x_n - \overline{x}|$, para todo n natural.
Multiplicando a desigualdade (4.17) membro a membro, obtemos,

$$|x_1 - \overline{x}||x_2 - \overline{x}| \ldots |x_{n-1} - \overline{x}||x_n - \overline{x}| < \theta^n |x_0 - \overline{x}||x_1 - \overline{x}||x_2 - \overline{x}| \ldots |x_{n-1} - \overline{x}|.$$

Desse modo, para todo n natural,

$$|x_n - \overline{x}| < \theta^n |x_0 - \overline{x}|.$$

Isso prova que (4.13) vale para todo n natural. O fato da sequência $\{x_n\}$ convergir para $\overline{x}$, segue facilmente de (4.13), uma vez que $\lim_{n\to\infty} \theta^n = 0$. □

Exemplo 2. *Usando a estimativa* (4.13) *observamos que à medida que* θ^n *converge para* 0*, a sequência se aproxima gradualmente do ponto mínimo, note que a convergência é influenciada pelos valores de* λ *e* γ*, de modo que quanto maior o valor de* λ *e maior a diferença entre* λ *e* γ *mais rápido ocorre a convegência. Usando a função* $f(x) = x^2 + x + 1$ *do* ***Exemplo 1*** *e acrescentando a seguinte função ao loop da nossa aplicação, podemos facilmente observar esse fato:*

```
public static double Funcao_Theta (double _a, double _lambda,
double _gamma,int _i)
    {
        double _theta =
        Math.pow(Math.sqrt(1 - ((4*_a*_lambda)/(2*_a*_gamma + 1))*
        Math.pow((2*_a*_gamma + 1)/(2*_a*_lambda + 1), 2)),_i);

        return _theta;
    }
```

Criando a variável θ e atribuindo a ela os valores da função criada da seguinte forma e valores:

```
theta = Funcao_Theta(a, lambda, gamma, i);
```

Obteremos os seguintes resultados:

Tabela 4.1: Resultados para $x_1 = 1$, $\lambda = 3$ e $\gamma = 1$

n	x_n	x_{n+1}	θ^n
1	$x_1 = 1,0$	$x_2 = -0.2857143$	$\theta^1 = 0.5150787536377128$
2	$x_2 = -0,2857143$	$x_3 = -0,46938777$	$\theta^2 = 0,26530612244897966$
3	$x_3 = -0,46938777$	$x_4 = -0,49562684$	$\theta^3 = 0,13665354688347486$
4	$x_4 = -0,49562684$	$x_5 = -0,49937528$	$\theta^4 = 0,070387338608913$
5	$x_5 = -0,49937528$	$x_6 = -0,49991077$	$\theta^5 = 0,03625502264255457$
6	$x_6 = -0,49991077$	$x_7 = -0,4999873$	$\theta^6 = 0,018674191875834065$
7	$x_7 = -0,4999873$	$x_8 = -0,4999982$	$\theta^7 = 0,009618679476596113$
8	$x_8 = -0,4999982$	$x_9 = -0,49999976$	$\theta^8 = 0,004954377436445774$
9	$x_9 = -0,49999976$	$x_{10} = -0,5$	$\theta^9 = 0,0025518945550152957$
10	$x_{10} = -0,5$	$x_{11} = -0,5$	$\theta^{10} = 0,0013144266668121444$

Fonte: Elaborada pelo autor.

Observe que quanto maior a diferença entre λ e γ mais rápido θ^n converge para 0, assim como mais rápida se torna a convergência do método.

Tabela 4.2: Resultados para $x_1 = 1$, $\lambda = 100$ e $\gamma = 49$.

n	x_n	x_{n+1}	θ^n
1	$x_1 = 1$	$x_2 = -0,4925373$	$\theta^1 = 1,0$
2	$x_2 = -0,4925373$	$x_3 = -0,49996287$	$\theta^2 = 0,140805688538158 67$
3	$x_3 = -0,49996287$	$x_4 = -0,4999998$	$\theta^3 = 0,019826241924704946$
4	$x_4 = -0,4999998$	$x_5 = -0,49999997$	$\theta^4 = 0,0027916476453321883$
5	$x_5 = -0,49999997$	$x_6 = -0,5$	$\theta^5 = 3,9307986885692815 \times 10^{-4}$

Fonte: Elaborada pelo autor.

Capítulo 5

Considerações finais

A incorporação da programação no ensino básico é uma estratégia essencial para preparar os alunos para um futuro dominado pela tecnologia. A integração de conceitos computacionais e matemáticos, especificamente através da programação e do estudo de funções quadráticas, não apenas melhora as habilidades analíticas e criativas dos estudantes, mas também democratiza o acesso à tecnologia e impulsiona a inclusão digital.

O método de minimização de funções quadráticas explorado neste trabalho exemplifica como conceitos matemáticos podem ser aplicados de maneira prática na programação. A escolha do Java como linguagem de programação e do software GeoGebra para visualização e análise gráfica permite que estudantes e educadores explorem o método de forma mais aplicada. A utilização do GeoGebra, em particular, facilita a compreensão visual das funções quadráticas e dos resultados do algoritmo, tornando os conceitos matemáticos mais tangíveis e acessíveis para os alunos.

É crucial reconhecer que, enquanto a programação oferece uma ferramenta poderosa para expressão criativa e resolução de problemas, seu ensino deve ser abordado de maneira que promova a compreensão ética e crítica das tecnologias. Como mencionado na citação da BNCC, o objetivo não é apenas usar tecnologia, mas também compreender e questionar suas implicações em várias práticas sociais.

Para avançar, recomenda-se que futuras pesquisas explorem metodologias de ensino inovadoras que possam integrar ainda mais a computação no currículo escolar de maneira interdisciplinar. Isso incluiria a exploração de como a programação pode ser combinada com outras disciplinas, como ciências, artes e humanidades, para enriquecer a experiência de aprendizado e promover uma compreensão mais holística do mundo digital.

A programação no ensino básico não é apenas uma adição curricular, mas uma necessidade imperativa para preparar os alunos para as demandas de um futuro digital. Ao ensinar programação, estamos não apenas educando os futuros cientistas

da computação, mas estamos formando cidadãos capazes de entender e moldar a tecnologia de maneiras que beneficiem a sociedade como um todo.

Referências

1 ALVES, N. D. C. et al. **Ensino de computação de forma interdisciplinar em disciplinas de história no ensino fundamental: um estudo de caso**. Revista Brasileira de Informática na Educação. Florianópolis. 2016.

2 ARIMOTO,M., OLIVEIRA,W..**Dificuldades no Processode Aprendizagem de Programação de Computadores: um Survey com Estudantes de Cursos da Área de Computação.** In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI),27.,2019, Belém. Porto Alegre: Sociedade Brasileira de Computação,2019.p.244-254.

3 BRASIL, Ministério da Educação. Base Nacional Comum Curricular(BNCC), 2018. Disponível em: <http://basenacionalcomum.mec.gov.br/.Acessoem: 27demarÃğode2024>.

4 DEITEL, II.M.**Java, como programar H.M. Deitel e P.J. Deitel**; trad. Edson Furnankiewicz. - 3. ed. - Porto Alegre : Brookman, 2001.

5 IUSEM, Alfredo. Métodos de ponto proximal em otimização. IMPA, 1995.

6 KAFAI, Y. B., BURKE, Q. (2013). **Computer Programming Goes Back to School**. Phi Delta Kappan, 95(1), 61-65. <https://doi.org/10.1177/003172171309500111>

7 MALTEMPI, M. V., and VALENTE, J. A. . **Melhorando e Diversificando a Aprendizagem via Programação de Computadores.** International Conference on Engeneering and Computer Education–ICECE. 2000.

8 MUNIZ NETO, Antônio Caminha. **Tópicos de Matemática Elementar: números reais/Caminha Muniz Neto**. - 2.ed. – Rio de Janeiro: SBM, 2013. v.1; 235 p. (Coleção do Professor de Matemática; 24).

9 SALVETI, D. D.; BARBOSA, L. M. . **Algoritmos**. São Paulo: Pearson Makron books, 1998. 300 p. ISBN 853460715-x.

10 SECRETARIA EXECUTIVA DE DESENVOLVIMENTO DA EDUCAÇÃO. Softwares de Simulação Matemática: Material de apoio docente à ação, 2023. Disponível em: <https://portal.educacao.pe.gov.br/wp-content/uploads/2023/08/Softwares-de-Simulacao-Matematica.pdf>. Acesso em: 14 de maio de 2024.

11 SECRETARIA EXECUTIVA DE DESENVOLVIMENTO DA EDUCAÇÃO. Trilhas Completas: Tecnologias Digitais, 2024. Disponível em: <https://sites.google.com/adm.educacao.pe.gov.br/trilhascompletas/in%C3%ADcio/m%C3%A9dio/tecnologias-digitais?authuser=0>. Acesso em: 14 de maio de 2024.

12 SIERRA, K.; BATES, B.**Use a Cabeça! Java**. Rio de Janeiro: AltaBooks, 2010.

13 VASCONCELOS, R. C. da S., MORAES NETO, A. J. (2020). **A Computação no Currículo da Educação Básica**. REVISTA EIXO, 9(2), 68-76. <https://doi.org/10.19123/eixo.v9i2.851>

14 ZAHN, MAURÍCIO**Análise real**. São Paulo: Blucher, 2022. 456 p.: il. ISBN 978-65-5506-542-8