UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Faculdade de Ciências e Tecnologia
Câmpus de Presidente Prudente

Método de Otimização Determinística E Fractais Aplicado À Determinação De Múltiplos Pontos De Mínimo Em Problemas De Otimização Não Linear

Ernesto Lucanga Hombo

Orientador: Prof. Dr. Antonio Roberto Balbo
Co-Orientadora: Prof. Dra. Tatiana Miguel Rodrigues

Programa: Matemática Aplicada e Computacional

Presidente Prudente, Junho de 2023

UNIVERSIDADE ESTADUAL PAULISTA

Faculdade de Ciências e Tecnologia de Presidente Prudente

Programa de Pós-Graduação em Matemática Aplicada e Computacional

Método de Otimização Determinística E Fractais Aplicado À Determinação De Múltiplos Pontos De Mínimo Em Problemas De Otimização Não Linear

Ernesto Lucanga Hombo

Orientador: Prof. Dr. Antonio Roberto Balbo

Co-Orientadora: Prof. Dra. Tatiana Miguel Rodrigues

Dissertação de Mestrado Apresentada ao Programa de Pós-Graduação em Matemática Aplicada e Computacional da Faculdade de Ciências e Tecnologia da UNESP, Campus de Presidente Prudente.

Presidente Prudente, Junho de 2023

H764m

Hombo, Ernesto Lucanga

Método de Otimização Determinística E Fractais Aplicado À Determinação De Múltiplos Pontos De Mínimo Em Problemas De Otimização Não Linear / Ernesto Lucanga Hombo. -- , 2023

146 p.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências e Tecnologia, Presidente Prudente,

Orientador: Antonio Roberto Balbo

Coorientadora: Tatiana Miguel Rodrigues

1. Otimização Determinística. 2. Métodos de Gradientes. 3. Algoritmo de Otimização e Caos. 4. Problemas Multimodais. 5. Problema de Despacho Econômico. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências e Tecnologia, Presidente Prudente. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Presidente Prudente

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Método de Otimização Determinística e Fractais Aplicado à Determinação de Múltiplos Pontos de Mínimo em Problemas de Otimização Não Linear

AUTOR: ERNESTO LUCANGA HOMBO
ORIENTADOR: ANTONIO ROBERTO BALBO
COORIENTADORA: TATIANA MIGUEL RODRIGUES

Aprovado como parte das exigências para obtenção do Título de Mestre em Matemática Aplicada e Computacional, pela Comissão Examinadora:

Prof. Dr. ANTONIO ROBERTO BALBO (Participaçao Virtual)
Departamento de Matematica / Faculdade de Ciencias de Bauru

Prof. Dr. FABIANO BORGES DA SILVA (Participaçao Virtual)
Departamento de Matematica / Faculdade de Ciencias de Bauru

Prof. Dr. ANDRE CHRISTOVAO PIO MARTINS (Participaçao Virtual)
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Bauru

Presidente Prudente, 05 de julho de 2023

Dedico esta Dissertação
A minha esposa Angelina Hombo, aos meus filhos Toni, Mizael e Nila

Agradecimentos

Agradeço a Deus, pois Ele é onisciente e detentor de todo o conhecimento. Sua orientação e sabedoria foram fundamentais ao longo desta jornada.

Expresso minha profunda gratidão aos meus orientadores, o Professor Balbo e a Professora Tatiana, que sempre estiveram ao meu lado, oferecendo orientação, esclarecendo minhas dúvidas e me dando o apoio necessário para seguir adiante.

Agradeço a banca, pelas sugestões e recomendações dadas, que ajudaram muito para que o trabalho fosse concretizado.

Não posso deixar de agradecer minha esposa, Angelina Hombo, por estar sempre presente em minha vida, apoiando-me incondicionalmente, independentemente de eu estar em casa ou distante.

Também sou grato aos meus colegas da UNESP, que demonstraram um espírito de companheirismo e me proporcionaram um ambiente de aprendizado e colaboração.

Não posso esquecer de mencionar meus parentes e amigos, que sempre me encorajaram e me deram forças para avançar nesta trajetória acadêmica.

Por fim, gostaria de expressar minha gratidão a todos aqueles que, de forma direta e indireta, contribuíram e apoiaram-me ao longo dessa jornada, tornando possível a concretização deste trabalho. Seu apoio foi fundamental e sou imensamente grato por isso.

Resumo

Neste trabalho propõe-se uma abordagem determinística baseada no algoritmo de otimização e caos (AOC) e em métodos de gradientes de otimização via teoria de Julia, para a determinação de múltiplos pontos de ótimos locais em problemas de otimização multimodais com funções objetivo não lineares e não convexas. O método é testado em problemas específicos, como o problema de Despacho Econômico (PDE) com carregamento de pontos de válvula, onde a função objetivo, além das características de não linearidade e não convexidade, é não diferenciável nesses pontos. Para viabilizar a aplicação dos métodos mencionados é utilizada a função de suavização hiperbólica, que aproxima a função valor absoluto senoidal da função de custos do PDE, tornando-a diferenciável. O método é avaliado e, entre os múltiplos pontos de mínimo encontrados no PDE e em outro problema multimodal testado, são determinados o pior, o intermediário e o melhor ponto de mínimo que minimizam a função objetivo desses problemas. Esses resultados fornecem uma visão mais abrangente e precisa das soluções encontradas.

Palavras-Chave: *Otimização Determinística, Algoritmo de Otimização e Caos, Métodos de Gradientes, Problemas Multimodais, Problema de Despacho Econômico*.

Abstract

In this work a deterministic approach based on the Chaos Optimization Algorithm (COA) and in optimization gradient methods using Julia's set theory is proposed for determining multiple local optimal points in multimodal optimization problems with nonlinear and non-convex objective functions. The method is tested on specific problems, such as the Economic Dispatch Problem (EDP) with valve point loading, where the objective function, in addition to its nonlinearity and non-convexity characteristics, is non-differentiable at these points. To enable the application of the mentioned methods a hyperbolic smoothing function is used to approximate the sinusoidal absolute value function of the EDP cost function, making it differentiable. The method is evaluated, and among the multiple minimum points found in the EDP and another tested multimodal problem, the worst, intermediate, and best minimum points that minimize the objective function of these problems are determined. These results provide a more comprehensive and accurate view of the solutions found.

Keywords: *Deterministic Optimization, Optimization and Chaos Algorithm, Gradient Methods, Multimodal Problems, Economic Dispatch Problem..*

Sumário

Capítulo

1

Introdução

1.1 Contextualização

Na atualidade, a otimização desempenha um papel fundamental na resolução de problemas em diversas áreas de pesquisa e aplicação. A necessidade de encontrar soluções eficientes e de alto desempenho para problemas complexos tem impulsionado o desenvolvimento de técnicas e algoritmos de otimização em diversas áreas, como engenharia, economia, logística e ciência da computação. Um dos desafios que se enquadra nesse contexto é a otimização de problemas restritos e irrestritos não lineares, esses problemas surgem em uma ampla variedade de aplicações práticas, e sua resolução eficiente é essencial para o avanço tecnológico e econômico. Exemplos de áreas em que a otimização não linear é crucial incluem a engenharia de processos, a alocação de recursos em empresas, o planejamento de rotas logísticas e a modelagem de sistemas complexos. Para resolver problemas de otimização não lineares, existem diversas abordagens e vários métodos disponíveis, entre eles, destacam-se os métodos de gradiente, algoritmos genéticos, otimização por enxame de partículas e meta-heurísticas, entre outros. Cada um desses métodos possui características e capacidades distintas e a escolha do método mais adequado depende do problema em questão e dos requisitos específicos. No entanto, problemas não lineares podem apresentar características desafiadoras, como a existência de múltiplos pontos de mínimo, funções não diferenciáveis e restrições complexas, como no caso do problema do despacho econômico (PDE) [28], que é da área de engenharia elétrica, cujo objetivo é determinar qual deve ser o ponto de operação para a produção de energia elétrica em cada usina de modo a atender a demanda das cargas e minimizar os custos de produção. O PDE pode ser apresentado como:

$$\begin{aligned} \text{minimize} \quad & f(P_G) = \sum_{i=1}^{n}(a_i P_{G,i}^2 + b_i P_{G,i} + c_i) + \sum_{i=1}^{n} |e_i sen(f_i(P_{G,i}^{Min} - P_{G,i}))| \\ \text{sujeito a} \quad & \sum_{i=1}^{n} P_{G,i} = P_d \\ & P_{G,i}^{min} \leq P_{G,i} \leq P_{G,i}^{max} \\ & i = 1, ..., n \end{aligned} \tag{1.1}$$

cujos detalhes são apresentados no **Capítulo 6**. A função objetivo $f(P_G)$, definida em (1.1), é não linear, não convexa e não diferenciável, devido às expressões definidas por funções valores absolutos senoidais. Métodos tradicionais de otimização (gradiente des-

cendente, Newton, e outros métodos que usam gradiente) podem ficar presos em mínimos locais e não serem capazes de explorar eficientemente o espaço de busca, para resolver problemas com múltiplos pontos de mínimo, como no caso do PDE. Nesse sentido, técnicas mais avançadas têm sido investigadas para melhorar o desempenho da otimização em problemas não lineares, incluindo o uso de algoritmos baseados em fractais e caos.

Fractais e caos são fenômenos presentes em sistemas dinâmicos não lineares, o caos é caracterizado pela sensibilidade às condições iniciais e pela imprevisibilidade. Algoritmos de otimização baseados em Fractais e caos exploram essas propriedades para realizar buscas mais abrangentes no espaço de soluções, aumentando a chance de encontrar ótimos globais ou múltiplos pontos de mínimo. Esses algoritmos têm sido aplicados com sucesso em problemas complexos de otimização [30], demonstrando potencial para melhorar a eficiência e a qualidade das soluções encontradas. Dessa forma, a otimização de problemas restritos e irrestritos não lineares é uma área de pesquisa e aplicação que apresenta desafios e demanda por abordagens inovadoras. Este trabalho propõe investigar e desenvolver uma abordagem baseada em técnicas de fractais e caos para a otimização desses problemas, visando superar as limitações dos métodos tradicionais e alcançar melhores soluções.

1.2 Objetivos e Metodologia

Para o desenvolvimento do presente trabalho, a definição de objetivos e metodologia, são de capital importância.

1.2.1 Objetivos

O trabalho objetiva propor uma abordagem baseada em métodos de otimização determinística, associada a métodos de gradientes [28], [31] e de otimização e caos [30] via sistemas dinâmicos [9], [10], [13], [24], [15], cuja aplicação visa a determinação, dentre os múltiplos pontos de mínimo de problemas de otimização não linear, do melhor entre aqueles que o método determinar a partir de um ponto inicial ou seja, aquele que mais minimiza a função objetivo desses problemas entre os múltiplos pontos de mínimo encontrados. Para alcançar esse objetivo, os seguintes objetivos específicos foram definidos:

1. Explorar a literatura existente sobre otimização não linear e algoritmos baseados em fractais e caos: Foi realizada uma revisão abrangente da literatura científica para identificar os métodos, técnicas e algoritmos mais relevantes no campo da otimização não linear e da aplicação de fractais e caos nesse contexto. Isso proporcionou uma base sólida de conhecimento para embasar o desenvolvimento da abordagem proposta.

2. Propor uma abordagem de otimização baseada em técnicas de Fractais e caos: Com base na revisão da literatura, foi proposta uma abordagem inovadora que combina fractais e caos com técnicas de otimização não linear para lidar com problemas restritos e irrestritos. A abordagem proposta buscou explorar as propriedades caóticas para realizar buscas mais abrangentes e eficientes no espaço de soluções, visando superar as limitações dos métodos tradicionais.

3. Implementar e avaliar a abordagem proposta: A abordagem proposta foi implementada em MATLAB, permitindo sua execução e avaliação em problemas de otimização não linear reais. Foram utilizados conjuntos de dados e análises disponíveis nas referências [28] e [30]. para avaliar o desempenho e a eficiência da abordagem

foram comparados os resultados numéricos obtidos com os resultados das referências mencionadas.

1.2.2 Metodologia

A metodologia deste trabalho está dividida em etapas distintas, que são descritas a seguir:

1. Revisão bibliográfica: Foi realizada uma revisão detalhada da literatura científica relacionada à otimização não linear [5], [31] e aos algoritmos baseados em Fractais e caos [30]. Foram consultados artigos científicos, livros e outros materiais relevantes para identificar os conceitos-chave, os métodos existentes e as abordagens propostas na área. Essa revisão forneceu o embasamento teórico necessário para o desenvolvimento da abordagem proposta.

2. Formulação da abordagem baseada em fractais e caos: Com base na revisão bibliográfica, foi formulada a abordagem de otimização baseada em técnicas de fractais e caos. Foram identificados os principais elementos e etapas da abordagem, como a seleção de mapas caóticos, a definição de estratégias de busca e a integração com algoritmos de otimização existentes. A abordagem proposta foi detalhadamente descrita e fundamentada teoricamente.

3. Estratégias consideradas:

   - A utilização de métodos baseados em gradientes bem como considerar funções de suavização para o tratamento de funções que contêm valores absolutos.
   - Para a determinação dos pontos de mínimo utiliza-se como estratégia o algoritmo de otimização de caos (AOC) definido em [30], o qual é baseado na teoria de fractal (Conjunto de Julia) em sistemas dinâmicos. Este método é baseado no fato de que a dependência sensível da condição inicial de uma técnica de determinação de soluções, por exemplo a resolução do sistema Newton de direções de busca e da determinação de uma nova solução a partir de uma dada solução inicial, tem uma natureza fractal e os passos definidos nessas direções podem ser controlados por esse algoritmo.

4. Implementação e experimentação: A abordagem proposta foi implementada em MATLAB e Python, que são linguagens de programação adequadas para otimização não linear. Foram realizados experimentos utilizando conjuntos de dados disponíveis em [28] e [30].

5. Conclusões e trabalhos futuros: Com base na análise dos resultados, foram elaboradas conclusões finais sobre a eficácia da abordagem proposta. Foram destacados os principais resultados obtidos, as contribuições do trabalho para a área de otimização não linear e as possíveis direções para trabalhos futuros, como a extensão da abordagem para outros tipos de problemas ou a combinação com outras técnicas avançadas.

A metodologia proposta visa garantir um processo rigoroso e sistemático de pesquisa e desenvolvimento, permitindo a validação dos resultados obtidos e a obtenção de conclusões confiáveis sobre a eficácia da abordagem baseada em técnicas de fractais e caos para a otimização não linear

1.3 Contribuições Esperadas

Este trabalho visa fornecer as seguintes contribuições no campo da otimização não linear e aplicação de técnicas de fractais e caos:

1. Contribuição teórica: Uma contribuição teórica foi feita por meio da revisão abrangente da literatura existente sobre otimização não linear e algoritmos baseados em fractais e caos. Essa revisão permitirá a identificação e síntese dos principais conceitos, métodos e abordagens propostas nessa área, fornecendo uma visão abrangente do estado atual do conhecimento. Além disso, a formulação da abordagem baseada em técnicas de fractais e caos proposta neste trabalho contribuirá para o avanço teórico ao combinar fractais e caos com técnicas de otimização determinística para lidar com problemas não lineares.

2. Contribuição metodológica: Uma contribuição metodológica foi feita por meio do desenvolvimento e implementação da abordagem baseada em técnicas de fractais e caos para a otimização não linear. A metodologia proposta abrangerá a seleção de mapas caóticos, estratégias de busca e integração com algoritmos de otimização existentes. A implementação e experimentação da abordagem permitirão avaliar seu desempenho em problemas reais e compará-lo com métodos tradicionais de otimização, contribuindo para o desenvolvimento de novas ferramentas e técnicas para resolver problemas complexos.

3. Contribuição prática: Uma contribuição prática foi feita por meio da aplicação da abordagem proposta em problemas reais de otimização não linear. A avaliação da eficácia da abordagem em conjuntos de dados disponíveis na literatura ([28] e [30]) permitirá verificar sua capacidade de encontrar soluções de alta qualidade, em tempo razoável, e superar as limitações dos métodos tradicionais. Essa contribuição prática pode ter impacto direto na resolução de problemas complexos em áreas como engenharia, economia, ciências da computação e outros campos onde a otimização é essencial.

4. Contribuição para futuras pesquisas: Este trabalho também pode contribuir para futuras pesquisas no campo da otimização não linear e aplicação de técnicas de fractais e caos. As conclusões obtidas ao longo do estudo podem inspirar e direcionar trabalhos futuros, como a extensão da abordagem proposta para problemas específicos, a combinação com outras técnicas avançadas de otimização ou a investigação de novas aplicações da abordagem baseada em fractais e caos. Essas contribuições podem fornecer novas direções e oportunidades para avançar ainda mais o conhecimento nessa área.

Portanto, as contribuições esperadas deste trabalho abrangem tanto o avanço teórico quanto o desenvolvimento prático no campo da otimização não linear, oferecendo uma abordagem inovadora baseada em técnicas de fractais e caos, contribuindo para o aprimoramento das soluções de otimização em problemas complexos e desafiadores.

1.4 Organização do Trabalho

Este trabalho está organizado em seis capítulos, conclusão, referências e apêndices, com o objetivo de fornecer uma estrutura clara e lógica para a apresentação dos resultados obtidos. A seguir, descrevemos brevemente o conteúdo de cada capítulo.

O Capítulo 1 é a Introdução do trabalho em estudo, este capítulo apresenta uma visão geral do tema de pesquisa, destacando sua importância e relevância. Além disso, são apresentados os objetivos do trabalho, metodologia, a contextualização do problema e as contribuições esperadas.

O Capítulo 2 do trabalho aborda os espaços métricos e seus conceitos fundamentais. São explorados aspectos como distância, métrica, completude e compacidade. Para embasar teoricamente esses conceitos, foram utilizadas as referências [17], [25] e [26]. Ao longo do capítulo, essas referências foram utilizadas para fundamentar e aprofundar os tópicos abordados. Foram discutidos exemplos concretos e aplicações práticas dos espaços métricos, fornecendo uma base sólida para compreender e aplicar os conceitos apresentados em capítulos subsequentes.

O Capítulo 3 do trabalho aborda o tema da otimização não linear, que envolve a busca pela melhor solução em problemas onde as restrições e/ou a função objetivo são não lineares. Para embasar teoricamente esse tema, foram utilizadas as seguintes referências:

- [5]: Esta referência apresenta os fundamentos da programação não linear, incluindo técnicas de otimização e algoritmos para encontrar soluções ótimas. São discutidas diferentes abordagens, como métodos de ponto interior e métodos de gradiente, além de exemplos e aplicações.

- [27]: Nesta referência, o autor explora os principais conceitos da otimização não linear, abordando desde a formulação matemática do problema até a aplicação de algoritmos para encontrar soluções. São apresentados métodos clássicos, como o método de Newton e o método dos mínimos quadrados.

- [30]: Esta referência discute a aplicação de técnicas de otimização não linear em sistemas caóticos, nomeadamente o Algoritmo de otimização e caos (AOC), e o algoritmo de otimização e caos via Julia (AOC via Julia), também são exploradas propriedades interessantes desses sistemas.

- [31]: Nesta referência, o autor apresenta conceitos matemáticos relevantes para a otimização não linear, como cálculo multivariável e métodos de otimização. São discutidos tópicos como pontos críticos, gradiente e Hessiana, fornecendo uma base sólida para compreensão do assunto.

Ao longo do capítulo, essas referências foram utilizadas para fornecer embasamento teórico e prático sobre otimização não linear. Foram apresentados conceitos, algoritmos e exemplos de aplicação, permitindo compreender os desafios e abordagens utilizadas nessa área de estudo.

O Capítulo 4 do trabalho aborda o tema dos fractais e do caos, explorando suas propriedades e aplicações, onde o Conjunto de Julia é de capital importância. Para embasar teoricamente esse tema, foram utilizadas as seguintes referências:

- [4]: Nesta referência, o autor explora de forma abrangente os conceitos dos fractais e sua aplicação em diversos campos, como matemática, física, biologia e computação. São discutidos algoritmos de construção de fractais e propriedades interessantes dessas estruturas.

  [3]: Esta referência aborda especificamente a compressão de imagens utilizando técnicas fractais. O autor explora os fundamentos teóricos desse método e sua aplicação prática, proporcionando uma compreensão profunda dessa área.

- [8]: Nesta referência, o autor discute as aplicações dos fractais em várias áreas das ciências naturais e aplicadas. São abordados exemplos de fractais presentes na natureza, bem como sua modelagem e análise matemática.

- [10]: Nesta obra introdutória, o autor explora os conceitos do caos e sistemas dinâmicos caóticos. São discutidas as propriedades do comportamento caótico, como a sensibilidade às condições iniciais, e apresentados exemplos concretos de sistemas caóticos.

- [13]: Esta referência aborda os fundamentos matemáticos da geometria fractal, fornecendo uma base sólida para compreensão e análise dessas estruturas. São discutidos conceitos como dimensão fractal, conjuntos auto-similares e aplicações em áreas como a física e a biologia.

- [24]: Nesta referência, o autor apresenta uma introdução à dinâmica dos fractais, onde é apresentado o conjunto de Julia, explorando conceitos e propriedades desse conjunto.

Ao longo do capítulo, essas referências foram utilizadas para fornecer uma abordagem teórica e prática sobre os fractais e o caos. Foram apresentados conceitos fundamentais, exemplos de estruturas fractais, aplicações em diversas áreas do conhecimento em relação a fractais e caos.

O Capítulo 5 do trabalho explora os métodos de otimização baseados em fractais e caos. Esses métodos são inspirados nas propriedades das estruturas fractais e na natureza caótica de certos sistemas. Para embasar essa discussão, foram utilizadas as referências [13], [15], [24], [30] e [31], onde:

- [15]: Essa referência explora a aplicação de métodos caóticos e fractais na engenharia. Foram utilizados conceitos e técnicas descritos pelo autor, para compreender como essas abordagens podem ser adaptadas para a resolução de problemas de otimização.

- [30]: Nessa referência, são discutidos os fundamentos teóricos da dinâmica caótica de funções. São utilizados conceitos e técnicas dessa obra para compreender como a natureza caótica dos sistemas pode ser explorada em métodos de otimização, o autor descreve os Métodos de otimização baseados em fractais e caos via conjunto de Julia.

Ao longo do capítulo, essas referências foram utilizadas para discutir os métodos de otimização baseados em fractais e caos. São apresentadas as principais ideias por trás desses métodos, suas vantagens e desafios, além de exemplos práticos de sua aplicação em problemas de otimização.

No capítulo 6, são apresentados e discutidos os resultados numéricos obtidos a partir da aplicação dos métodos de otimização baseados em fractais e caos. São comparados os resultados obtidos no trabalho com os resultados das referências [28] e [30]. Além disso, são apresentados as trajetórias de pontos, tendo em conta os ciclos do algoritmo proposto no trabalho.

Na conclusão, são apresentadas as principais conclusões do trabalho, destacando as contribuições alcançadas, os resultados obtidos e as possíveis direções para pesquisas futuras. São ressaltadas as principais contribuições teóricas, metodológicas e práticas deste estudo. Estão listadas as referências bibliográficas utilizadas ao longo do trabalho,

por fim o Apêndice, onde são apresentados os códigos em MATLAB e python, usados para a execução de algoritmos e gráficos.

Dessa forma, concluímos o **Capítulo 1** apresentando uma introdução ao nosso trabalho. No próximo capítulo, abordaremos Espaços Métricos, explorando tópicos preliminares essenciais para o nosso estudo, como sequências, convergência, compacidade e completude em espaços de dimensão finita, entre outros assuntos relevantes.

Capítulo

2

Espaços Métricos

De acordo com [17], um espaço métrico é um conjunto X com uma métrica. A métrica associa qualquer par de elementos (pontos) de X à uma distância. Uma métrica é definida por axiomas, que definem uma série de propriedades ligadas a distância entre dois pontos na reta real $\mathbb{R}$ ou no plano complexo $\mathbb{C}$.

Neste capítulo são apresentados várias definições e conceitos que servirão de base para estudos posteriores, para o desenvolvimento do capítulo, as principais referências são [17] e [25].

2.1 Espaço Métrico

De acordo com [17], no cálculo são estudadas funções, denominadas como funções distância, que podem ser representadas por d, em que para cada par de elementos x, y do domínio de d podem ser associados a uma distância $d(x, y)$ dada por $d(x, y) = |x - y|$.

Definição 1. *Um espaço métrico* $(\mathbf{X}, d)$ *é um conjunto* $\mathbf{X}$ *com uma função* $d : \mathbf{X}\times\mathbf{X} \to \mathbb{R}$*, que mede a distância entre dois pontos* $x, y \in \mathbf{X}$*. Onde* d *satisfaz os seguintes axiomas:*

a) $0 < d(x, y) < \infty,\ \forall\ x, y \in \mathbf{X},\ x \neq y$*. Este axioma estabelece que* d *é um número real, finito e não negativo.*

b) $d(x, x) = 0,\ \forall\ x \in \mathbf{X}$*. Este axioma estabelece que a distância de um elemento para si mesmo é zero.*

c) $d(x, y) = d(y, x),\ \forall\ x, y \in \mathbf{X}$*. Este é o axioma da simetria, ou seja, a distância de* x *a* y *é igual a distância de* y *a* x

d) $d(x, y) \leq d(x, z) + d(z, y),\ \forall\ x, y, z \in \mathbf{X}$*. Este último axioma é conhecido como a desigualdade triangular.*

.

Exemplo 1. *(Métrica Euclidiana): Métrica que é muito usada, na qual, se considerarmos o espaço métrico* $(\mathbb{R}^n, d)$*, para algum* $n \in \mathbb{N}$*, então ela é definida da forma:*

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}\ ,$$

no qual $x, y \in \mathbb{R}^n$ *e* $x = (x_1, \ldots, x_i, \ldots, x_n)$*,* $y = (y_1, \ldots, y_i, \ldots, y_n)$*.*

Exemplo 2. *(Espaço de Funções C[a,b]): Dado um conjunto X, tomemos o conjunto de todas as funções $x, y, ...$ de variável real, em que cada uma das funções tem variável independente t, são definidas e contínuas num dado intervalo $J = [a, b]$. Assim sendo, a métrica escolhida é definida por:*

$d(x, y) = max|x(t) - y(t)|, t \in J$

Definição 2. *Duas métricas d_1 e d_2 num espaço $\mathbf{X}$ são equivalentes se existem constantes $0 < c_1 < c_2 < \infty$ de forma que*

$$c_1 d_1(x, y) \leq d_2(x, y) \leq c_2 d_1(x, y), \ \forall \ (x, y) \in \mathbf{X} \times \mathbf{X}.$$

Definição 3. *Dois espaços métricos $(\mathbf{X}_1, d_1)$ e $(\mathbf{X}_2, d_2)$ são equivalentes se existe uma função $h : \mathbf{X}_1 \to \mathbf{X}_2$ bijetora e, portanto, invertível, na qual a métrica $\tilde{d}_1$ em $\mathbf{X}_1$ definida por:*

$$\tilde{d}_1(x, y) = d_2(h(x), h(y)), \ \forall \ x, y \in \mathbf{X}_1$$

é equivalente a d_1.

2.2 Conjunto Aberto, Conjunto Fechado, Vizinhança

Nesta secção, veremos algumas definições importantes sobre conjuntos.

Definição 4. *(Bola e Esfera):*

Dado um ponto $x_0 \in (\mathbf{X}, d)$ e um número real $\epsilon > 0$, definiremos trêz tipos de conjuntos:

*(a) $B(x_0, \epsilon) = \{x \in \mathbf{X} | d(x, x_0) < \epsilon\}$ (**Bola Aberta**)*

*(b) $\bar{B}(x_0, \epsilon) = \{x \in \mathbf{X} | d(x, x_0) \leq \epsilon\}$ (**Bola Fechada**)*

*(c) $S(x_0, \epsilon) = \{x \in \mathbf{X} | d(x, x_0) = \epsilon\}$ (**Esfera**)*

Definição 5. *(Conjunto Aberto, Conjunto Fechado) Um subconjunto M de um espaço métrico X é aberto se contiver uma bola aberta em torno de cada um de seus pontos. Um subconjunto K é dito fechado se o seu complementar (em X) é aberto, Isto é, $K^c = X - K$*

Definição 6. *(Vizinhança): Uma bola aberta $B(x_0, \epsilon)$ de raio ϵ é frequentemente chamada de vizinhança de x_0.*

Definição 7. *(Ponto interior): Dizemos que x_0 é um ponto interior de $M \subset X$, se M é uma vizinhança de x_0. O interior de M é o conjunto de todos os pontos interiores de M e é denotado por M^0 ou int(M).*

Definição 8. *A função $f : \mathbf{X}_1 \to \mathbf{X}_2$ de um espaço métrico $(\mathbf{X}_1, d_1)$ para um espaço métrico $(\mathbf{X}_2, d_2)$ é contínua se, para cada $\epsilon > 0$ e $x \in \mathbf{X}_1$, existe um $\delta > 0$ tal que*

$$d_1(x, y) < \delta \ \Rightarrow \ d_2(f(x), f(y)) < \epsilon.$$

Se f é bijetora e então invertível e ainda, se sua inversa f^{-1} também for contínua, então, dizemos que f é "Homeomorfismo" entre $\mathbf{X}_1$ e $\mathbf{X}_2$. Nesse caso, dizemos que $\mathbf{X}_1$ e $\mathbf{X}_2$ são homeomorfos.

Dizer que dois espaços são equivalentes é mais forte que dizer que são homeomorfos. Para ser equivalente, precisa existir uma restrita relação entre ϵ e δ independente de x.

Definição 9. *(Ponto de acumulação): Seja M um subconjunto de um espaço métrico X. Então um ponto x_0 de X (que pode ou não ser um ponto de M) é chamado de ponto de acumulação de M (ou ponto limite de M) se cada vizinhança de x_0 contém pelo menos um ponto $y \in M$ distinto de x_0. O conjunto formado pelos pontos de M e os pontos de acumulação de M é chamado de fechamento de M e é denotado por $\bar{M}$*

Definição 10. *(Conjunto Denso): Um subconjunto M do espaço métrico X é dito denso em X se $\overline{M} = X$.*

Definição 11. *(Espaço Métrico Separável): O espaço métrico X é separável se tem um subconjunto enumerável que é denso em X.*

Exemplo 3. *Um exemplo de espaço métrico separável é o espaço métrico dos números reais, denotado por $\mathbb{R}$. O subconjunto enumerável denso em $\mathbb{R}$ é o conjunto dos números racionais $\mathbb{Q}$. Em outras palavras, o conjunto dos números racionais é denso em $\mathbb{R}$, o que significa que qualquer número real pode ser aproximado arbitrariamente por um número racional.*

2.3 Sequências de Cauchy

De acordo com [17], as sequências de números reais desempenham um papel muito importante no cálculo, e é a métrica em $\mathbb{R}$ que nos permite definir o conceito básico de convergência de tal sequência

2.3.1 Sequência convergente, Limite

Definição 12. *Uma sequência (x_n) num espaço métrico $X = (X, d)$ é convergente se existe um $x \in X$ tal que*

$$\lim_{n\to\infty} d(x_n, x) = 0$$

x é denominado de limite de (x_n) e podemos escrever

$$\lim_{n\to\infty} x_n = x$$

ou simplesmente

$$x_n \to x.$$

Dizemos que (x_n) converge para x ou tem limite x. Se (x_n) não é convergente, é dita divergente.

Lema 1. *Dado $X = (X, d)$ um espaço métrico, então:*
(a) *Uma sequência convergente em X é limitada e seu limite é único.*
(b) *Se $x_n \to x$ e $y_n \to y$ em X, então $d(x_n, y_n) \to d(x, y)$.*

Demonstração. **(a)** Vamos supor que $x_n \to x$. Então, tomando $\epsilon = 1$, podemos encontrar um N tal que $d(x_n, x) < 1$ para todo $n > N$. Assim sendo, pela desigualdade triangular (axioma d), para todo n nós teremos que $d(x_n, x) < 1 + a$, onde $a = max\{d(x_1, x), ..., d(x_N, x)\}$. Isso mostra que (x_n) é limitada. Assumindo que $x_n \to x$ e $x_n \to z$, nós obtemos $0 \leq d(x, z) \leq d(x, x_n) + d(x_n, z) \to 0 + 0$, logo $x = z$ o que prova que o limite é único.
(b) Pela desigualdade triangular, temos que

$$d(x_n, y_n) \leq d(x_n, x) + d(x, y) + d(y, y_n)$$

Assim obtemos

$$d(x_n, y_n) - d(x, y) \leq d(x_n, x) + d(y_n, y)$$

obteremos uma desigualdade similar trocando x_n e x assim como y_n e y e multiplicando por -1. Teremos que

$$|d(x_n, y_n) - d(x, y)| \leq d(x_n, x) + d(y_n, y) \to 0$$

Com $n \to \infty$ ■

2.3.2 Sequência de Cauchy

Definição 13. *Uma sequencia $\{x_n\}_{n=1}^{\infty}$ de pontos num espaço métrico $(\mathbf{X}, d)$ é chamada de "Sequência de Cauchy" se, dado qualquer número $\epsilon > 0$, existe um inteiro $N > 0$, tal que*

$$d(x_n, x_m) < \epsilon, \ \forall \ n, m > N.$$

Em outras palavras, na medida em que $n \to \infty$ mais próximos seus elementos estarão.

Teorema 1. *Se a sequência de pontos $\{x_n\}_{n=1}^{\infty}$ num espaço métrico $(\mathbf{X}, d)$ converge para um ponto $x \in \mathbf{X}$, então a sequência $\{x_n\}_{n=1}^{\infty}$ é de Cauchy.*

Demonstração. Como a sequência $\{x_n\}_{n=1}^{\infty}$ converge para um ponto $x \in \mathbf{X}$, então, para todo $\epsilon_0 > 0$, existe $n \geq N$ de forma que $d(x_n, x) < \epsilon_0$. Considere agora que para cada ϵ_0 existe um ϵ de forma que $\epsilon_0 = \frac{\epsilon}{2}$. Assim, $d(x_n, x) < \frac{\epsilon}{2}$. Suponha então $m, n \geq N$, logo, a partir da Definição 1 item (d):

$$d(x_n, x_m) \leq d(x_n, x) + d(x_m, x) < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon.$$

Portanto, $d(x_n, x_m) < \epsilon$, ou seja, $\{x_n\}_{n=1}^{\infty}$ é de Cauchy. ■

Definição 14. *Um espaço métrico é completo se toda sequência de Cauchy $\{x_n\}_{n=1}^{\infty}$ em $\mathbf{X}$ possui um limite $x \in \mathbf{X}$.*

Em outras palavras, o limite existe e pertence ao espaço em que a sequência está. Então, se $\{x_n\}_{n=1}^{\infty}$ é uma sequência de Cauchy de pontos em $\mathbf{X}$ e este espaço é completo, existe um ponto $x \in \mathbf{X}$ de forma que, para todo $\epsilon > 0$, a bola fechada $B(x, \epsilon)$ contêm x_n.

Exemplo 4. *O espaço métrico que apresentamos anteriormente, o $\mathbb{R}^n$ com a métrica Euclidiana é um exemplo de espaço métrico completo.*

Exemplo 5. *Um exemplo de espaço métrico que não é completo é o espaço métrico dos números racionais com a métrica usual $d(x, y) = |x - y|$. Este espaço métrico é denotado por Q e é um subconjunto de $\mathbb{R}$ (o conjunto dos números reais). Para ver que Q não é completo, considere a sequência de Cauchy $(3, 3.1, 3.14, 3.141, 3.1415, ...)$, que é uma sequência de números racionais que se aproxima de π (o número irracional). Embora essa sequência seja de Cauchy em Q, ela não converge para nenhum número racional, pois π não é um número racional. Portanto, a sequência de Cauchy não possui limite em Q e, portanto, Q não é completo. Isso significa que existem sequências em Q que são de Cauchy, mas não têm limite em Q, e, portanto, não é possível encontrar uma convergência dentro de Q para elas.*

O **Exemplo 5**, é um exemplo importante de espaço métrico incompleto, pois mostra que nem todos os espaços métricos são completos e que a completude é uma propriedade fundamental para muitos teoremas importantes na análise matemática.

2.4 Conjuntos Compactos

Definição 15. *Seja* $\mathbf{S} \subset \mathbf{X}$ *um subconjunto de um espaço métrico* $(\mathbf{X}, d)$. $\mathbf{S}$ *é compacto se toda sequência infinita* $\{x_n\}_{n=1}^{\infty}$ *contém uma subsequência convergente com limite em* $\mathbf{S}$.

Definição 16. *Seja* $\mathbf{S} \subset \mathbf{X}$ *um subconjunto de um espaço métrico* $(\mathbf{X}, d)$. $\mathbf{S}$ *é limitado se existe um ponto* $a \in \mathbf{X}$ *e um número* $R > 0$*, tal que*

$$d(a, x) < R, \ \forall \ x \in \mathbf{S}.$$

Definição 17. *Seja* $\mathbf{S} \subset \mathbf{X}$ *um subconjunto de um espaço métrico* $(\mathbf{X}, d)$. $\mathbf{S}$ *é totalmente limitado se, dado* $\epsilon > 0$ *existe um conjunto finito de pontos* $\{y_1, y_2, \ldots, y_n\} \subset \mathbf{S}$ *tal que, para todo* $x \in \mathbf{S}$*,* $d(x, y_i) < \epsilon$ *para algum* $y_i \in \{y_1, y_2, \ldots, y_n\}$.

Em outras palavras, um conjunto $\mathbf{S}$ ser totalmente limitado significa que podemos "cobri-lo" totalmente com um número finito de bolas abertas de raio r (quantas forem necessárias), de forma que a intersecção entre $\mathbf{S}$ e a união de todas as bolas abertas seja o próprio $\mathbf{S}$. Assim, o ponto que está no centro de cada uma delas, que podemos chamar de y_i, formam o conjunto ϵ-*net*.

A seguir apresentamos o "Princípio da casa dos pombos", que usaremos para demonstrar o Teorema 3.

Teorema 2 (Princípio da casa dos pombos)**.** *Se* $|A| > |B|$ *(o conjunto* A *tem mais elementos do que o conjunto* B*), então, para cada função* $f : A \to B$ *existe pelo menos dois elementos de* A *que são levados a um mesmo elemento de* B *pela* f.

Demonstração. Suponha, por contradição que nenhum elemento $b \in B$ é obtido por mais de um elemento $a \in A$. Como f é uma função, todos os elementos de A precisam de pelo menos uma imagem em B, então, para todos os elementos $a \in A$, existe $b \in B$, tal que $b = f(a)$, assim, ou A e B possuem o mesmo número de elementos ou B possui mais elementos que A, ou seja, $|A| = |B|$ ou $|A| < |B|$. Uma contradição, pois $|A| > |B|$.
Portanto, ao menos um elemento de B é imagem de pelo menos dois elementos de A. ■

Em outras palavras, se m pombos são colocados aleatoriamente em n caixas diferentes e $m > n$, então, certamente uma das caixa terá mais que um único pombo. Por ser um evento aleatório, pode ocorrer de todos os pombos estarem em uma única caixa, assim como podem estar distribuídos de forma semelhantes, mas, como o número de pombos é maior, uma das caixas terá mais que um pombo.

Teorema 3. *Sejam* $(\mathbf{X}, d)$ *um espaço métrico completo e um subconjunto* $\mathbf{S} \subset \mathbf{X}$*. Então* $\mathbf{S}$ *é compacto se e, somente se, é fechado e totalmente limitado.*

Demonstração. Suponha que $\mathbf{S}$ é fechado e totalmente limitado.
Seja $\{x_n \in \mathbf{S}\}$ uma sequência infinita qualquer de pontos em $\mathbf{S}$. Como $\mathbf{S}$ é totalmente limitado, podemos achar uma coleção finita de bolas abertas de raio $r = 1$ centradas em cada ponto $y_i \in \{y_1, y_2, \ldots, y_n\} \subset \mathbf{S}$, de forma que $\mathbf{S}$ é contido na união de todas elas. Como possuímos infinitos elementos x_n e apenas uma coleção finita de bolas abertas, temos mais pontos x_n do que bolas abertas, assim, pelo "Princípio da casa dos pombos" (Teorema 2), uma dessas bolas terá infinitamente muitos dos pontos x_n, vamos chama-la de B_1.
Agora, escolha N_1 para que $x_{N_1} \in B_1$. Então, para algum i, $B_1 = B_1(y_i, 1)$, ou seja, $d(x_{N_1}, y_i) < \epsilon$ (pois, S é totalmente limitado), mas $x_{N_1}, y_i \in B_1 \cap \mathbf{S}$, ou seja, este conjunto

também é totalmente limitado.
Assim, de forma análoga, podemos cobrir $B_1 \cap \mathbf{S}$ por um conjunto finito de bolas de raio $\frac{1}{2}$. Pelo "Princípio da casa dos pombos", uma das bolas conterá infinitamente muito dos pontos $x_n \in B_1 \cap \mathbf{S}$, seja ela nomeada de B_2. Escolha agora um $N_2 > N_1$, de forma que $x_{N_2} \in B_2$. Portanto, com um argumento semelhante ao já utilizado, afirmamos que $B_2 \cap (B_1 \cap \mathbf{S})$ é totalmente limitado.
De forma indutiva, continuamos esse processo, gerando uma sequência de conjuntos encaixados:

$$B_1 \supset B_2 \supset B_3 \supset B_4 \supset \cdots \supset B_n \supset \cdots$$

Para simplificar a notação, considere $B_n = B_n \cap B_{n-1} \cap \cdots \cap B_1 \cap \mathbf{S}$.
Assim, B_n tem raio $r = \dfrac{1}{2^{n-1}}$ e existe uma sequência de inteiros $\{N_n\}_{n=1}^{\infty}$ tal que $x_{N_n} \in B_n$, ou seja, podemos criar a sequência $\{x_{N_n}\}_{n=1}^{\infty}$. Pela forma como foi definido cada x_{N_n}, temos que ela é uma subsequência de $\{x_n\}$, e ainda, podemos afirmar que é uma sequência de Cauchy em $\mathbf{S}$, pois, visto que $x_{N_{n+1}}, x_{N_n} \in B_{N_n}$, temos que $d(x_{N_{n+1}}, x_{N_n}) < \dfrac{1}{2^{n-1}} < \epsilon$, para um certo ϵ.
Como $\mathbf{S}$ é fechado e $\{x_{N_n}\}$ é de Cauchy, então ela converge para um ponto $x \in \mathbf{S}$. Em outras palavras, dado uma sequência qualquer de $\mathbf{S}$, existe uma subsequência convergente com limite em $\mathbf{S}$. Portanto, $\mathbf{S}$ é compacto.

Suponha agora que $\mathbf{S}$ é compacto e, por contradição, que não exista um ϵ-*net* para $\mathbf{S}$.
Então, existe uma sequência infinita de pontos $\{x_n \in \mathbf{S}\}$ com $d(x_i, x_j) \geq \epsilon$, $\forall\ i \neq j$ e $\epsilon > 0$. Mas essa sequência deve possuir uma subsequência convergente $\{x_{N_i}\}$, pois $\mathbf{S}$ é compacto. Pelo Teorema 1 essa subsequência é de Cauchy e então, podemos achar um par de inteiros N_i, N_j com $N_i \neq N_j$ tal que $d(x_{N_i}, x_{N_j}) < \epsilon$. Mas $d(x_{N_i}, x_{N_j}) \geq \epsilon$, gerando uma contração. Então existe um ϵ-*net*, ou seja, $\mathbf{S}$ é totalmente limitado. ■

Exemplo 6. *Um exemplo clássico de um conjunto compacto é o intervalo fechado* $[0, 1]$ *na reta real. Este conjunto é limitado, ou seja, está contido em um intervalo de comprimento finito, e é fechado, ou seja, contém todos os seus pontos de acumulação. Para mostrar que* $[0, 1]$ *é compacto, vamos usar o Teorema 3, que nos dá a garantia de que um conjunto em* $\mathbb{R}$ *é compacto se, e somente se, ele é fechado e limitado. No caso do intervalo* $[0, 1]$*, podemos ver que ele é limitado, pois está contido no intervalo* $[-1, 2]$*. Além disso, podemos mostrar que é fechado, observando que ele contém todos os seus pontos de acumulação, ou seja, se uma sequência de números reais converge para um ponto fora de* $[0, 1]$*, então essa sequência não pode estar contida em* $[0, 1]$*. Assim,* $[0, 1]$ *é limitado e fechado, e portanto é compacto.*

O Exemplo 6 é muito importante, pois muitos teoremas na análise real e funcional dependem da compacidade de conjuntos para provar resultados importantes.

Exemplo 7. *O conjunto dos números reais* $\mathbb{R}$*, considerado com a métrica usual* $d(x, y) = |x - y|$*, não é compacto.*

Podemos ver que $\mathbb{R}$ *não é compacto observando que a sequência* (n) *de números inteiros não possui sub-sequência convergente dentro de* $\mathbb{R}$*. Ou seja, não podemos escolher uma subsequência finita de* (n) *que converja para um ponto em* $\mathbb{R}$*. Isso ocorre porque a distância entre dois números inteiros consecutivos é sempre 1, enquanto a distância entre qualquer número real e um número inteiro é sempre maior ou igual a* $\frac{1}{2}$*. Essa propriedade impede que qualquer subsequência de* (n) *possa convergir para um ponto em* $\mathbb{R}$*. Portanto, não há subsequência convergente de* (n) *em* $\mathbb{R}$*, e* $\mathbb{R}$ *não é compacto.*

2.5 Conjuntos Conexos

Definição 18. *Um espaço métrico $(\mathbf{X}, d)$ é conexo se os únicos dois subconjuntos de $\mathbf{X}$ que são simultaneamente abertos e fechados são os conjuntos $\mathbf{X}$ e $\varnothing$. Um subconjunto $\mathbf{S} \subset \mathbf{X}$ é conexo se o espaço métrico $(\mathbf{S}, d)$ é conexo. $\mathbf{S}$ é desconexo se ele não é conexo. $\mathbf{S}$ é totalmente desconexo se os únicos subconjuntos conexos não vazios de $\mathbf{S}$ são subconjuntos formados por apenas um ponto.*

Definição 19. *Seja $\mathbf{S} \subset \mathbf{X}$ um subconjunto de um espaço métrico $(\mathbf{X}, d)$. Então, $\mathbf{S}$ é conexo por caminho se, para todo par de pontos x e y em $\mathbf{S}$, existe uma função contínua $f : [0,1] \to \mathbf{S}$, do espaço métrico $([0,1], Euclidiana)$ em $(\mathbf{S}, d)$, de forma que $f(0) = x$ e $f(1) = y$. Tal função f é chamada de caminho de x a y em $\mathbf{S}$. $\mathbf{S}$ é desconexo por caminho se não é conexo por caminho.*

Também pode-se definir simplesmente conexo e multiplicadamente conexo. Seja $\mathbf{S}$ conexo por caminho. Um par de pontos $x, y \in \mathbf{S}$ é simplesmente conexo em $\mathbf{S}$ se, dado quaisquer dois caminhos f_0 e f_1 conectando $x, y \in \mathbf{S}$, podemos continuamente deformar f_0 em f_1 sem sair do subconjunto $\mathbf{S}$. O que significa que:

Dado pois pontos $x, y \in \mathbf{S}$ e dois caminhos $f_0, f_1 \in \mathbf{S}$ conectando x, y. em outras palavras, f_0, f_1 são duas funções contínuas indo de $[0,1]$ em $\mathbf{S}$ de forma que $f_0(0) = f_1(0) = x$ e $f_0(1) = f_1(1) = y$. Uma deformação contínua de f_0 e f_1 dentro de $\mathbf{S}$, significa uma função g de $[0,1] \times [0,1]$ para $\mathbf{S}$, então:

a) $g(s,0) = f_0(s)$, para $(0 \leq s \leq 1)$;

b) $g(s,1) = f_1(s)$, para $(0 \leq s \leq 1)$;

c) $g(0,t) = x$, para $(0 \leq t \leq 1)$;

d) $g(1,t) = y$, para $(0 \leq t \leq 1)$.

Então, dizemos que dois pontos x, y em $\mathbf{S}$ são simplesmente conexos em $\mathbf{S}$ se, dado dois caminhos f_0, f_1 indo de x para y em $\mathbf{S}$, existe uma função g que acabamos de definir.
Se x, y não são simplesmente conexos em $\mathbf{S}$, então dizemos que x, y são multiplicadamente conexos em $\mathbf{S}$.
$\mathbf{S}$ é chamado de simplesmente conexo se todo par de pontos x, y em $\mathbf{S}$ são simplesmente conexos, do contrário, é chamado de multiplicadamente conexo.

Exemplo 8. *Considere o conjunto $A = (x,y) \in \mathbb{R}^2 \mid x^2 + y^2 < 1$. Este conjunto é o interior da circunferência unitária no plano cartesiano, e é um exemplo de conjunto conexo. Para ver isso, podemos mostrar que, para qualquer dois pontos $p, q \in A$, existe um caminho contínuo dentro de A que liga p e q. Podemos construir esse caminho seguindo uma reta que liga p e q caso estes pontos sejam distintos. Caso contrário, $p = q$ e o caminho será reduzido a um ponto. Dessa forma de acordo com a Definição 19, o conjunto A é conexo, uma vez que qualquer par de pontos em A pode ser ligado por um caminho contínuo dentro de A.*

Exemplo 9. *Considere o conjunto $A = (x,y) \in \mathbb{R}^2 \mid x^2 + y^2 < 1 \cup (x,y) \in \mathbb{R}^2 \mid x^2 + (y-2)^2 < 1$. Este conjunto é a união de duas circunferências unitárias com centros separados verticalmente por uma distância de 2 unidades, e não é conexo. Para ver isso, podemos mostrar que existem dois conjuntos disjuntos não-vazios U e V em $\mathbb{R}^2$ que são abertos em A e cuja união é A. Podemos escolher U e V da seguinte forma:*
$U = (x,y) \in A \mid y < 1$ é o conjunto que contém apenas os pontos dentro da circunferência inferior. $V = (x,y) \in A \mid y > 1$ é o conjunto que contém apenas os pontos dentro da

circunferência superior. Esses conjuntos são disjuntos, não-vazios e abertos em A. Além disso, a união de U e V é igual a A. Portanto, A não é conexo, já que podemos separá-lo em dois conjuntos disjuntos não-vazios abertos.

O Exemplo 9, ilustra a importância da noção de conexidade em topologia. A conexidade é uma propriedade importante para a continuidade e a análise de funções em espaços topológicos, e muitas vezes é utilizada em provas e definições de conceitos fundamentais em matemática.

2.6 Contração em Espaços Métricos

Definição 20. *Uma transformação $f : \mathbf{X} \to \mathbf{X}$ num espaço métrico $(\mathbf{X}, d)$ é chamada de contração se existe uma constante $0 \leq s < 1$ de forma que:*

$$d(f(x), f(y)) \leq s \cdot d(x, y), \quad \forall\ x, y \in \mathbf{X}.$$

O número s é chamado de fator de contração de f.

Seria conveniente falar do maior e do menor número de um conjunto de números reais. Porém, um conjunto como $\mathbf{S} = (-\infty, 3)$ não possui nenhum dos dois. Para isso, segue a definição.

Definição 21. *Seja $\mathbf{S}$ um conjunto de números reais. Então, o ínfimo de $\mathbf{S}$ é $-\infty$ se contém números arbitrariamente maiores. Caso contrário, o ínfimo será $\max\{x \in \mathbb{R} : x \leq s \ \ \forall\ s \in \mathbf{S}\}$. O ínfimo de $\mathbf{S}$ sempre existe, por causa da natureza do sistema dos números reais, e é denotado por $\inf \mathbf{S}$. O supremo de $\mathbf{S}$ é similarmente definido, é igual a $+\infty$, caso necessário ou é igual a $\min\{x \in \mathbb{R} : x \geq s \ \ \forall\ s \in \mathbf{S}\}$ e é denotado por $\sup \mathbf{S}$.*

Teorema 4 (Teorema da Contração)**.** *Seja $f : \mathbf{X} \to \mathbf{X}$ uma contração num espaço métrico completo $(\mathbf{X}, d)$. Então f possui exatamente um ponto fixo $x_f \in \mathbf{X}$ e ainda, para cada ponto $x \in \mathbf{X}$, a sequência $\{f^n(x) : \ n = 1, 2, \dots\}$ converge para x_f. Ou seja:*

$$\lim_{n\to\infty} f^n(x) = x_f, \quad \forall\ x \in \mathbf{X}.$$

Demonstração. Seja a sequência $\{x_n\} = \{x_0,\ x_1 = f(x_0),\ x_2 = f(x_1),\ \dots, x_{n+1} = f(x_n),\ \dots\}$, que é equivalente a $\{f^n(x)\}$, basta observar que $f^n(x_0) = f^{n-1}(f(x_0)) = f^{n-1}(x_1) = \cdots = f(x_{n-1})$. Queremos provar que ela é convergente.
Como f é uma contração, então, existe uma constante $0 \leq s < 1$ de forma que $d(f(x), f(y)) \leq s \cdot d(x, y)$, assim:

$$\begin{aligned} d(x_{n+1}, x_n) = d(f(x_n), f(x_{n-1})) &\leq sd(x_n, x_{n-1}) = sd(f(x_{n-1}), f(x_{n-2})) \\ &\leq s^2 d(x_{n-1}, x_{n-2}) = s^2 d(f(x_{n-2}), f(x_{n-3})) \\ &\vdots \\ &\leq s^n d(x_1, x_0). \end{aligned}$$

Ou seja, $d(x_{n+1}, x_n) \leq s^n d(x_1, x_0)$.
Agora, para $p \geq 0$:

$$\begin{aligned} d(x_n, x_{n+p}) \leq d(x_n, x_{n+1}) + d(x_{n+1}, x_{n+2}) + \cdots + \\ + d(x_{n+p-2}, x_{n+p-1}) + d(x_{n+p-1}, x_{n+p}). \end{aligned}$$

Continuando o raciocínio e utilizando o fato de que dada uma sequência finita da forma (com $q \neq 1$) $\{a_n\} = \{a_1,\ a_1q,\ a_1q^2,\ \ldots,\ a_1q^n\}$, a soma de todos os seus termos é dada pela fórmula $\sum_{i=0}^{n} a_1q^i = \frac{a_1q^n - a_1}{q-1}$, deste modo, temos:

$$\begin{aligned} d(x_n, x_{n+p}) &= \sum_{i=0}^{p-1} d(x_{n+i}, x_{n+i+1}) \\ &\leq \sum_{i=0}^{p-1} s^{n+i} \cdot d(x_1, x_0) \\ &\leq \frac{s^n}{1-s} \cdot d(x_1, x_0). \end{aligned}$$

Como $0 \leq s < 1$, então:

$$\begin{aligned} &\lim_{n\to\infty} d(x_n, x_{n+p}) \leq \lim_{n\to\infty} \frac{s^n}{1-s} \cdot d(x_1, x_0) && \Leftrightarrow \\ &\lim_{n\to\infty} d(x_n, x_{n+p}) \leq 0 \cdot d(x_1, x_0) && \Leftrightarrow \\ &\lim_{n\to\infty} d(x_n, x_{n+p}) = 0 \end{aligned}$$

Portanto, $\{x_n\}$ e, por consequência, $\{f^n(x)\}$ são sequências de Cauchy em $\mathbf{X}$, que, por ser um espaço métrico completo, garante que $\lim f^n(x) = x_f \in \mathbf{X}$.
Observe que x_f é ponto fixo de f, pois, como f é uma contração ela também é contínua, logo:

$$f(x_f) = f\left(\lim_{n\to\infty} f^n(x)\right) = \lim_{n\to\infty} f \circ f^n(x) = \lim_{n\to\infty} f^{n+1}(x) = x_f.$$

Para finalizar, resta provar que esse ponto x_f é único.
Suponha por contradição que exista um outro ponto $y_f \neq x_f$ que também seja ponto fixo de f. Assim,

$$\begin{aligned} &d(x_f, y_f) = d\left(f(x_f), f(y_f)\right) \leq sd(x_f, y_f) && \Leftrightarrow \\ &d(x_f, y_f) \leq sd(x_f, y_f) && \Leftrightarrow \\ &(1-s)d(x_f, y_f) \leq 0. \end{aligned}$$

Como $s < 1$, então $(1-s) > 0$, logo $d(x_f, y_f) \leq 0$, o que só acontece quando $x_f = y_f$, uma contradição, portanto, existe um único ponto fixo. ■

Exemplo 10. *Seja (X, d) um espaço métrico e considere a função $f : X \to X$ dada por $f(x) = \frac{x}{2}$, para todo $x \in X$. A função f é uma contração em (X, d).*

De fato, para todo $x, y \in X$, temos:

$$d(f(x), f(y)) = d(\frac{x}{2}, \frac{y}{2}) \ = \frac{1}{2}d(x, y) \ \leq \frac{1}{2}d(x, y)$$

Portanto, $d(f(x), f(y)) \leq \frac{1}{2}d(x, y)$, para todo $x, y \in X$, o que significa que f é uma contração em (X, d) com constante de contração $k = \frac{1}{2}$.

A importância de funções de contração em espaços métricos se dá pela existência do Teorema do Ponto Fixo de Banach (Teorema 4), que afirma que toda função de contração

em um espaço métrico completo tem um único ponto fixo. Esse resultado é útil em diversas áreas da matemática, como na análise funcional e na teoria de equações diferenciais [17].

Dessa forma, concluímos o capítulo sobre Espaços Métricos, onde o estudo dos Espaços Métricos revelou-se fundamental para o desenvolvimento de nosso trabalho. Através da análise de sequências, convergência, compacidade e completude em espaços de dimensão finita, obtivemos uma base sólida para explorar e compreender conceitos mais avançados nas áreas subsequentes.

No próximo capítulo, abordaremos a Otimização não Linear, explorando tanto problemas restritos quanto irrestritos, além de discutirmos as condições necessárias e suficientes de otimalidade. Também examinaremos métodos aproximativos e abordaremos a limitação dos métodos baseados em gradientes.

Capítulo

3

Otimização Não Linear

Neste capítulo, abordaremos a temática da otimização não linear, com foco nos estudos de problemas irrestritos e restritos. Serão exploradas as condições fundamentais de otimalidade, bem como diversos métodos baseados em gradientes, tanto para problemas irrestritos quanto para problemas restritos. Além disso, discutiremos as limitações inerentes aos métodos de gradientes. As principais referências para este capítulo são [5], [27], [30] e [31].

3.1 Problemas Irrestritos

Definição 22. *Consideremos uma função $f : \mathbb{R}^n \to \mathbb{R}$ um problema irrestrito é do tipo:*

$$\min_{x \in X} f(x) \tag{3.1}$$

Onde $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$ e $X \subseteq \mathbb{R}^n$ é um conjunto aberto.

Definição 23. *Seja o problema irrestrito definido pela equação (3.1), e seja $\bar{x} \in X \subset \mathbb{R}^n$.*
(a) Se $f(\bar{x}) \leq f(x)$ para todo $x \in X$, diz-se que $\bar{x}$ é um minimizante global.
(b) Se existe uma vizinhança de raio ϵ, dada por $N_\epsilon(\bar{x})$ em torno da qual temos:
$f(\bar{x}) \leq f(x)$ para todo $x \in N_\epsilon(\bar{x})$ diz-se que $\bar{x}$ é um minimizante local

3.1.1 Condições Necessárias de Otimalidade

Para esta seção e o nosso estudo de forma geral, os conceitos de diferenciabilidade são de capital importância, daí as seguintes definições:

Definição 24. *(Diferenciabilidade de Primeira Ordem): Seja $f : X \to \mathbb{R}$, onde $X \subset \mathbb{R}^n$ é um aberto. Dizemos que f é diferenciável em $\bar{x} \in X$ se existir um vetor gradiente $\bigtriangledown f(\bar{x})$ tal que para cada $x \in X$ vale:*
$f(x) = f(\bar{x}) + \bigtriangledown f(\bar{x})^T(x - \bar{x}) + ||x - \bar{x}||\alpha(\bar{x}, x - \bar{x})$
Com $\lim_{||y|| \to 0} \alpha(\bar{x}, y) = 0$. A função f é diferenciável em X se f for diferenciável para todo o $\bar{x} \in X$

Da definição anterior, temos que o vetor gradiente é o vetor das derivadas parciais, e é denotado por:

$$\bigtriangledown f(\bar{x}) = \left(\frac{\partial f(\bar{x})}{\partial x_1}, ..., \frac{\partial f(\bar{x})}{\partial x_n} \right) \tag{3.2}$$

Definição 25. *(Derivada Direcional): A derivada direcional de f em $\bar{x}$ na direcão d é dada por*

$$\lim_{\lambda \to 0} \frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda} = \bigtriangledown f(\bar{x})^T d \tag{3.3}$$

Definição 26. *Uma função f diz-se duas vezes diferenciável em $\bar{x} \in X$ se existir um vetor $\bigtriangledown f(\bar{x})$ e uma matriz simétrica $H(\bar{x})$ pertencente ao $\mathbb{R}^{n \times n}$ (Hessiana de f em $\bar{x}$) tal que para cada $x \in X$ temos que:*

$f(x) = f(\bar{x}) + \bigtriangledown f(\bar{x})^T (x - \bar{x}) + \frac{1}{2}(x - \bar{x})^T H(\bar{x})(x - \bar{x}) + ||x - \bar{x}||^2 \alpha(\bar{x}, x - \bar{x})$

Com $\lim_{||y|| \to 0} \alpha(\bar{x}, y) = 0.$

A função f é duas vezes diferenciável em X se f for duas vezes diferenciável para todo o $\bar{x} \in X$.

A matriz Hessiana é a matriz das derivadas parciais de segunda ordem:

$$H(\bar{x})_{ij} = \frac{\partial^2 f(\bar{x})}{\partial x_i \partial x_j}, i, j \in \{1, ..., n\} \tag{3.4}$$

Um conceito muito importante a se ter em conta é o de direção de descida como veremos a seguir.

Definição 27. *(Direção de descida): A direção $\bar{d}$ diz-se uma direção de descida de f em $\bar{x}$ se*

$$f(\bar{x} + \epsilon \bar{d}) < f(\bar{x})$$

Para todo $\epsilon > 0$ suficientemente pequeno.

O teorema a seguir, nos permite identificar uma direção de descida de f em $\bar{x}$ tendo em conta o gradiente da f em $\bar{x}$.

Teorema 5. *Suponhamos que f é diferenciável em $\bar{x}$. Se existe um vetor d tal que $\bigtriangledown f(\bar{x})^T d < 0$, então $f(\bar{x} + \lambda d) < f(\bar{x})$ para todo $\lambda > 0$ suficientemente pequeno, isto é, d é uma direção descendente de f em $\bar{x}$.*

Demonstração. Temos que

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \lambda \bigtriangledown f(\bar{x})^T d + \lambda ||d|| \alpha(\bar{x}, \lambda d), \tag{3.5}$$

Onde $\alpha(\bar{x}, \lambda d) \to 0$ quando $\lambda \to 0$. A igualdade (3.5) pode ser escrita na forma:

$$\frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda} = \bigtriangledown f(\bar{x})^T d + ||d|| \alpha(\bar{x}, \lambda d)$$

.

Como $\bigtriangledown f(\bar{x})^T d < 0$ e $\alpha(\bar{x}, \lambda d) \to 0$ a medida que $\lambda \to 0$, concluimos que $f(\bar{x} + \lambda d) - f(\bar{x}) < 0$ para todo $\lambda > 0$ suficientemente pequeno. ■

Como colorário deste teorema, em seguida enunciaremos uma condição necessária de primeira ordem, conhecida como teorema de Fermat.

Teorema 6. *(Condição Necessária de Primeira Ordem): Suponhamos que f é diferenciável em $\bar{x}$. Se $\bar{x}$ é um minimizante local, então $\bigtriangledown f(\bar{x}) = 0$.*

Demonstração. Se $\bigtriangledown f(\bar{x}) \neq 0$, então resulta do **Teorema 5** que $d = -\bigtriangledown f(\bar{x}) = 0$ é uma direção de descida. Logo, pela definição de direção de descida, $\bar{x}$ não é um minimizante local. ■

Definição 28. *(Ponto Crítico): Seja $f : X \to \mathbb{R}$ uma função definida num conjunto $X \subseteq \mathbb{R}^n$. Suponhamos que $\bar{x} \in int(X)$ e que f é diferenciável numa vizinhanca de $\bar{x}$. Então $\bar{x}$ é chamado de ponto crítico de f se $\nabla f(\bar{x}) = 0$.*

O teorema a seguir estabelece uma condição necessária de segunda ordem.

Teorema 7. *(Condição Necessária de Segunda Ordem): Suponhamos que f é duas vezes diferenciável em $\bar{x} \in X$. Se $\bar{x}$ é minimizante local, então $\bigtriangledown f(\bar{x}) = 0$ e $H(\bar{x})$ é semidefinida positiva.*

Demonstração. Da condição necessária de primeira ordem, sabemos que $\bar{x}$ ser minimizante local implica $\bigtriangledown f(\bar{x}) = 0$. Falta demonstrar que se $\bar{x}$ for minimizante local, então $H(\bar{x}) \geq 0$ (isto é, $H(\bar{x})$) é semidefinida positiva). Fazemos a demonstração por contradição: Vamos mostrar que se $H(\bar{x})$ não for semidefinida positiva, então $\bar{x}$ não pode ser minimizante local. Com efeito, se $H(\bar{x})$ não é semidefinida positiva, então existe um vetor d tal que $d^T H(\bar{x})d < 0$. Logo,

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \lambda \bigtriangledown f(\bar{x})^T d + \frac{1}{2}\lambda^2 d^T H(\bar{x})d + \lambda^2 ||d||^2 \alpha(\bar{x}, \lambda d)$$

$f(\bar{x}+\lambda d) = f(\bar{x})+\frac{1}{2}\lambda^2 d^T H(\bar{x})d+\lambda^2||d||^2\alpha(\bar{x}, \lambda d)$ onde $\alpha(\bar{x}, \lambda d) \to 0$ à medida que $\lambda \to 0$.

De modo equivalente,

$$\frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda^2} = \frac{1}{2} d^T H(\bar{x})d + ||d||^2 \alpha(\bar{x}, \lambda d).$$

Uma vez que $d^T H(\bar{x})d < 0$ e $\alpha(\bar{x}, \lambda d) \to 0$ quando $\lambda \to 0$, concluímos que $f(\bar{x} + \lambda d) - f(\bar{x}) < 0$ para todo $\lambda > 0$ suficientemente pequeno, isto é, $\bar{x}$ não é minimizante local. ■

Se $\bigtriangledown f(\bar{x}) = 0$ e a $H(\bar{x})$ é semidefinida positiva, não podemos ter a certeza se $\bar{x}$ é um minimizante local. Os teoremas de condição necessária, como descrito, não são suficientes, para tal, enunciaremos a seguir o teorema que estabelece condições suficientes de otimalidade.

Teorema 8. *(Condições Suficientes): Suponhamos que f é duas vezes diferenciável em $\bar{x} \in X$. Se $\bigtriangledown f(\bar{x}) = 0$ e $H(\bar{x})$é definida positiva, então $\bar{x}$ é um minimizante local estrito*

Demonstração. Ver em [31] ■

3.1.2 Métodos Aproximativos

Para a presente seção, veremos alguns métodos aproximativos para resolver problemas de otimização, nomeadamente o método do gradiente descendente, o método de Newton e o método de Levenberg-Marquardt. A presente seção tem como fontes [31] e [27].

3.1.2.1 Método de Descida Mais Rápida ou Gradiente Descendente

Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função de classe C^1. Consideremos o problema irrestrito:

$$\min_{x \in \mathbb{R}^n} f(x) \tag{3.6}$$

O método do gradiente descendente, trata de encontrar a partir de um dado ponto $x^0 \in \mathbb{R}^n$ (aproximação inicial) o minimizante local de $f(x)$ mais próximo de x^0. Como o

gradiente num ponto aponta na direção de maior crescimento da função nesse ponto, o método do gradiente descendente procura em cada ponto caminhar na direção oposta do gradiente nesse ponto. A cada iteração k, o método do Gradiente Descendente atualiza a aproximação x^k de acordo com a seguinte regra:

$$x^{k+1} = x^k + \alpha_k d^k \tag{3.7}$$

.

Onde:

1. d^k é a direção de busca no ponto x^k, dada por $d^k = -\nabla f(x^k)$, ou seja, a direção oposta ao gradiente da função no ponto x^k. Essa escolha garante que estejamos caminhando na direção de descida mais rápida em cada iteração.

2. α_k é o tamanho do passo ou taxa de aprendizagem na iteração k. É um valor positivo que controla o tamanho do avanço que fazemos a cada iteração. É importante escolher um valor adequado para α_k, pois um valor muito pequeno pode tornar a convergência lenta, enquanto um valor muito grande pode causar oscilações ou até mesmo a divergência do método, e é determinado resolvendo o problema:

$$\min_{\alpha \in \mathbb{R}} \psi_k(\alpha) = f(x^k + \alpha d^k). \tag{3.8}$$

O método é descrito abaixo pelo **Algoritmo 1**:

Algoritmo 1: Método do Gradiente Descendente

Entrada: Função f, aproximação inicial $x^0 \in \mathbb{R}^n$, tolerância $\epsilon > 0$
Saída : Aproximação x^k com $||\nabla f(x^k)|| \leq \epsilon$
1. Inicialização: $k := 0$.
2. Direção de procura: $d^k := -\nabla f(x^k)$.
3. Critério de parada: Se $\|d^k\| \leq \epsilon$, então pare; caso contrário, continue.
4. Procura unidimensional do tamanho do passo: Resolver o problema
$\min_{\alpha \in \mathbb{R}} \psi_k(\alpha) = f(x^k + \alpha d^k)$.
Seja α_k uma aproximação do minimizante desse problema.
5. Nova aproximação: Fazer
$x^{k+1} := x^k + \alpha_k d^k$
$k := k + 1$
e voltar a 2.

Neste algoritmo, segue a explicação de cada passo:

1. Inicialização: Neste passo, definimos a iteração inicial como $k = 0$, indicando que estamos começando a partir da aproximação inicial x^0.

2. Direção de procura: Calculamos a direção de procura d^k como o oposto do gradiente da função no ponto atual x^k. Isso ocorre porque o gradiente aponta na direção de maior crescimento da função, e queremos caminhar na direção oposta para encontrar o mínimo.

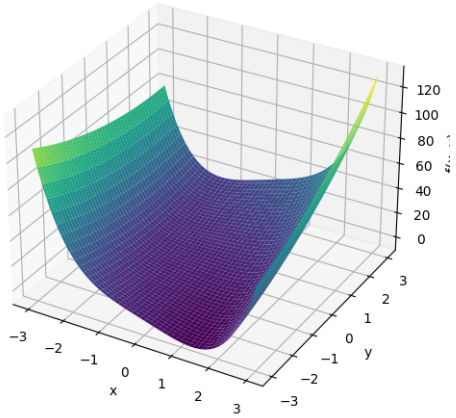
3. Critério de parada: Verificamos se a norma da direção de procura $|d^k|$ é menor ou igual à tolerância ϵ. Se for, significa que o gradiente é quase nulo e estamos próximos o suficiente do mínimo desejado e podemos parar o algoritmo. Caso contrário, continuamos para o próximo passo.

4. Procura unidimensional do tamanho do passo: Neste passo, realizamos uma busca unidimensional para encontrar o tamanho de passo α_k que minimiza a função $\psi_k(\alpha) = f(x^k + \alpha d^k)$. Existem várias estratégias para encontrar esse valor.

5. Nova aproximação: Atualizamos a aproximação atual x^k somando o tamanho de passo α_k multiplicado pela direção de procura d^k. Isso nos dá a nova aproximação x^{k+1}. Em seguida, incrementamos k em 1 para passar para a próxima iteração. Voltamos ao passo 2 para calcular uma nova direção de procura e repetimos o processo até que o critério de parada seja satisfeito.

Exemplo 11. *Neste exemplo, vamos efetuar as duas primeiras iterações do método do gradiente descendente na minimização da função* $f(x, y) = x^4 + 3xy + (y + 2)^2$, *partindo de* $x^0 = (-2, 3)$ *e tolerância* $\epsilon = 0.1$. *A Figura 3.1, ilustra o gráfico da função.*

Figura 3.1: Gráfico da função



Fonte: Autor

Para efetuar as duas primeiras iterações do método do gradiente descendente na minimização da função $f(x, y) = x^4 + 3xy + (y + 2)^2$, *partindo de* $x^0 = (-2; 3)$ *e tolerância* $\epsilon = 0.1$, *podemos seguir os passos do método da seguinte maneira:*

1. *Iteração 1:*

   $x^0 = (-2; 3)$;

   $\nabla f(-2; 3) = \big(-23\ ; 16\big)$;

   $d^0 = -\nabla f(-2; 3) = \big(23\ ; -16\big)$;

   $\alpha^0 = 0.1$ *(tamanho do passo fixo);*

   $(x^1, y^1) = (-2; 3) + 0.1\,\big(23\ ; -16\big) = (-0.7; 1.4)$;

   $\nabla f(-0,7; 1,4) = \big(-1.732\ ; 8.6\big)$;

   $\|\nabla f(-0,7; 1,4)\| = \sqrt{(-1.732)^2 + 8.6^2} \approx 8.772675 > 0.1$

2. *Iteração 2:*

   $(x^1, y^1) = (-0.7; 1.4)$;

   $\nabla f(-0.7; 1.4) = \big(-1.732\ ; 8.6\big)$;

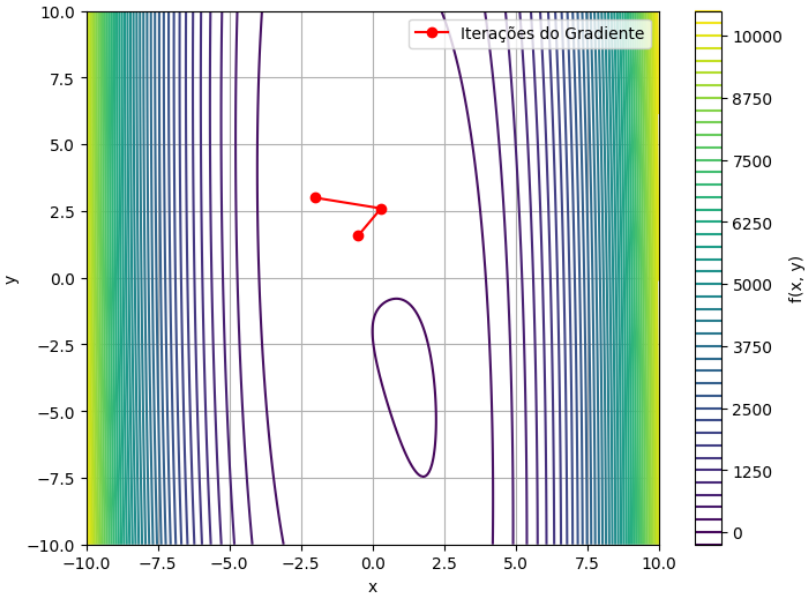
   $d^1 = -\nabla f(-0.7; 1.4) = \big(1.732\ ; -8.6\big)$

   $\alpha^1 = 0.1$ *(tamanho do passo fixo);*

$$(x^2, y^2) = (-0.7, 1.4) + 0.1\left(1.732\ ; -8.6\right) = (-0.5268; -0.46);$$
$$\nabla f(-0.5268; -0.46) = \left(-0.763\ ; 4.868\right);$$
$$\|\nabla f(-0.5268; -0.46)\| = \sqrt{(-0.763)^2 + (4.868)^2} \approx 4.93 > 0.1$$

Nas duas primeiras iterações, obtemos as aproximações $x^1 = (-0.7; 1.4)$ *e* $x^2 = (-0.5268; -0.46)$, *com valores de gradiente ainda maiores que a tolerância* $\epsilon = 0.1$. *Isso indica que a função está descendo e que estamos nos aproximando de um possível mínimo local. A Figura 3.2, apresenta as curvas de nível da função* $f(x, y)$ *onde são apresentadas as iterações do método do gradiente descendente. As setas vermelhas indicam a direção das iterações. O processo de otimização começa em* $x^0 = (-2, 3)$ *e avança em direção ao mínimo local da função.*

Figura 3.2: Curvas de Nível da Função



Fonte: Autor

Portanto, de acordo com [31], o método do gradiente descendente é bastante simples em termos computacionais e, devido a sua simplicidade, é bastante utilizado. Necessita de pouca informação, exigindo apenas as derivadas de primeira ordem para o cálculo do gradiente. A sua principal desvantagem é que apresenta uma convergência lenta.

3.1.2.2 Método de Newton

Suponhamos que queremos resolver o problema

$$\min_{x \in \mathbb{R}^n} f(x) \tag{3.9}$$

Para x numa vizinhança de $\bar{x}$, o valor $f(x)$ pode ser aproximado através da expansão quadrática de Taylor:

$$f(x) \approx h(x) := f(\bar{x}) + \bigtriangledown f(\bar{x})^T(x - \bar{x}) + \frac{1}{2}(x - \bar{x})^T H(\bar{x})(x - \bar{x}). \tag{3.10}$$

Notar que h é uma função quadrática, cujo minimizante local é encontrado resolvendo o sistema $\bigtriangledown h(x) = 0$. Uma vez que o gradiente de h é dado por

$$\bigtriangledown h(x) = \bigtriangledown f(\bar{x}) + H(\bar{x})(x - \bar{x}), \tag{3.11}$$

somos motivados a resolver o sistema

$$\bigtriangledown f(\bar{x}) + H(\bar{x})(x - \bar{x}) = 0. \tag{3.12}$$

Assumindo que $H(\bar{x})$ é não singular obtemos da equação anterior que

$$x - \bar{x} = -H(\bar{x})^{-1} \bigtriangledown f(\bar{x}) \Leftrightarrow x = \bar{x} - H(\bar{x})^{-1} \bigtriangledown f(\bar{x}).$$

Portanto, no método de Newton, a cada iteração, atualizamos a aproximação atual x^k usando a fórmula

$$x^{k+1} = x^k - \alpha_k H(\bar{x})^{-1} \bigtriangledown f(x^k), \tag{3.13}$$

onde $H(\bar{x})$ é a matriz Hessiana de f avaliada em $\bar{x}$, $\bigtriangledown f(x^k)$ é o gradiente de f avaliado em x^k e α_k é o tamanho do passo, que é o minimizante da função

$$\psi_k(\alpha) = f(x^k + \alpha d^k) \tag{3.14}$$

Isto nos conduz ao seguinte **Algoritmo 2**, para determinar uma aproximação da solução do problema.

Algoritmo 2: Método de Newton

Entrada: Função f, aproximação inicial $x^0 \in \mathbb{R}^n$, tolerância $\epsilon > 0$
Saída : Aproximação x^k com $||\nabla f(x^k)|| \leq \epsilon$
1. **Inicialização**: $k := 0$
2. **Critério de parada**: Se $||\nabla f(x^k)|| \leq \epsilon$, então pare; senão, continue.
3. **Direção de procura**: $d^k := -H(x^k)^{-1}\nabla f(x^k)$.
4. **Tamanho do passo**: $\alpha_k := \min \psi_k(\alpha) = f(x^k + \alpha d^k)$.
5. **Nova aproximação**: Fazer
   $x^{k+1} := x^k + \alpha_k d^k$
   $k := k + 1$
   e voltar ao passo 2.

Assim, eis a explicação detalhada do **Algoritmo 2**:

1. Inicialização:
   - Neste passo, definimos a variável k como zero, que será utilizada para contar o número de iterações.
2. Critério de parada:
   - Verificamos se a norma do gradiente $\nabla f(x^k)$ é menor ou igual à tolerância ϵ.
   - Se a condição for satisfeita, o algoritmo para, pois encontramos uma aproximação da solução com a precisão desejada.
   - Caso contrário, continuamos para o próximo passo.
3. Direção de procura:
   - Calculamos a direção de procura d^k utilizando a fórmula $d^k = -H(x^k)^{-1}\nabla f(x^k)$.
   - Aqui, $H(x^k)$ é a matriz Hessiana de f avaliada no ponto x^k.

4. Tamanho do passo:

   - Determinamos o tamanho de passo α_k que minimiza a função $\psi_k(\alpha) = f(x^k + \alpha d^k)$.
   - Isso pode ser feito por meio de uma busca unidimensional ou através de outros métodos, como a busca em linha exata ou a busca em linha inexacta.
   - A escolha do método de determinação do tamanho de passo depende da natureza do problema e das características da função a ser otimizada.

5. Nova aproximação:

   - Atualizamos a aproximação atual x^k usando a fórmula $x^{k+1} = x^k + \alpha_k d^k$.
   - Incrementamos o contador k em 1 para indicar que uma nova iteração foi realizada.
   - Em seguida, voltamos ao passo 2 para verificar o critério de parada e continuar o processo de iteração.

O algoritmo continua iterando até que o critério de parada seja satisfeito, ou seja, a norma do gradiente seja menor ou igual à tolerância desejada.

3.1.2.3 Método de Levenberg-Marquardt

Para este algoritmo, vamos começar considerando um problema de ajuste de curvas em que temos um conjunto de dados observados (x_i, y_i), onde x_i é a variável independente e y_i é o valor observado correspondente. Suponha que tenhamos um modelo matemático parametrizado $f(x;\theta)$, onde θ são os parâmetros do modelo. Consideremos a função $R(x;\theta)$ é definida como:

$$R(x;\theta) = y - f(x;\theta), \tag{3.15}$$

onde y é o valor observado e $f(x;\theta)$ é o valor previsto pelo modelo para uma dada entrada x e parâmetros θ.

Definição 29. *A função $R(x;\theta)$ definida pela equação (3.15) recebe o nome de função residual, ou seja, é uma função que representa a diferença entre os valores observados e os valores previstos pelo modelo matemático em um problema de ajuste de curvas. Em um contexto mais geral, a função residual mede o erro ou a discrepância entre os dados medidos e os valores calculados pelo modelo.*

Exemplo 12. *Vejamos um exemplo numérico para ilustrar o conceito. Considere um problema simples de ajuste de curvas, onde temos os seguintes dados observados:*

$$(x_1, y_1) = (1, 3), (x_2, y_2) = (2, 5), (x_3, y_3) = (3, 7).$$

Suponha que desejamos ajustar uma reta ao conjunto de dados, cujo modelo é dado por $f(x;\theta) = \theta_1 x + \theta_2$, onde θ_1 e θ_2 são os parâmetros da reta. A função residual seria definida como:

$$R(x;\theta) = y - f(x;\theta).$$

Por exemplo, para o primeiro ponto (x_1, y_1), a função residual seria:

$$R(x_1;\theta) = y_1 - f(x_1;\theta) = 3 - (\theta_1 \cdot 1 + \theta_2) \rightarrow R(x_1;\theta) = \theta_2 - \theta_1 + 3.$$

De maneira similar, podemos calcular as funções residuais para os outros pontos.

Para abordagem do método de Levenberg-Marquardt, seja $R : \mathbb{R}^n \to \mathbb{R}^m$ uma função residual, vamos considerar o problema

$$\min_{x \in \mathbb{R}^n} \|R(x)\| \tag{3.16}$$

Dado um ponto inicial x_0, o método de Levenberg-Marquardt aplicado ao problema (3.16) visa encontrar, a cada iteração k, uma direção $d(\lambda_k) \in \mathbb{R}^n$ que seja a solução do sistema linear

$$(J(x_k)^T J(x_k) + \lambda_k I)d(\lambda_k) = -J(x_k)^T R(x_k) \tag{3.17}$$

onde $J(x) \in \mathbb{R}^{m \times n}$ é a matriz Jacobiana de $R(x)$ e $\lambda \in \mathbb{R}_+$ é uma constante denominada parâmetro de Levenberg-Marquardt ou parâmetro de damping. A direção de Levenberg-Marquardt $d(\lambda)$ também é denominada por d^{LM}. Este método, vem para resolver problemas em que a matriz $J(x_k)^T J(x_k)$ não é definida positiva. Se $\lambda_k > 0$, em cada iteração k o sistema linear (3.17) tem solução e esta é única. Consequentemente o método está bem definido.

Teorema 9. *Sejam $A \in \mathbb{R}^{m \times n}$ e $\lambda \in \mathbb{R}_+$. A matriz $A^T A + \lambda I$ é definida positiva.*

Demonstração. Como

$$v^T A^T A v = (Av)^T (Av) = ||Av||_2^2 \geq 0$$

para todo vetor $v \in \mathbb{R}^n$, $A^T A$ é semidefinida positiva. Seja $\lambda \in \mathbb{R}_+^*$ fixado. Assím, temos que

$$v^T (A^T A + \lambda I)v = v^T A^T A v + \lambda v^T v = ||Av||^2 + \lambda ||v||^2 > 0$$

para todo vetor não-nulo $v \in \mathbb{R}^n$. Portanto, a matriz $A^T A + \lambda I$ é definida positiva. ■

De acordo com [22], o método de Levenberg-Marquardt é amplamente utilizado em problemas de ajuste de curvas não lineares, reconstrução 3D, processamento de imagem e muitas outras aplicações onde é necessário encontrar parâmetros ótimos para um modelo não linear que se ajuste aos dados observados. Assim, segue o **Algoritmo 3**.

Algoritmo 3: Algoritmo de Levenberg-Marquardt

Entrada: Ponto inicial x_0, valor inicial para λ_0, tolerância ϵ
Saída : Resultado

i- Defina um ponto inicial x_0, um valor inicial para o parâmetro de damping λ_0 e uma tolerância ϵ.

ii- Calcule o residual $R(x_0)$ e a matriz Jacobiana $J(x_0)$ de $R(x)$.

iii- Enquanto $||R(x_k)|| > \epsilon$ ou $||\Delta x|| > \epsilon$ faça:
Calcule a direção de Levenberg-Marquardt d^{LM} resolvendo o sistema linear $(J(x_k)^T J(x_k) + \lambda_k I)d^{LM} = -J(x_k)^T R(x_k)$.
Calcule o novo ponto $x_{k+1} = x_k + d^{LM}$.
se $||R(x_{k+1})|| < ||R(x_k)||$ **então**
| Atualize o valor do parâmetro de damping $\lambda_{k+1} = \frac{1}{2}\lambda_k$.
fim
senão
| Mantenha o valor do parâmetro de damping $\lambda_{k+1} = 2\lambda_k$.
fim

iv- Atualize $k \leftarrow k + 1$ e retorne ao passo ii.

Do **Algoritmo 3**, temos que:

1. Definição da entrada e saída: O algoritmo recebe como entrada o ponto inicial x_0, o valor inicial do parâmetro de damping λ_0 e a tolerância ϵ. A saída é o resultado final, que é o vetor de parâmetros otimizado.

2. Cálculo do residual e da matriz Jacobiana: Nesse passo, é calculado o residual $R(x_0)$ e a matriz Jacobiana $J(x_0)$ da função residual $R(x)$ no ponto inicial x_0.

3. Iterações principais: O algoritmo entra em um loop enquanto a norma do residual $||R(x_k)||$ for maior do que a tolerância ϵ ou enquanto a norma da atualização dos parâmetros $||\Delta x||$ for maior do que a tolerância ϵ.

    - Cálculo da direção de Levenberg-Marquardt: Nesse passo, é resolvido o sistema linear $(J(x_k)^T J(x_k) + \lambda_k I)d^{LM} = -J(x_k)^T R(x_k)$ para obter a direção de Levenberg-Marquardt d^{LM}.
    - Atualização do ponto: O novo ponto x_{k+1} é calculado somando-se a direção de Levenberg-Marquardt ao ponto atual x_k.
    - Atualização do parâmetro de damping: Se a norma do residual no novo ponto $||R(x_{k+1})||$ for menor do que a norma do residual no ponto atual $||R(x_k)||$, isso indica uma melhoria na função objetivo. Nesse caso, o valor do parâmetro de damping λ_{k+1} é reduzido pela metade, tornando as atualizações dos parâmetros mais agressivas. Caso contrário, o valor do parâmetro de damping λ_{k+1} é dobrado, tornando as atualizações mais conservadoras.

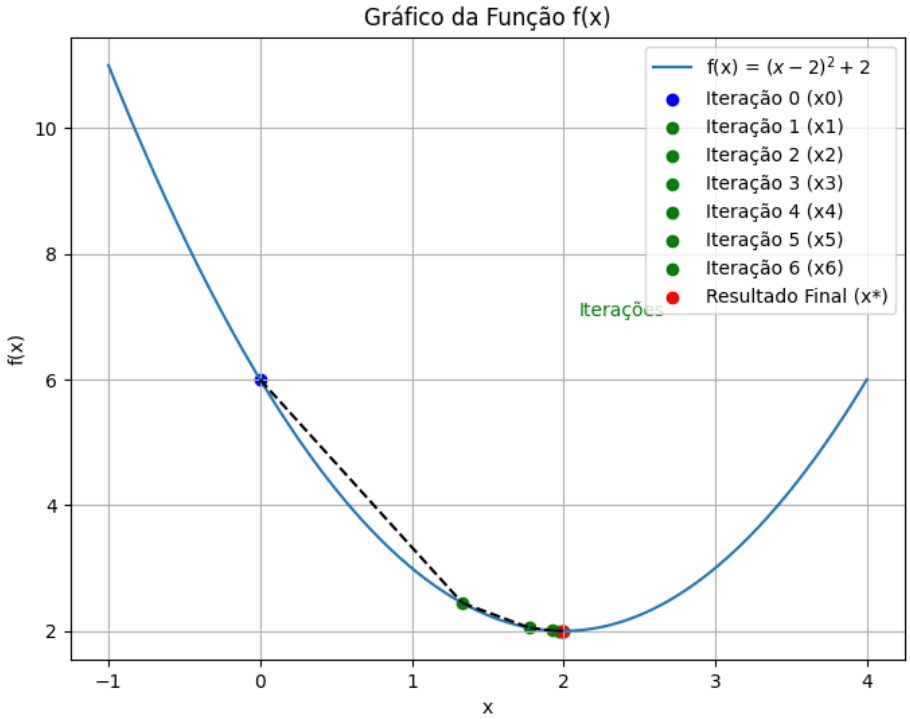
4. Atualização do contador e retorno ao passo 2: O contador k é incrementado e o algoritmo retorna ao passo 2 para a próxima iteração. Esse processo é repetido até que a condição de parada seja satisfeita.

Exemplo 13. *Neste exemplo, vamos usar o algoritmo de Levenberg-Marquardt para minimizar a função* $f(x) = (x-2)^2+2$. *A função residual pode ser definida como* $R(x) = x-2$, *visto que, para cada valor de* x, *o resíduo é calculado como a diferença entre* x *e* 2. *Por exemplo, se* $x = 0$, *então o resíduo é* $R(0) = 0 - 2 = -2$, *se* $x = 1$, *então* $R(1) = 1 - 2 = -1$, *e assim por diante. Assim sendo, o problema se torna minimizar a norma do resíduo* $\|R(x)\|^2$. *Para aplicar o algoritmo de Levenberg-Marquardt, precisamos calcular a matriz Jacobiana de* $R(x)$, *que é dada por* $J(x) = 1$. *Agora, podemos aplicar o algoritmo. Suponha que escolhemos o ponto inicial* $x^0 = 0$, *o valor inicial para o parâmetro de damping* $\lambda_0 = 1$ *e a tolerância* $\epsilon = 0.001$. *Aplicando o algoritmo de Levenberg-Marquardt por algumas iterações, obtemos os seguintes resultados:*

- ***Iteração 0:*** $x = 0 \rightarrow \|R(0)\| = 2 > \epsilon$
- ***Iteração 1:*** $x = 1.33333333 \rightarrow \|R(1.33333333)\| = 0.66666667 > \epsilon$
- ***Iteração 2:*** $x = 1.77777778 \rightarrow \|R(1.77777778)\| = 0.22222222 > \epsilon$
- ***Iteração 3:*** $x = 1.92592593 \rightarrow \|R(1.92592593)\| = 0.07407407 > \epsilon$
- ***Iteração 4:*** $x = 1.97530864 \rightarrow \|R(1.97530864)\| = 0.02469136 > \epsilon$
- ***Iteração 5:*** $x = 1.99176955 \rightarrow \|R(1.99176955)\| = 0.00823045 > \epsilon$
- ***Iteração 6:*** $x = 1.99725652 \rightarrow \|R(1.99725652)\| = 0.00274348 > \epsilon$

Podemos ver que mesmo não satisfazendo o critério de parada, o método converge rapidamente para a solução ótima $x^* = 2$, que é o minimizante global da função $f(x)$. A Figura 3.3 apresenta o ponto de mínimo final, que é destacado em vermelho, e as iterações anteriores são representadas em verde.

Figura 3.3: Representação gráfica da função e pontos da iteração



Fonte: Autor

3.2 Problemas Restritos

Vamos considerar o problema de minimizar $f(x)$ sujeito as condições impostas como descritas a seguir:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{sujeito a} \quad & g(x) \leq 0 \\ & h(x) = 0 \\ & x \in X \end{aligned} \tag{3.18}$$

onde $X \subset \mathbb{R}^n$ é um conjunto aberto, $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, isto é, $g(x) = (g_1(x), ..., g_m(x))$ e $h : \mathbb{R}^n \to \mathbb{R}^l$, isto é, $h(x) = (h_1(x), ..., h_l(x))$, $l \in \mathbb{N}_0$ (se $l = 0$, então o problema não tem restrição de igualdade).

Definição 30. *(Problema Restrito). O problema apresentado pela equação (3.18) recebe o nome de Problema restrito ou com restrição.*

Definição 31. *(Região de Factibilidade ou Conjunto Admissível): Consideremos um conjunto A do problema dado pela equação (3.18). O conjunto A é admissível se $A = \{x \in X : g(x) \leq 0, h(x) = 0\}$. Assim, a equação (problema) 3.18 pode ser escrita como:*

$$\min_{x \in A} f(x).$$

Uma questão muito importante a se ter em conta, é sobre a dimensão, de acordo com [31] só faz sentido considerar o problema (3.18) com $m+1 \leq n$, com efeito se tivermos mais restrições do que grau de liberdade, então em geral a região admissível é vazia e o problema não tem solução.

Definição 32. *Consideremos $g : \mathbb{R}^n \to \mathbb{R}^m$ diferenciável, onde $g(x) = (g_1(x), ..., g_m(x))$ e $x \in \mathbb{R}^n$, a matriz Jacobiana de g será denotada por:*

$$\bigtriangledown g(x) = \begin{pmatrix} \bigtriangledown g_1(x)^T \\ \vdots \\ \bigtriangledown g_m(x)^T \end{pmatrix}, \tag{3.19}$$

onde cada $\nabla g_i(x)^T$ representa o gradiente (vetor linha) da função $g_i(x)$, para $i = 1, 2, ..., m$.

Da **Definição (32)**, resulta que:

$$\nabla g(x) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1}(x) & \frac{\partial g_1}{\partial x_2}(x) & \cdots & \frac{\partial g_1}{\partial x_n}(x) \\ \frac{\partial g_2}{\partial x_1}(x) & \frac{\partial g_2}{\partial x_2}(x) & \cdots & \frac{\partial g_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1}(x) & \frac{\partial g_m}{\partial x_2}(x) & \cdots & \frac{\partial g_m}{\partial x_n}(x) \end{pmatrix}.$$

Se considerarmos $m = 1$, teremos o gradiente da g como:

$$\nabla^T g(x) = \begin{pmatrix} \frac{\partial g}{\partial x_1}(x) \\ \frac{\partial g}{\partial x_2}(x) \\ \vdots \\ \frac{\partial g}{\partial x_n}(x) \end{pmatrix}, \tag{3.20}$$

onde cada $\frac{\partial g}{\partial x_i}(x)$ representa a derivada parcial de $g(x)$ em relação a x_i, para $i = 1, 2, ..., n$.

Definição 33. *Consideremos $h : \mathbb{R}^n \to \mathbb{R}^l$ diferenciável, ou seja, $h(x) = (h_1(x), ..., h_l(x)$, onde $x \in \mathbb{R}^n$, a matriz Jacobiana de h é dada por $\bigtriangledown h(x) = \begin{pmatrix} \bigtriangledown h_1(x)^T \\ \vdots \\ \bigtriangledown h_l(x)^T \end{pmatrix}$.*

Se expandirmos $\bigtriangledown h(x)$ e consideramos $l = 1$, o resultado é análogo ao caso da **Definição 32**, ou seja, termos o gradiente de $h(x)$.

3.2.1 Convexidade em Otimização

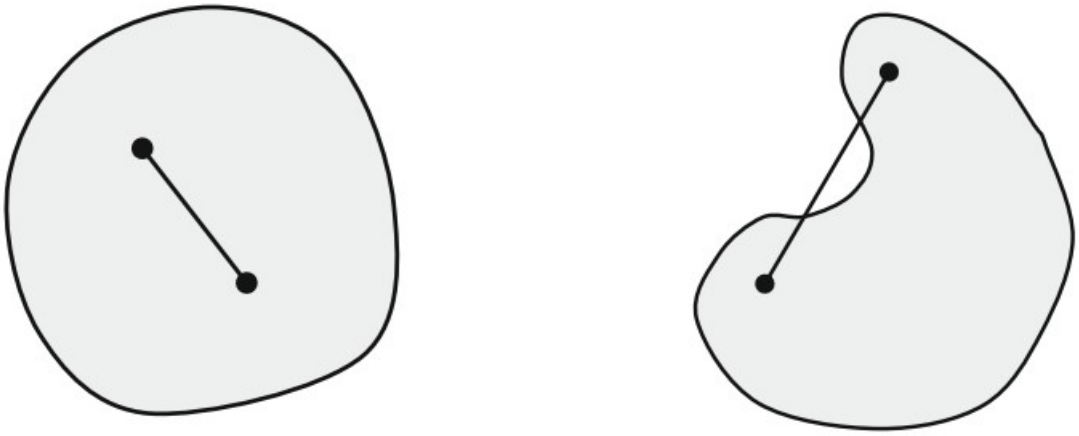
De acordo com [31], a convexidade (ou concavidade) desempenha um papel fundamental na resolução de problemas de otimização, sejam eles restritos ou irrestritos. Nesse sentido, é importante explorar conceitos gerais relacionados à convexidade, os quais englobam conjuntos, funções e problemas de otimização. Essas propriedades são essenciais, pois permitem identificar potenciais soluções ótimas para os problemas em questão. A seguir, apresentaremos diversos conceitos e definições relevantes acerca da convexidade.

Definição 34. *Sejam $x, y \in \mathbb{R}^n$. Os pontos na forma $\lambda x + (1-\lambda)y$ para $\lambda \in [0, 1]$ são chamados de combinações convexas de x e y.*

Definição 35. *(Conjunto Convexo): Um conjunto $S \subset \mathbb{R}^n$ diz-se convexo se para todo $\lambda \in [0,1]$ se tem $\lambda x + (1-\lambda)y \in S$. O que significa que qualquer dois pontos distintos de S passa uma reta contida em S.*

A Figura 3.4 ilustra-nos um exemplo de conjunto convexo, onde a esquerda temos um conjunto convexo e a direita um não convexo.

Figura 3.4: Conjunto Convexo (esquerda) e não convexo (direita)



Fonte: [11]

Exemplo 14. *Considere o conjunto $X = \{(x_1, x_2) \in \mathbb{R}^2 : x_2 > 0\}$. Sejam $x = (x_1, x_2)$ e $y = (y_1, y_2)$ elementos de X, o que implica que $x_2 > 0$ e $y_2 > 0$. Agora, vamos verificar se a combinação convexa $\lambda x + (1-\lambda)y = (\lambda x_1 + (1-\lambda)y_1, \lambda x_2 + (1-\lambda)y_2)$ pertence a X para qualquer valor de λ no intervalo $[0,1]$. Observamos que a segunda coordenada da combinação convexa é $\lambda x_2 + (1-\lambda)y_2$. Dado que $x_2 > 0$ e $y_2 > 0$, podemos afirmar que $\lambda x_2 + (1-\lambda)y_2 > 0$ para qualquer valor de λ no intervalo $[0,1]$. Portanto, a combinação convexa está contida em X. Concluímos, então, que o conjunto X é convexo, uma vez que qualquer combinação convexa de pontos em X também pertence a X.*

Proposição 1. *Um conjunto $X \subset \mathbb{R}^n$ é convexo se, e somente se, para $k \geq 2$, vale:*
$\sum_{i=1}^{k} \lambda_i x^i \in X$, $\forall\ \lambda_i \in [0,1]$ *e* $\sum_{i=1}^{k} \lambda_i = 1$, *o que significa que o conjunto X contém todas as combinações convexas de pontos de X.*

Demonstração.

($\Rightarrow$) Primeiro, vamos provar que se X é convexo, então contém todas as combinações convexas de pontos em X. Para fazer isso, vamos supor que X é convexo e mostrar que qualquer combinação convexa de pontos em X também está em X. Seja $x^1, x^2, \ldots, x^k \subset X$ e $\lambda_1, \lambda_2, \ldots, \lambda_k$ uma coleção de coeficientes não negativos que somam 1. Então, a combinação convexa de pontos em X é definida como

$$\sum_{i=1}^{k} \lambda_i x^i \tag{3.21}$$

Para mostrar que a expressão (3.21) está em X, precisamos mostrar que $\sum_{i=1}^{k} \lambda_i x^i$ satisfaz a definição de convexidade de X, ou seja, que para quaisquer dois pontos $x, y \in X$ e

$\alpha \in [0,1]$, temos que $\alpha x + (1-\alpha)y \in X$. Podemos fazer isso usando a definição de combinação convexa:

$$\alpha x + (1-\alpha)y = \sum_{i=1}^{2} \lambda_i x^i, \text{ onde } \lambda_1 = \alpha,\ \lambda_2 = 1-\alpha,\ x^1 = x,\ x^2 = y \tag{3.22}$$

Como X é convexo, temos que $\alpha x + (1-\alpha)y \in X$. Como isso é verdadeiro para qualquer $x, y \in X$ e $\alpha \in [0,1]$, concluímos que $\sum_{i=1}^{k} \lambda_i x^i \in X$.

($\Leftarrow$) Vamos provar que se X contém todas as combinações convexas de pontos em X, então X é convexo. Para fazer isso, vamos supor que X contém todas as combinações convexas de pontos em X e mostrar que X satisfaz a definição de convexidade. Sejam $x, y \in X$ e $\alpha \in [0,1]$, precisamos mostrar que $\alpha x + (1-\alpha)y \in X$. Podemos fazer isso usando a definição de combinação convexa:

$$\alpha x + (1-\alpha)y = \sum_{i=1}^{2} \lambda_i x^i, \text{ onde } \lambda_1 = \alpha,\ \lambda_2 = 1-\alpha,\ x^1 = x,\ x^2 = y, \tag{3.23}$$

como X contém todas as combinações convexas de pontos em X, temos que $\sum_{i=1}^{2} \lambda_i x^i \in X$. Portanto, $\alpha x + (1-\alpha)y \in X$, o que mostra que X é convexo. Concluímos que X é convexo se e somente se contém todas as combinações convexas de pontos em X.

■

Proposição 2. *Seja $I \subset \mathbb{N}$ um conjunto de índices (finito ou infinito), $X_i, i \in I$ conjuntos convexos. Então o conjunto $X = \bigcap_{i \in I} X_i$ é convexo.*

Demonstração. Para provar que a interseção $\bigcap_{i \in I} X_i$ é convexa, precisamos mostrar que, para quaisquer dois pontos x, y na interseção e $\alpha \in [0,1]$, o ponto $\alpha x + (1-\alpha)y$ também está na interseção. Primeiro, note que, como x e y estão em todas as X_i, então $\alpha x + (1-\alpha)y$ também estará em cada X_i, pois cada X_i é convexo e, portanto, contém a linha que passa por x e y. Agora, para mostrar que $\alpha x + (1-\alpha)y$ está na interseção $\bigcap_{i \in I} X_i$, precisamos mostrar que $\alpha x + (1-\alpha)y$ está em cada X_i. Como $\alpha x + (1-\alpha)y$ está em cada X_i, é claro que também está na interseção $\bigcap_{i \in I} X_i$. Portanto, a interseção $\bigcap_{i \in I} X_i$ é convexa. ■

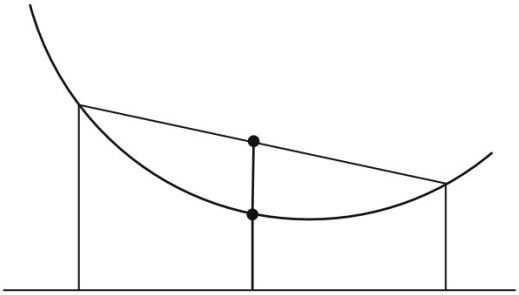
Definição 36. *(Função Convexa): Seja $S \subset \mathbb{R}^n$ um conjunto convexo não vazio. Uma função $f : S \to \mathbb{R}$ diz-se convexa se:*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y)$$

para todo $x, y \in S$ e para todo $\lambda \in [0,1]$.

A Figura 3.5, ilustra um exemplo de uma função convexa.

Figura 3.5: Função convexa



Fonte: [11]

Exemplo 15. *: A função $f(x) = a^T x + b$ é convexa. Para verificar que a função $f(x) = a^T x + b$ é convexa, vamos utilizar a definição de função convexa, na qual, para quaisquer dois pontos x_1 e x_2 no seu domínio e para qualquer valor λ no intervalo $[0, 1]$, a seguinte condição é satisfeita:*

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2). \tag{3.24}$$

Agora, vamos aplicar essa definição à função $f(x) = a^T x + b$. Considere dois pontos x_1 e x_2 no domínio de $f(x)$ e um valor λ no intervalo $[0, 1]$. Substituindo esses valores na desigualdade, temos:

$$f(\lambda x_1 + (1 - \lambda)x_2) = a^T(\lambda x_1 + (1 - \lambda)x_2) + b. \tag{3.25}$$

Expandindo a expressão, temos:

$$f(\lambda x_1 + (1 - \lambda)x_2) = \lambda(a^T x_1) + (1 - \lambda)(a^T x_2) + b. \tag{3.26}$$

Agora, vamos analisar o lado direito da desigualdade:

$$\lambda f(x_1) + (1 - \lambda)f(x_2) = \lambda(a^T x_1 + b) + (1 - \lambda)(a^T x_2 + b). \tag{3.27}$$

Expandindo a expressão, temos:

$$\lambda f(x_1) + (1 - \lambda)f(x_2) = \lambda(a^T x_1) + \lambda(b) + (1 - \lambda)(a^T x_2) + (1 - \lambda)(b). \tag{3.28}$$

Agora, vamos comparar os dois lados da desigualdade:

$$f(\lambda x_1 + (1 - \lambda)x_2) = \lambda(a^T x_1) + (1 - \lambda)(a^T x_2) + b,\ \lambda f(x_1) + (1 - \lambda)f(x_2),$$

logo, teremos que

$$f(\lambda x_1 + (1 - \lambda)x_2) = \lambda(a^T x_1) + \lambda(b) + (1 - \lambda)(a^T x_2) + (1 - \lambda)(b). \tag{3.29}$$

Podemos observar que os termos das equações (3.28) e (3.29) são idênticos em ambos os lados da desigualdade. Portanto, concluímos que a função $f(x) = a^T x + b$ satisfaz a condição de convexidade e, portanto, é uma função convexa.

Proposição 3. *A soma de funções convexas é uma função convexa*

Demonstração. Sejam $f_1 : \mathbb{R}^n \to \mathbb{R}$ e $f_2 : \mathbb{R}^n \to \mathbb{R}$ duas funções convexas. Queremos mostrar que a função $f(x) = f_1(x) + f_2(x)$ é convexa. Para isso, vamos considerar dois pontos $x_1, x_2 \in \mathbb{R}^n$ e um valor $\lambda \in [0, 1]$. Precisamos mostrar que:

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2). \tag{3.30}$$

Vamos começar provando uma propriedade auxiliar. Como f_1 é convexa, temos:

$$f_1(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f_1(x_1) + (1 - \lambda)f_1(x_2). \tag{3.31}$$

Da mesma forma, como f_2 é convexa, temos:

$$f_2(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f_2(x_1) + (1 - \lambda)f_2(x_2). \tag{3.32}$$

Agora, somando as desigualdades (3.31) e (3.32), obtemos:

$$f_1(\lambda x_1 + (1 - \lambda)x_2) + f_2(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f_1(x_1) + (1 - \lambda)f_1(x_2) + \lambda f_2(x_1)$$

$$+(1 - \lambda)f_2(x_2)\ f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda(f_1(x_1) + f_2(x_1)) + (1 - \lambda)(f_1(x_2) + f_2(x_2))\ f(\lambda x_1 +$$

$$(1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2). \tag{3.33}$$

Portanto, mostramos que $f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$, o que significa que a função $f(x) = f_1(x) + f_2(x)$ é convexa. ■

Definição 37. *Seja $X \subseteq \mathbb{R}^n$ um conjunto convexo. Dizemos que a função $f : X \to \mathbb{R}$ é quase convexa se*

$$f(\lambda x + (1 - \lambda)y) \leq \max\{f(x), f(y)\}, \forall\ x, y \in X, \forall\ \lambda \in [0, 1];$$

E a função é quase-côncava se

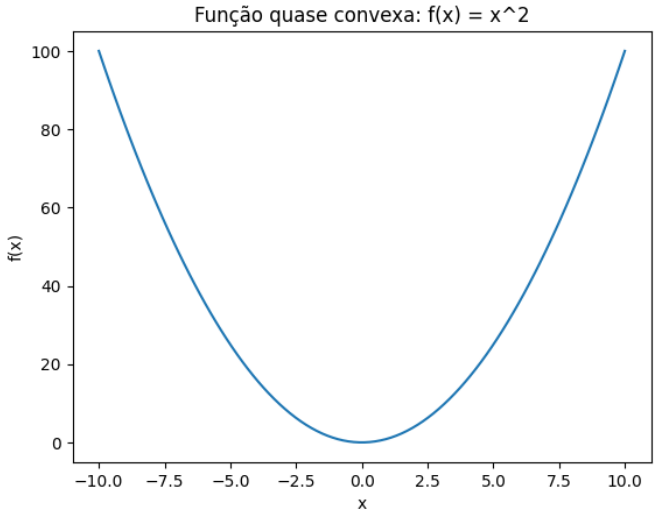
$$f(\lambda x + (1 - \lambda)y) \geq \min\{f(x), f(y)\}, \forall\ x, y \in X, \forall\ \lambda \in [0, 1].$$

Exemplo 16. *Um exemplo de uma função quase convexa é a função $f(x) = x^2, x \in \mathbb{R}$. Para verificar que esta função é quase convexa, podemos aplicar a definição, para qualquer $\lambda \in [0, 1]$, temos:*

$$f(\lambda x + (1 - \lambda)y) = (\lambda x + (1 - \lambda)y)^2 \ = \lambda^2 x^2 + 2\lambda(1 - \lambda)xy + (1 - \lambda)^2 y^2 \quad \leq \lambda x^2 + (1 - \lambda)y^2$$
$$\leq \max\{f(x), f(y)\}.$$

Portanto, $f(x) = x^2$ é uma função quase convexa, que está representada na Figura 3.6.

Figura 3.6: Função quase convexa



Fonte: Autor

Proposição 4. *Se f é uma função convexa, então f é quase-convexa.*

Demonstração. Suponha que f seja uma função convexa. Queremos mostrar que f é quase-convexa. Sejam $x_1, x_2 \in \mathbb{R}^n$ e $\lambda \in [0,1]$ tais que $f(x_1) \leq f(x_2)$. Precisamos mostrar que $f(\lambda x_1 + (1-\lambda)x_2) \leq \max\{f(x_1), f(x_2)\}$. Usando a convexidade de f, temos:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2). \tag{3.34}$$

Como $f(x_1) \leq f(x_2)$, podemos substituir a expressão (3.34) por $f(x_2)$:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2) \leq \lambda f(x_2) + (1-\lambda) f(x_2) = f(x_2). \tag{3.35}$$

Portanto, concluímos que $f(\lambda x_1 + (1-\lambda)x_2) \leq f(x_2)$, o que implica na desigualdade $f(\lambda x_1 + (1-\lambda)x_2) \leq \max\{f(x_1), f(x_2)\}$. ■

Definição 38. *(Conjunto de Nível): Seja $f : X \to \mathbb{R}$. Os conjuntos de nível de f são os conjuntos*

$$N_\alpha := \{x \in X : f(x) \leq \alpha\}$$

para cada $\alpha \in \mathbb{R}$.

O teorema a seguir, cuja demonstração está em [31], estabelece uma relação entre a quase-convexidade de uma função e a convexidade dos seus conjuntos de nível.

Teorema 10. *Uma função f é quase convexa se, e somente se, N_α é um conjunto convexo $\forall\ \alpha \in \mathbb{R}$.*

Corolário 1. *Se f é uma função convexa, então os seus conjuntos de nível são convexos.*

Definição 39. *(Função pseudoconvexa e função pseudocôncava): Seja $X \subseteq \mathbb{R}^n$ um conjunto convexo. A função diferenciável $f : X \to \mathbb{R}$ diz-se pseudoconvexa se*

$$\forall\ x, y \in X, \bigtriangledown f(x)^T (y - x) \geq 0 \Rightarrow f(y) \geq f(x);$$

e é pseudocôncava se

$$\forall\ x, y \in X, \bigtriangledown f(x)^T (y - x) \leq 0 \Rightarrow f(y) \leq f(x).$$

Exemplo 17. *Um exemplo de uma função pseudoconvexa, é a função* $f : \mathbb{R}^2 \to \mathbb{R}$*, dada por* $f(x, y) = x^2 + y^2$*. Para verificar que a função é pseudoconvexa, podemos usar a definição dada. Sejam* $x, y \in \mathbb{R}^2$ *quaisquer. O gradiente de* $f(x, y)$ *é dado por* $\nabla f(x, y) = [2x, 2y]^T$*. Então, temos que:*

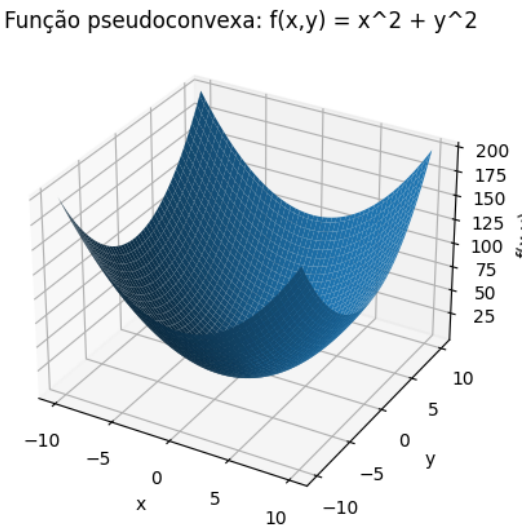
$$\nabla f(x)^T(y - x) = [2x, 2y] \begin{bmatrix} y_1 - x_1 \\ y_2 - x_2 \end{bmatrix} = 2x(y_1 - x_1) + 2y(y_2 - x_2). \tag{3.36}$$

Suponha que $\nabla f(x)^T(y - x) \geq 0$*, então, temos que:*

$$f(y) - f(x) = (y_1^2 + y_2^2) - (x_1^2 + x_2^2) = (y_1 - x_1)(y_1 + x_1) + (y_2 - x_2)(y_2 + x_2). \tag{3.37}$$

Como $\nabla f(x)^T(y - x) \geq 0$*, temos que* $x(y_1 - x_1) + y(y_2 - x_2) \geq 0$*. Como* $x, y \geq 0$*, temos que* $(y_1 - x_1)(y_1 + x_1) + (y_2 - x_2)(y_2 + x_2) \geq 0$*, ou seja,* $f(y) \geq f(x)$*. Portanto, é uma função pseudoconvexa, cuja representação é dada pela Figura 3.7.*

Figura 3.7: Função Pseudoconvexa



Fonte: Autor

Dessa forma, discutimos alguns aspetos fundamentais sobre a convexidade de conjuntos e funções. Na próxima subseção, abordaremos as condições de otimalidade para problemas restritos, explorando as propriedades de convexidade que podem ser utilizadas nesse contexto.

3.2.2 Condições de Otimalidade

Para a resolução de um problema de otimização, é necessário que se cumpram algumas condições, que de certa forma nos garantam a existência de minimizantes, quer locais quer global. Nesta subseção, veremos a condição de Karush-Kuhn-Tucker (KKT).

Teorema 11. *(Condições necessárias de KKT): Consideremos o problema de otimização*

$$\begin{aligned} &\textit{minimize} && f(x) \\ &\textit{sujeito a} && g(x) \leq 0 \\ & && h(x) = 0 \\ & && x \in \mathbb{R}^n, \end{aligned} \tag{3.38}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ e $h : \mathbb{R}^n \to \mathbb{R}^l$ são funções diferenciáveis. Seja $\bar{x}$ admissivel para o problema (3.38), e seja $I = \{i : g_i(\bar{x}) = 0\}$. Suponhamos que o conjunto de vetores $\bigtriangledown h_i(\bar{x})$, $i = 1, ..., l$ e $\bigtriangledown g_i(\bar{x})$, $i \in I$, é um conjunto linearmente independente. Se $\bar{x}$ é um minimizante local de (3.38), então existem multiplicadores $(u, v) \in \mathbb{R}^{m+l}$ tais que

$$\bigtriangledown f(\bar{x}) + \bigtriangledown g(\bar{x})^T u + \bigtriangledown h(\bar{x})^T v = 0 \tag{3.39}$$

$$u \geq 0 \tag{3.40}$$

$$u_i g_i(\bar{x}) = 0, i = 1, ..., m. \tag{3.41}$$

Demonstração. Seja $\bar{x}$ minimizante local de (3.38). Então, $\bar{x}$ tem de satisfazer, obrigatóriamente as condições de necessárias de Fritz John (ver [31]). Se $u_0 > 0$, pelas condições de Fritz John, temos que

$$\bigtriangledown f(\bar{x}) + \sum_{i=1}^{m} \frac{u_i}{u_0} \bigtriangledown g_i(\bar{x}) + \sum_{i=1}^{l} \frac{v_i}{u_0} \bigtriangledown h_i(\bar{x}) = 0,$$

$$1 \geq 0, \frac{u}{u_0} \geq 0,$$

$$(1, \frac{u}{u_0}, \frac{v}{u_0}) \neq 0,$$

$$\frac{u_i}{u_0} g_i(\bar{x}) = 0, i = 1, ..., m.$$

O resultado pretendido segue-se redefinindo

$$u := \frac{u}{u_0}, v := \frac{v}{u_0}.$$

Caso contrário, isto é, se $u_0 = 0$, então

$$\sum_{i \in I} u_i \bigtriangledown g_i(\bar{x}) + \sum_{i=1}^{l} v_i \bigtriangledown h_i(\bar{x}) = 0,$$

e por conseguinte, os gradientes acima são linearmente dependentes. Isto contradiz a hipótese do teorema, pelo que $u_0 = 0$. ■

As condições de KKT (Karush-Kuhn-Tucker) são um conjunto de condições necessárias para que um ponto seja um minimizante local de um problema de otimização restrito. Essas condições podem ser interpretadas algébricamente como um sistema de equações que relaciona os gradientes das funções objetivo e restrições com os multiplicadores de Lagrange. Assim, temos as equações das condições de KKT apresentadas a seguir:

1. $\nabla f(\bar{x}) + \sum_{i=1}^{m} u_i \nabla g_i(\bar{x}) + \sum_{i=1}^{l} v_i \nabla h_i(\bar{x}) = 0;$
2. $u_i g_i(\bar{x}) = 0, \quad \forall\ i = 1, ..., m;$
3. $g_i(\bar{x}) \leq 0, \quad \forall\ i = 1, ..., m$
4. $h_i(\bar{x}) = 0, \quad \forall\ i = 1, ..., l$
5. $u_i \geq 0, \quad \forall\ i = 1, ..., m$

Nesse sistema, o gradiente da função objetivo $\nabla f(\bar{x})$ é combinado com os gradientes das restrições $\nabla g_i(\bar{x})$ e $\nabla h_i(\bar{x})$ através dos multiplicadores de Lagrange u_i e v_i. A primeira equação do sistema estabelece que a combinação linear dos gradientes das funções deve ser igual a zero, indicando um ponto de mínimo. As condições $u_i g_i(\bar{x}) = 0$ representam a complementaridade entre as restrições de desigualdade $g_i(\bar{x}) \leq 0$ e os multiplicadores de Lagrange. Se $u_i > 0$, implica que a restrição $g_i(\bar{x})$ é ativa no ponto $\bar{x}$, ou seja, ela é satisfeita com igualdade. Por outro lado, se $u_i = 0$, a restrição $g_i(\bar{x})$ não é ativa. As restrições de desigualdade e igualdade são expressas como $g_i(\bar{x}) \leq 0$ e $h_i(\bar{x}) = 0$, respectivamente. E os multiplicadores de Lagrange devem ser não negativos, ou seja, $u_i \geq 0$.

Em resumo, as condições de KKT são uma interpretação algébrica das restrições e objetivos do problema de otimização, utilizando os gradientes e multiplicadores de Lagrange para relacioná-los e determinar os pontos que satisfazem as condições de minimização.

Exemplo 18. *O problema restrito a seguir:*

$$\begin{aligned} \textit{minimize} \quad & f(x) = x_1^2 + x_2^2 \\ \textit{sujeito a} \quad & g(x) = 0 \\ & h(x) = x_2^2 - x_1^2 = 0 \\ & x \in \mathbb{R}^n, \end{aligned} \tag{3.42}$$

Satisfaz as condições necessárias de KKT.

Demonstração. Para verificar se o problema dado satisfaz as condições necessárias de KKT, precisamos encontrar os multiplicadores de Lagrange e verificar se as condições são satisfeitas. Calculando os gradientes das restrições:

$$\begin{aligned} \nabla f(x) &= \begin{pmatrix} 2x_1 & 2x_2 \end{pmatrix} \\ \nabla g(x) &= 0 \\ \nabla h(x) &= (-2x_1, 2x_2) \end{aligned}$$

Agora, vamos verificar as condições necessárias de KKT:

1. Condição de estacionaridade:

$$\nabla f(x) + \sum_{i=1}^{m} u_i \nabla g_i(x) + \sum_{i=1}^{l} v_i \nabla h_i(x) = 0$$

Substituindo os gradientes:

$$\begin{pmatrix} 2x_1 & 2x_2 \end{pmatrix} + u \begin{pmatrix} 0 & 0 \end{pmatrix} + v \begin{pmatrix} -2x_1 & 2x_2 \end{pmatrix} = \begin{pmatrix} 2x_1 - 2vx_1 & 2x_2 + 2vx_2 \end{pmatrix} = \begin{pmatrix} (2-2v)x_1 & (2+2v)x_2 \end{pmatrix}$$

Igualando a zero, temos:

$$\begin{aligned} (2-2v)x_1 &= 0 \\ (2+2v)x_2 &= 0 \end{aligned}$$

Para que a condição de estacionaridade seja satisfeita, precisamos ter $x_1 = x_2 = 0$ ou $v = 1$.

2. Condição de complementaridade:

$$u_i \cdot g_i(x) = 0, \quad \forall\ i = 1, ..., m$$

Neste caso, temos apenas uma restrição $g(x) = 0$, então a condição de complementaridade é $u \cdot g(x) = 0$. Portanto, se $x_1 = x_2 = 0$, temos $u = 0$. Caso contrário, se $v = 1$, temos $u \cdot g(x) = 0$.

3. Condição de factibilidade:

$$g(x) \leq 0, \quad \forall\ i = 1, ..., m\ h(x) = 0$$

A restrição $g(x) = 0$ já é satisfeita. Quanto à restrição $h(x) = 0$, temos $x_2^2 - x_1^2 = 0$, que pode ser reescrito como $(x_2 - x_1)(x_2 + x_1) = 0$. Portanto, temos duas possibilidades: $x_2 = x_1$ ou $x_2 = -x_1$.

4. Condição de dualidade:

$$u \geq 0$$

A condição de dualidade é satisfeita quando $u = 0$.

Concluindo, temos as seguintes possibilidades para satisfazer as condições necessárias de KKT:

- Se $x_1 = x_2 = 0$, então $u = 0$ e v pode assumir qualquer valor.
- Se $x_2 = x_1$ ou $x_2 = -x_1$, então u pode assumir qualquer valor não negativo e $v = 1$.

Portanto, o problema dado satisfaz as condições necessárias de KKT. ■

Teorema 12. *(Condições suficientes de KKT). Consideremos o problema de otimização*

$$\begin{array}{ll} \textit{minimize} & f(x) \\ \textit{sujeito a} & g(x) \leq 0 \\ & h(x) = 0 \\ & x \in \mathbb{R}^n, \end{array} \tag{3.43}$$

onde $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ e $h : \mathbb{R}^n \to \mathbb{R}^l$ são funções diferenciáveis. Seja $\bar{x}$ admissivel para o problema, e suponhamos que $\bar{x}$ conjuntamente com os multiplicadores $(u, v) \in \mathbb{R}^{m+l}$ satisfazem as condições necessárias de KKT. Se f é uma função pseudo-convexa, $g_i, i = 1, ..., m$, são funções quase-convexas e $h_i, i = 1, ..., l$ são funções lineares, então $\bar{x}$ é minimizante global do problema.

Demonstração. Uma vez que cada g_i é uma função quase-convexa, os conjuntos de nível $C_i := \{x \in X : g_i \leq 0\}$, $i = 1, ..., m$ são conjuntos convexos (Teorema 16). Por outro lado, dada a linearidade das funções h_i, os conjuntos $D_i := \{x \in X : h_i(x) = 0\}$, $i = 1, ..., l$ são também convexos:

$$x, y \in D_i \Rightarrow h_i(x) = 0 = h_i(y) \Rightarrow h_i(\lambda x + (1 - \lambda)y) = \lambda h_i(x) + (1 - \lambda)h_i(y) = 0,$$

isto é, $\lambda x + (1 - \lambda)y \in D_i$. Assím, uma vez que a intersecção de conjuntos convexos é também um conjunto convexo, a região admissível

$$A = \{x \in X : g(x) \leq 0, h(x) = 0\} = \bigcap_{i=1}^{m} C_i \bigcap_{i=1}^{l} D_i$$

é um conjunto convexo. Seja $I := \{i : g_i(\bar{x}) = 0\}$ o conjunto dos índices das restrições ativas em $\bar{x}$. Seja $x \in A$ um qualquer ponto admissível diferente de $\bar{x}$ (se tal x não existir e $\bar{x}$ for o único ponto admissível, então a solução do problema é trivial e claramente $\bar{x}$ é minimizante global). Então, $\lambda x + (1-\lambda)\bar{x}$ é admissível qualquer que seja o $\lambda \in [0,1]$. Por conseguinte, para $i \in I$ temos

$$g_i(\lambda x + (1-\lambda)\bar{x}) = g_i(\bar{x} + \lambda(1-\bar{x})) \geq 0 = g_i(\bar{x}), \forall\ \lambda \in [0,1].$$

Uma vez que o valor de g_i em $\bar{x}$ não aumenta ao deslocarmo-nos na direção $x-\bar{x}$, temos que $\bigtriangledown g_i(\bar{x})^T(x-\bar{x}) \leq 0$, $\forall\ i \in I$. De modo semelhante, da admissibilidade de $\lambda x + (1-\lambda)\bar{x} = \bar{x} + \lambda(x-\bar{x})$ sabemos que $h_i(\bar{x}+\lambda(x-\bar{x})) = 0$ e, portanto, $\bigtriangledown h_i(\bar{x})^T(x-\bar{x}) = 0\ \forall\ i = 1, ..., l$. Deste modo, da condição de estacionaridade

$$\bigtriangledown f(\bar{x}) = -\bigtriangledown g(\bar{x})^T u - \bigtriangledown h(\bar{x})^T v$$

e da condição de não negatividade $u \geq 0$, resulta que

$$\bigtriangledown f(\bar{x})(x-\bar{x}) = -(\bigtriangledown g(\bar{x})^T u + \bigtriangledown h(\bar{x})^T v)^T (x-\bar{x}) \geq 0$$

o que, pela pseudoconvexidade de f, implica que $f(x) \geq f(\bar{x})$ (para todo o x admissível). ■

As condições suficientes de KKT estabelecem uma condição adicional que, quando satisfeita juntamente com as condições necessárias de KKT, garante que um ponto de solução viável seja um minimizante global do problema de otimização. As principais condições envolvidas são:

1. Pseudoconvexidade da função objetivo f: A pseudoconvexidade é uma propriedade que implica que o conjunto de nível de uma função é convexo. Nesse caso, a função objetivo f é pseudoconvexa, o que garante que os conjuntos de nível C_i são convexos.

2. Quase-convexidade das funções de restrição g_i: As funções de restrição g_i são quase-convexas. Isso significa que os conjuntos de nível de g_i, denotados por C_i, também são convexos.

3. Linearidade das funções de restrição h_i: As funções de restrição h_i são lineares. Isso implica que os conjuntos de nível de h_i, denotados por D_i, são convexos.

4. Convexidade da região admissível A: A interseção dos conjuntos de nível C_i e D_i, ou seja, a região admissível A, é um conjunto convexo.

O **Teorema 12** estabelece que, se um ponto $\bar{x}$, juntamente com os multiplicadores de Lagrange (u, v), satisfaz as condições necessárias de KKT e as condições suficientes mencionadas acima, então $\bar{x}$ é um minimizante global do problema de otimização.

Definição 40. *O problema de otimização dado pela equação 3.18, diz-se convexo se X for um conjunto aberto convexo, f, g_i, $i = 1, ..., m$ forem funções convexas e h_i, $i = 1, ..., l$, são funções lineares.*

Corolário 2. *Consideremos um problema de otimização convexo, e $\bar{x}$ é admissivel e satisfaz, conjuntamente com os multiplicadores $(u, v) \in \mathbb{R}^n$, as condições necessárias de KKT, então $\bar{x}$ é um minimizante global.*

Demonstração. Ver [31] ■

3.2.3 Métodos de otimização para Problemas Restritos

Nesta seção, faremos uma abordagem sobre alguns métodos iterativos para problemas restritos, onde as referências principais são [30] e [31].

3.2.3.1 Métodos de Penalização

Consideremos a região admissível do problema de otimização (problema 3.18) dada por:

$$A := \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, ..., m, h_i(x) = 0, i = 1, ..., l\},$$

em todo $\mathbb{R}^n$. O método de penalização consiste em substituir a resolução de um problema restrito por um irrestrito, tornando mais fácil a resolução. Para isso, adiciona-se um custo muito grande (uma penalização) à função objetivo f, que pretende-se minimizar. Sem perda de generalidade, pode-se restringir a problemas com apenas restrições de desigualdade todo e qualquer problema com restrição de igualdade, fazendo

$$h_i(x) = 0 \Rightarrow \begin{cases} h_i(x) \leq 0 \\ -h_i(x) \leq 0 \end{cases} \tag{3.44}$$

Assim sendo, todo o problema de otimização pode ser reescrito como

$$\begin{aligned} \min \quad & f(x) \\ \text{sujeita a} \quad & G(x) \leq 0 \\ & x \in \mathbb{R}^n \end{aligned}$$

onde $G(x) = (G_1(x), G_2(x), ..., G_{m+2l}(x))$ é uma função do $\mathbb{R}^n$ em $\mathbb{R}^{m+2l}$.

Definição 41. *(Função de Penalização): Uma função $p : \mathbb{R}^n \to \mathbb{R}$ é chamada de função de penalização para (3.18) se ela satisfaz, para todo o $x \in \mathbb{R}^n$, as seguintes condições:*

1. *$p(x) = 0$ se $G(x) \leq 0$ (não há penalização para os x admissíveis);*
2. *$p(x) > 0$ se $G(x) \nleq 0 \Leftrightarrow \exists i : G_i(x) > 0$ (há penalização sempre que x não é admissível).*

Exemplo 19. *A função*

$$p(x) = \sum_{i=1}^{m} (\max\{0, G_i(x)\})^q \tag{3.45}$$

é de penalização, para $q \geq 0$.

Assim sendo, o método de penalização substitui o problema restrito pelo irrestrito da forma

$$\min_{x \in \mathbb{R}^n} q(c, x) = (f(x) + cp(x)), \tag{3.46}$$

onde c é o parâmetro de penalização. Assim sendo, teremos o algoritmo de penalização (**Algoritmo 4**):

Algoritmo 4: Algoritmo de Penalização

Entrada: Função f e restrições g que definem o problema que pretende-se resolver, função de penalização p, parâmetro de penalização inicial c_0, parâmetro $\eta > 1$ de crescimento das constantes c_k, tolerância $\epsilon > 0$ para o critério de parada, aproximação inicial $x^0 \in \mathbb{R}^n$ que viola pelo menos uma restrição (método de ponto exterior).

Saída: Aproximação x^k para a solução do problema com

$$||x^k - x^{k-1}|| < \epsilon$$

ou

$$|f(x^k) - f(x^{k-1})| < \epsilon.$$

1. Inicialização: Formular a função objetivo aumentada $q(c_0, x)$ baseada nas restrições g_i violadas em x_0. Fazer $k = 1$.
2. Partindo de x^{k-1}, usar um dos métodos aproximativos para resolver o problema irrestrito $\min_{x \in \mathbb{R}^n} q(c_{k-1}, x)$. Seja x^k a aproximação encontrada.
3. Se $||x^k - x^{k-1}|| < \epsilon$ ou $|f(x^k) - f(x^{k-1})| < \epsilon$, então pare, com x^k a aproximação da solução do problema com restrições; Caso contrário, prosseguir.
4. Fazer $c_k := \eta c_{k-1}$ e formular $q(c_k, x)$ baseada nas restrições g_i que são violadas em x^k.
5. Fazer $k := k + 1$ e voltar ao passo 2.

Neste algoritmo, temos que:

1. Inicialização: Neste passo, formulamos a função objetivo aumentada $q(c_0, x)$ com base nas restrições g_i violadas em x_0. Também iniciamos o contador k com o valor 1.

2. Laço principal: Dentro deste laço, partimos da aproximação anterior x^{k-1} e usamos um método aproximativo para resolver o problema irrestrito $\min_{x \in \mathbb{R}^n} q(c_{k-1}, x)$. A aproximação encontrada é denotada por x^k.

3. Critério de parada: Verificamos se a diferença entre as aproximações x^k e x^{k-1} é menor que a tolerância ϵ, ou se a diferença entre os valores da função objetivo $f(x^k)$ e $f(x^{k-1})$ é menor que ϵ. Se um desses critérios for satisfeito, paramos o algoritmo e retornamos a aproximação x^k como a solução do problema com restrições. Caso contrário, continuamos para o próximo passo.

4. Atualização dos parâmetros: Neste passo, atualizamos o parâmetro de penalização c_k multiplicando o parâmetro anterior c_{k-1} pelo fator de crescimento $\eta > 1$. Também formulamos a função objetivo aumentada $q(c_k, x)$ com base nas restrições g_i que são violadas em x^k.

5. Atualização do contador: Incrementamos o contador k em 1 e retornamos ao passo 2 para repetir o processo com a nova aproximação x^k.

3.2.3.2 Métodos de Barreira

Num método de barreira, assumimos que nos é dada uma aproximação inicial x^0 da solução do problema

$$\begin{gathered} minimizar\ f(x), \\ sujeita\ a\ g(x) \leq 0 \\ x \in \mathbb{R}^n, \end{gathered} \tag{3.47}$$

que pretendemos resolver, pertencente ao interior da região admissível A. Impomos um custo muito grande às aproximações admissíveis geradas pelo método que estão perto da fronteira de A, deste modo criando uma "barreira"que evita que as aproximações saiam da região admissível. Nestes métodos, partimos de uma aproximação dentro da região admissível, aproximando-nos da solução de (3.18) a partir do interior da região admissível, com valores da função objetivo cada vez menores $f(x^{k+1}) \leq f(x^k)$ e com valores da chamada função de barreira b cada vez maiores em cada iteração $b(x^{k+1}) \geq b(x^k)$, conforme veremos a seguir.

Definição 42. *(Função de Barreira). Dizemos que $b : \mathbb{R}^n \to \mathbb{R}$ é uma função de barreira para o problema (3.47) se ela satisfaz, para todo o $x \in \mathbb{R}^n$, as seguintes condições:*

1. $b(x) \geq 0$ ***se*** $g(x) < 0$;
2. $b(x) \to +\infty$ ***se*** $\max\limits_{i=1,..,m} \{g_i(x)\} \to 0$

Exemplo 20. *A função*

$$b(x) = \sum_{i=1}^{m} (-g_i(x))^{-q}, q > 0$$

é uma função de barreira.

Exemplo 21. *A função*

$$b(x) = -\sum_{i=1}^{m} ln(-g_i(x))$$

é conhecida como a função de barreira logarítmica.

A ideia do método de barreira é substituir a resolução de um problema com restrições na resolução de uma sequência de problemas de otimização sem restrições do tipo:

$$\min_{x \in X} \left(f(x) + \frac{1}{c} b(x) \right), \tag{3.48}$$

onde X é o conjunto aberto $X = \{x \in \mathbb{R}^n : g(x) < 0\}$ e c é uma sucessão de constantes c_k, em que $\lim\limits_{k \to \infty} c_k \to +\infty$. A seguir, temos o algoritmo de Barreira (**Algoritmo 5**).

Algoritmo 5: Algoritmo de Barreira

Entrada: x^0 (factível), f, restrições g_i, função de barreira b, parâmetro inicial c_0, parâmetro $\eta > 1$ de crescimento das constantes c_k, tolerância $\epsilon > 0$.

Saída: Aproximação x^k para a solução do problema com $||x^k - x^{k-1}|| < \epsilon$ ou $|f(x^k) - f(x^{k-1})| < \epsilon$.

1. Inicialização: Formular a função objetivo $f(x) + \frac{1}{c}b(x)$. Fazer $k := 1$.
2. Partindo de x^{k-1}, usar um método aproximativo estudado neste capítulo para resolver o problema sem restrição $\min_{x \in X} \left(f(x) + \frac{1}{c}b(x) \right)$. Seja x^k a aproximação encontrada.
3. Se $||x^k - x^{k-1}|| < \epsilon$ ou $|f(x^k) - f(x^{k-1})| < \epsilon$, então, pare, com x^k a aproximação da solução do problema com restrições; caso contrário, prosseguir.
4. Fazer $c_k := \eta c_{k-1}$ e formular $\min_{x \in X} \left(f(x) + \frac{1}{c}b(x) \right)$.
5. Fazer $k := k + 1$ e voltar ao passo 2.

3.2.3.3 Método do gradiente Ponderado

Consideremos o problema de otimização não linear:

$$\begin{aligned} \min \quad & f(x) \\ \text{sujeita a} \quad & g_i(x) \leq 0, i = 1, ..., m \\ & x \in \mathbb{R}^n, \end{aligned} \tag{3.49}$$

onde $f(x)$ e $g_i(x)$ são diferenciáveis. A equação (3.49) pode ser reescrita como:

$$\min_{x \in \mathbb{R}^n} P(\sigma, x) = f(x) + \sigma \sum_{i=1}^{m} (\max\{0, G_i(x)\}), i = 1, ..., m, \tag{3.50}$$

onde σ é o parâmetro de penalização. Consideremos a região admissível de (3.49), dada por:

$$A := \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, ..., m\}. \tag{3.51}$$

Para um valor de x, se $x \in A$, movendo x ao longo da direção negativa do gradiente $- \bigtriangledown f(x)$ da função objetivo, a função objetivo pode melhorar. Se $x \notin A$, significa que x está fora da região admissível. Vamos assumir que:

$$I(x) = \{i | g_i(x) > 0, \ i = 1, 2, ..., m\}, \tag{3.52}$$

para $i \in I$, movendo x ao longo da direção do gradiente negativo $- \bigtriangledown g(x)$, $g_i(x)$ decresce e pode satisfazer $g_i(x) \leq 0$. Tendo em conta este fato, a direção ponderada pode ser definida a seguir como:

$$d(x) = - \bigtriangledown f(x) - \sum_{i=1}^{n} w_i \bigtriangledown g_i(x), \tag{3.53}$$

onde w_i são os pesos da direção do gradiente, e são definidos como:

$$w_i = \begin{cases} 0, & g_i(x) \leq 0 \\ \delta_i, & g_i(x) > 0, \end{cases} \tag{3.54}$$

onde:

$$\delta_i = \frac{1}{1 - G_i(x)/(G_{\max(x)} + \epsilon)}, \tag{3.55}$$

$$G_i(x) = \frac{g_i(x)}{\|\nabla g_i(x)\|} \tag{3.56}$$

e

$$G_{\max}(x) = \max\{0, G_i(x),\ i = 1, 2, ..., m\}. \tag{3.57}$$

em (3.55), ϵ é um número positivo, suficientemente pequeno. Então $x^{(k+1)}$ é gerado de x^k por atualizações ao longo da direção do gradiente ponderado $d(x)$ e é descrito como:

$$x^{(k+1)} = x^{(k)} + \lambda d(x^{(k)}), \tag{3.58}$$

onde $\lambda > 0$ é o parâmetro que indica o tamanho do passo e satisfaz:

$$\frac{d}{d\lambda} P(x^{(k)} + \lambda d(x^{(k)}), \sigma) = 0 \tag{3.59}$$

De (3.53) a (3.54), é óbvio que para $g_i(x) > 0$, o aumento de $g_i(x)$ causa o aumento do peso da direção do gradiente. Além disso, $d(x)$ é efetivamente uma direção de busca, porque se $x \in A$, então $d(x) = - \bigtriangledown f(x)$ e se move ao longo da direção de $- \bigtriangledown f(x)$. Consequentemente, a função objetivo pode ser melhorada. Também, se $x \notin A$, quanto maior for $g_i(x)$, mais distante x está do conjunto admissível A, o peso elevado w_i pode ser obtido para se mover para o conjunto admissível. Portanto, x converge para uma solução de mínimo local pela direção de gradiente ponderado. Assim, temos o **Algoritmo 6**.

Algoritmo 6: Método do Gradiente Ponderado

Entrada:

- Função objetivo f
- Restrições g_i
- Aproximação inicial x^0
- Tamanho de passo λ
- Tolerância ϵ

Saída:

- Aproximação x para a solução do problema de otimização

1: **Inicialização:** Definir $k = 0$
2: Calcular o conjunto $I(x^k)$ das restrições ativas em x^k
3: Calcular os pesos w_i para as restrições ativas de acordo com a equação (3.54)
4: Calcular a direção ponderada $d(x^k)$ de acordo com a equação (3.53)
5: Atualizar a aproximação: $x^{k+1} = x^k + \lambda d(x^k)$
6: Verificar o critério de parada: se $\|x^{k+1} - x^k\| < \epsilon$ ou $|f(x^{k+1}) - f(x^k)| < \epsilon$, então parar e retornar x^{k+1} como a aproximação da solução
7: Caso contrário, atualizar k: $k = k + 1$ e voltar ao passo 2

Os métodos de otimização estudados até agora, tanto irrestritos quanto restritos, compartilham uma característica em comum: eles pressupõem que as funções envolvidas sejam

diferenciáveis. No entanto, existem casos em que precisamos minimizar funções que não são diferenciáveis, como é o caso do problema do despacho econômico com efeito do ponto de carregamento de válvula. Portanto, é necessário abordar essa questão e buscar soluções para minimizar funções não diferenciáveis. Na próxima seção, vamos explorar uma abordagem conhecida como suavização. Essa técnica tem como objetivo resolver o problema de minimização de funções não diferenciáveis, aproximando-as por funções suaves que possam ser otimizadas utilizando os métodos tradicionais. A suavização envolve a substituição da função original por uma versão suave, que preserva as principais características da função original, mas permite a aplicação de métodos diferenciáveis.

3.2.4 Suavização

De acordo com [28], a suavização é uma técnica amplamente utilizada em otimização para lidar com funções não diferenciáveis. Consiste em substituir uma função não diferenciável por uma versão suave que seja diferenciável, permitindo assim a aplicação de métodos de otimização que requerem derivadas. Uma vez que a função não diferenciável tenha sido suavizada, podemos aplicar métodos de otimização, como o gradiente descendente, Newton e outros métodos baseados em gradiente, para encontrar o mínimo da função suavizada. A solução obtida será uma aproximação do mínimo da função original. É importante destacar que a suavização é uma técnica aproximada e a solução encontrada será uma aproximação do mínimo verdadeiro da função não diferenciável. No entanto, em muitos casos práticos, essa abordagem pode fornecer resultados satisfatórios e permitir a aplicação de métodos de otimização mais eficientes. A suavização em otimização é especialmente útil quando lidamos com problemas reais que envolvem restrições ou funções não diferenciáveis, como no caso do despacho econômico, um problema importante na engenharia elétrica. Ao suavizar as funções envolvidas, podemos aplicar métodos de otimização para obter soluções aproximadas e viáveis para esses problemas complexos.

Definição 43. *(Função de Suavização): Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função não diferenciável. A função de suavização $S(f) : \mathbb{R}^n \to \mathbb{R}$ é definida como uma transformação que mapeia f em uma função suave g tal que:*

$$S(f)(x) = g(x),$$

onde $x \in \mathbb{R}^n$.

A função suave $g(x)$ é obtida utilizando-se uma técnica de suavização, como a suavização hiperbólica (que será estudada nesta seção) ou uma outra técnica de suavização, preservando as características essenciais de f e tornando-a diferenciável. O objetivo é permitir a aplicação de métodos de otimização que requerem derivadas em $g(x)$ para encontrar um mínimo aproximado da função original f.

3.2.4.1 Suavização Hiperbólica

Como estamos interessados em problemas que envolvem funções modulares, sem perda de generalidade, vamos abordar a suavização hiperbólica para o caso de funções modulares. A função módulo, denotada por $|x|$, é uma função não diferenciável definida como:

$$|x| = \begin{cases} x, & \text{se } x \geq 0, \\ -x, & \text{se } x < 0. \end{cases}$$

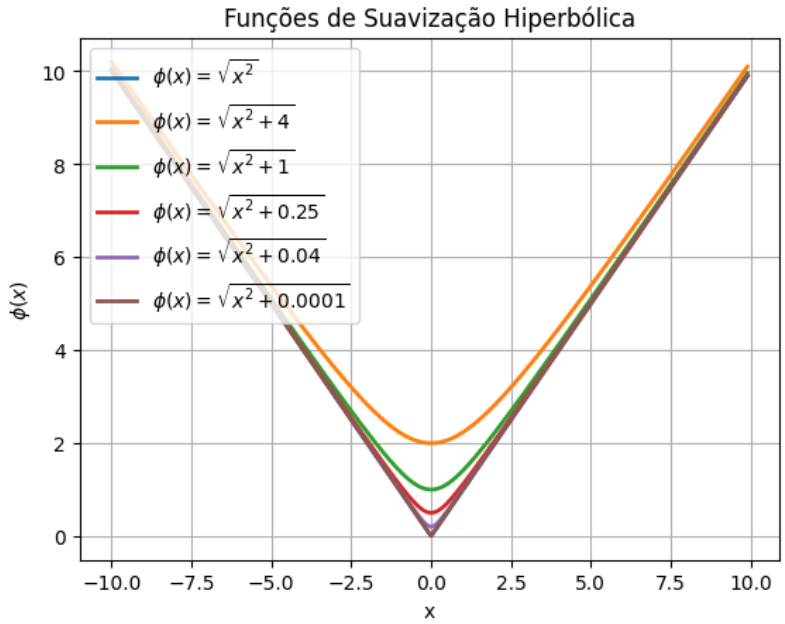
Para realizar a suavização hiperbólica usando a função módulo, podemos substituir a função módulo por uma sequência de funções suaves que se aproximam da função módulo.

Uma função suave comumente usada é a função de suavização hiperbólica, que é definida como:

$$\phi(x, \eta) = \sqrt{x^2 + \eta},$$

onde η é um parâmetro positivo que controla a suavidade da função. Ao substituir a função módulo pela função de suavização hiperbólica, obtemos uma função suave e diferenciável que se aproxima da função módulo. Essa função suave pode ser usada em problemas de otimização que envolvem a função módulo, permitindo a aplicação de métodos tradicionais de otimização que requerem derivadas. A Figura 3.8 representa a suavização hiperbólica da função $f(x) = |x|$, no caso $\phi(x) = \sqrt{x^2 + \eta}$, para diferentes valores de η. O que podemos observar é que na medida em que $\eta \to 0$, a função suavizada fica mais próxima da função original, ao passo que na medida que $\eta \to \infty$, a função suavizada fica cada vez mais distante da função original.

Figura 3.8: Função de Suavização Hiperbólica para diferentes valores de η



Assim, a função de suavização hiperbólica $\phi(x, \eta) = \sqrt{x^2 + \eta}$ possui as seguintes propriedades:

1. Suavização: A função $\phi(x, \eta)$ é diferenciável para todos os valores de x, incluindo $x = 0$, e seu valor absoluto é igual a $|x|$ para $|x|$ suficientemente grande. Conforme η se aproxima de zero, a função se torna cada vez mais próxima da função módulo $|x|$.

2. Suavidade crescente: À medida que o valor de η aumenta, a função $\phi(x, \eta)$ se torna mais suave, ou seja, sua derivada se torna mais contínua. Isso significa que a inclinação da função em torno de pontos críticos, como mínimos e máximos, se torna menos acentuada à medida que η aumenta.

3. Conservação de mínimos: A função $\phi(x, \eta)$ preserva os mínimos da função módulo $|x|$. Isso significa que se um mínimo ocorre em x_0 na função $|x|$, então também ocorrerá um mínimo em x_0 na função suavizada $\phi(x, \eta)$ para qualquer valor de $\eta > 0$. No entanto, o valor do mínimo na função suavizada será ligeiramente maior do que na função módulo.

4. Estabilidade numérica: A função de suavização hiperbólica $\phi(x, \eta)$ é numericamente estável, o que significa que ela pode ser avaliada e diferenciada com precisão mesmo para valores de x muito grandes ou muito pequenos. Isso torna a função útil para aplicações numéricas e algoritmos de otimização.

Essas propriedades tornam a suavização hiperbólica uma ferramenta valiosa na otimização e em outras áreas onde é necessário lidar com funções não diferenciáveis. A escolha adequada do parâmetro η permite ajustar o grau de suavidade da função de acordo com as necessidades específicas do problema. Apesar das soluções propostas para lidar com a suavização, os algoritmos estudados compartilham um critério de parada que resulta na estagnação do minimizante em uma vizinhança de pontos, à medida que a norma do gradiente se aproxima de zero. Essa característica dificulta a resolução de problemas multimodais. Na próxima seção, abordaremos essa limitação de forma mais detalhada.

3.2.5 Limitação dos Métodos de Gradiente Para Problemas Multimodais

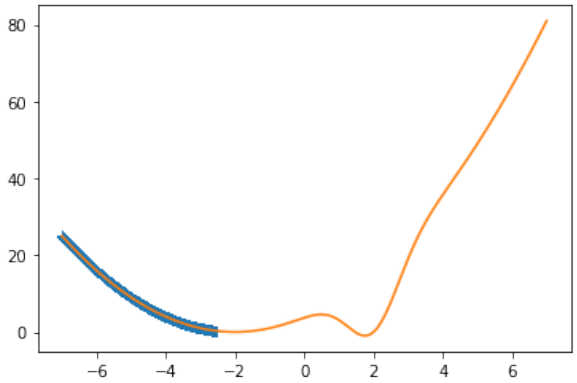
A limitação mencionada no final da seção anterior (**Seção 3.2.4**), refere-se à dificuldade dos algoritmos de otimização baseados em gradientes em lidar com problemas multimodais. Como os problemas multimodais possuem múltiplas soluções ótimas, cada solução representa um mínimo local na função objetivo. Quando a norma do gradiente tende a zero durante a otimização, indica que o algoritmo está se aproximando de um mínimo local, no entanto, essa convergência para um mínimo local pode fazer com que o algoritmo fique preso nessa vizinhança de pontos, não conseguindo explorar outras regiões do espaço de busca para encontrar possíveis mínimos globais. Essa limitação pode ser problemática em situações em que o objetivo é encontrar a melhor solução global possível.

Para melhor compreensão, veremos um exemplo de minimização de uma função, usando um dos métodos de gradiente, isto é, o método do gradiente descendente.

Exemplo 22. *Para este exemplo, vamos minimizar a função* $f(x) = (x+2)^2 - 16e^{-(x-2)^2}$.

Considerado o ponto inicial $x_0 = -7$ e uma precisão de 0.000999, obtemos como minimizante o ponto -2.5483126946918735 e o mínimo é 0.3006467945718216, como ilustrado na Figura 3.9.

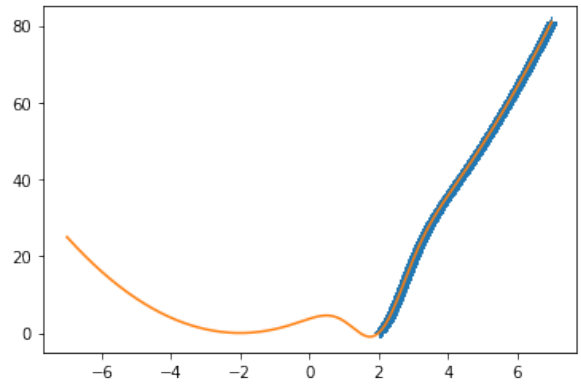
Figura 3.9: Método do Gradiente Descendente para $x_0 = -7$ e $\epsilon = 0.000999$



Fonte: Autor

Quando considerado o ponto inicial $x_0 = 7$ e uma precisão de 0.000999, obtemos como minimizante o ponto 2.0321487333200623 e o mínimo é 0.2747515217663832, como ilustrado na Figura 3.10.

Figura 3.10: Método do Gradiente Descendente para $x_0 = 7$ e $\epsilon = 0.000999$



Fonte: Autor

Com base na escolha do ponto inicial x_0, observamos que a solução do problema é diferente, indicando que se trata de um problema multimodal com dois pontos de mínimo. Para resolver problemas multimodais, os métodos de gradiente mostram-se ineficientes, especialmente quando o ponto inicial x_0 está longe do melhor mínimo global, mas próximo do pior mínimo global. Nessas situações, o método pode convergir para um mínimo que não é o ótimo global. Assim, é necessário aprimorar os algoritmos de otimização, incorporando recursos adicionais que permitam lidar com a determinação de múltiplos pontos de mínimo.

Concluindo este capítulo, observamos que o estudo da otimização não linear desempenha um papel fundamental na compreensão e resolução de uma ampla gama de problemas complexos. Ao explorar tanto problemas irrestritos quanto restritos, pudemos analisar e aplicar as condições de otimalidade pertinentes. Além disso, ao investigar diversos métodos baseados em gradientes, adquirimos ferramentas práticas para encontrar soluções aproximadas. No entanto, é igualmente importante reconhecer as limitações intrínsecas desses métodos e a necessidade de considerar abordagens alternativas para problemas que não se adequam à sua estrutura. O estudo da otimização não linear fornece uma base sólida para aprofundar-se em áreas relacionadas, abrindo caminho para a resolução de desafiantes problemas em diversos campos, como ciência, engenharia e economia. As referências citadas neste capítulo oferecem uma base bibliográfica sólida, permitindo-nos explorar as contribuições de renomados pesquisadores nesse campo dinâmico e em constante evolução.

Considerando as limitações dos métodos de gradientes discutidos neste capítulo, no próximo capítulo, exploraremos os sistemas dinâmicos, com ênfase no caos e na teoria de Julia. Essas abordagens têm o potencial de aprimorar a identificação de pontos de mínimo em problemas de otimização multimodal. Ao incorporar os princípios dessas teorias, poderemos expandir nossa compreensão das dinâmicas complexas dos problemas e, assim, melhorar a capacidade de encontrar soluções ótimas.

Capítulo

4

Fractais e Caos

A geometria clássica estuda assuntos como formas suaves e regulares. Durante a maioria dos estudos em matemática, são abordados temas relacionados a funções, que são contínuas, diferenciáveis e têm uma representação simples. No entanto, ao observarmos ao nosso redor, não estamos habituados a ver formas regulares e familiares, como funções seno, exponenciais e polinômios. Em vez disso, vemos formas complicadas que parecem difíceis de entender, como a fronteira das nuvens, o perfil das montanhas, os galhos de uma árvore, o desdobramento de um delta de um rio e a bifurcação de um raio. Todos esses fenômenos parecem ter aspectos que não são descritos na geometria clássica, mas muitas dessas formas têm certas propriedades em comum. O estudo dessas formas complexas deu origem a um ramo da matemática conhecido como Geometria Fractal, que será abordado neste capítulo. Nele, exploraremos alguns objetos que ao longo dos séculos passaram a ser chamados de fractais, com destaque para o conjunto de Julia (Fractal de Julia) e suas propriedades fundamentais. Para este capítulo, utilizaremos as referências [3], [4], [8], [10], [13] e [24].

4.1 Fractais

Nesta seção, antes de definirmos fractal, veremos em primeiro lugar alguns exemplos de fractais, dos quais destacamos o conjunto de Cantor, curva de Von Koch e o triângulo de Sierpinski.

O primeiro exemplo é o conjunto de Cantor, que é um exemplo de fractal cuja construção é muito simples. Construímos o Conjunto de Cantor do terço médio a partir do intervalo $[0,1] \subset \mathbb{R}$ removendo repetidamente certos intervalos abertos. Na primeira etapa, remove-se o terço médio $(\frac{1}{3}, \frac{2}{3})$, deixando $[0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$. Chame este estágio de C_1. Para prosseguir para C_2, remove-se o terço médio de cada um dos dois intervalos fechados restantes, deixando $[0, \frac{1}{9}] \cup [\frac{2}{9}, \frac{1}{3}] \cup [\frac{2}{3}, \frac{7}{9}] \cup [\frac{8}{9}, 1]$. Em geral, passamos do passo C_k para o passo C_{k+1} removendo o terceiro intervalo aberto do meio de cada um dos 2^k intervalos fechados restantes. O Conjunto de Cantor C é então formado por todos os pontos que permanecem em C_k para todos os números naturais k, ou explicitamente,

$$C = \bigcap_{i=1}^{\infty} C_i. \tag{4.1}$$

O comprimento de cada intervalo fechado na k-ésima iteração é 3^{-k}, que tende a zero quando k tende ao infinito, então uma questão válida seria se esse processo de remoção

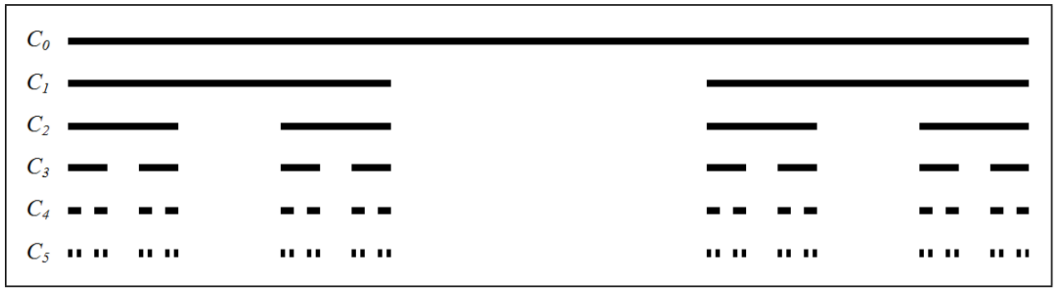
deixa algum elemento em $[0, 1]$. Bem, alguns pontos permanecem; 0 e 1, por exemplo, claramente nunca são removidos por este processo e, de fato, o Conjunto de Cantor C consiste inteiramente em um número incontável e infinito de pontos. Isso revela algumas propriedades importantes a se destacar;

i) Como o comprimento de cada intervalo na k-ésima iteração tende a zero quando k vai para infinito, os detalhes do conjunto da Cantor são exibidos em escalas arbitrariamente pequenas, ou seja, os intervalos ficam cada vez menores.

ii) O comprimento de cada passo C_k é $(\frac{2}{3})^k$, então por qualquer definição usual, o conjunto limite C deve ter comprimento zero. No entanto, também é incontávelmente infinito, então, num sentido, é um conjunto grande e, noutro, é muito pequeno. O que torna difícil classificar o Conjunto de Cantor por propriedades como tamanho.

iii) Finalmente, embora C exiba uma estrutura intrincadamente detalhada, na verdade ela é descrita de maneira muito direta por meio de um procedimento recursivo simples. No entanto, não pode ser facilmente descrito em um sentido clássico. Não é o lugar geométrico dos pontos que satisfazem alguma condição geométrica simples, nem é o conjunto de soluções de qualquer equação simples. A seguir, temos a ilustração do conjunto de Cantor na Figura 4.1.

Figura 4.1: Conjunto de Cantor



Fonte: [8]

O segundo exemplo, é a curva de von Koch. A sua construção é baseada num processo iterativo, similar ao do conjunto de Cantor. Para construir a curva, começamos com um segmento de reta de unidade de comprimento, V_0, e um gerador, V_1. Neste caso, o gerador é o segmento de reta unitária com o terço médio substituído por triângulo equilátero, cujo comprimento da base é o terço médio que é retirado (veja a Figura 4.2). V_2 é gerado substituindo cada segmento de linha reta em V_1 por uma cópia apropriadamente dimensionada e rotacionada do gerador. Em geral, passamos da iteração V_k para a iteração V_{k+1} substituindo cada segmento de linha reta de V_k por uma cópia adequadamente dimensionada e rotacionada de V_1, o gerador. Vamos denotar por f a função que recebe V_k e retorna V_{k+1}, de modo que $f^k(V_0) = V_k$, com f^k representando a aplicação k- dobra (k-fold) de f. Então, a curva de Von Koch, V , é

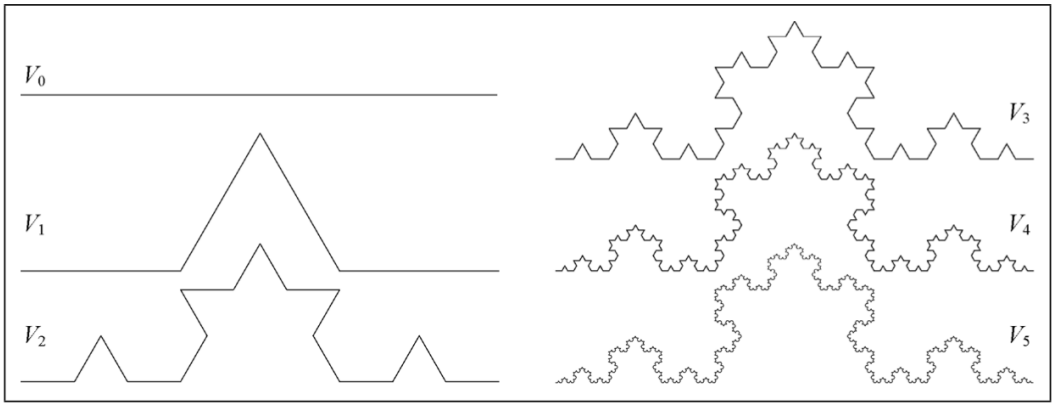
$$V = \lim_{k\to\infty}(f^k(V_0)) \tag{4.2}$$

Na construção, cada vez que um segmento de reta é alterado por f, ele é substituído por quatro segmentos de reta, cada um com um terço do comprimento removido. Então, V_k, o k-ésimo passo da construção tem comprimento $(\frac{4}{3})^k$. A curva limite V, portanto, tem comprimento infinito. Se tomarmos V_0 como o intervalo $[0, 1]$ no eixo x como um subconjunto de $\mathbb{R}^2$, a área delimitado por V e o eixo x não é infinito; certamente é limitada acima pela área do retângulo $[0, 1] \times [0, \frac{\sqrt{3}}{6}]$. Assim, temos uma curva de comprimento

infinito que encerra uma área finita. A curva em si, no entanto, não ocupa nenhuma área no plano e, portanto, torna difícil falar sobre o "tamanho" dessa curva em um sentido clássico.

Há também outras propriedades que a curva de von Koch tem em comum com o nosso exemplo anterior. Primeiro, ela exibe uma espécie de auto-simetria. Se considerarmos qualquer parte da curva e ampliá-la, dará uma imagem um pouco parecida com a original. Em segundo lugar, seria trabalhoso para descrevê-la adequadamente usando a linguagem da geometria euclidiana e, finalmente, podemos realmente defini-la de uma maneira muito simples, embora recursiva, como ilustrada na Figura 4.2.

Figura 4.2: Curva de Von Koch



Fonte: [8]

O nosso terceiro exemplo de fractal é triângulo de Sierpinski. Para gerar o fractal, começamos com um triângulo equilátero preenchido, chamado S_0. Para passar para S_1 divide-se ao meio cada lado do triângulo equilátero inicial. Unem-se esses pontos obtendo um triângulo equilátero, removemos esse triângulo equilátero invertido de S_0 para deixar a forma com três triângulos equiláteros preenchidos idênticos. Em geral, para passar do passo S_k para S_{k+1}, removemos um triângulo equilátero invertido de cada um dos 3^k triângulos preenchidos restantes, deixando 3^{k+1} desses triângulos; as primeiras aplicações deste processo podem ser vistas na Figura 4.3. Para obter o fractal, tomamos um processo de limitação com o seguinte algoritmo: Seja f a função que remove os triângulos invertidos 3^k no passo k, tal que $f(S_k) = S_{k+1}$ e $f^k(S_0) = S_k$ com f^k sendo novamente a aplicação k- removedora (k-fold) de f. Então o Triângulo de Sierpinski, S é dado por

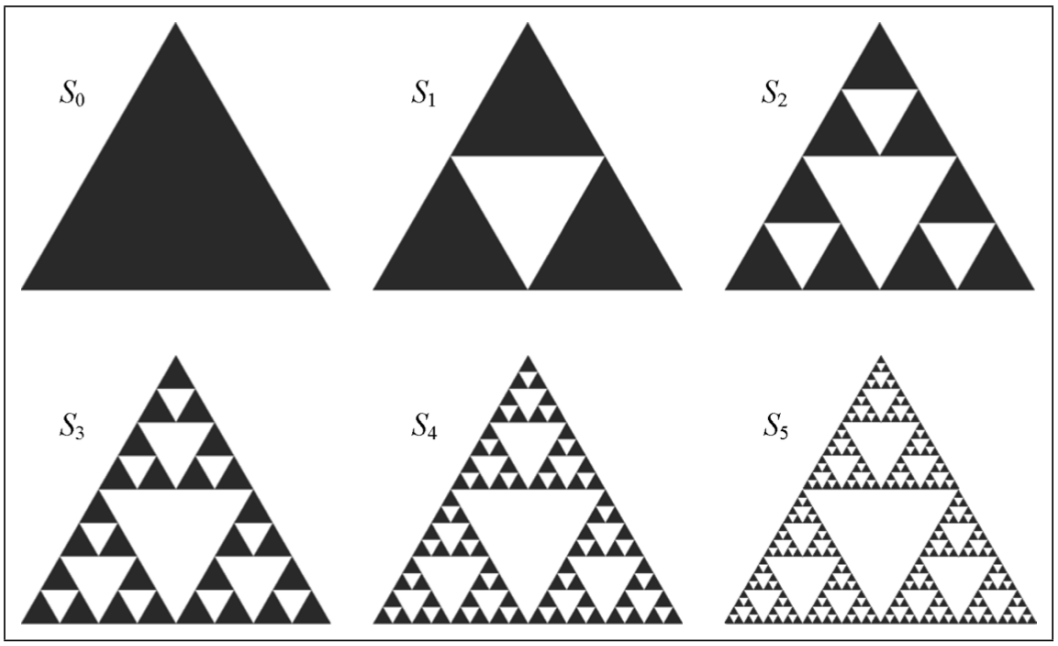
$$S = \lim_{k\to\infty}(f^k(S_0)). \tag{4.3}$$

Podemos ver que na k-ésimo etapa do processo iterativo, restam 3^k triângulos preenchidos. o perímetro de cada pequeno triângulo é metade do maior que gerou isso na etapa anterior. Temos, portanto, que o perímetro fracionário total de triângulos preenchidos em S_k é $(\frac{3}{2})^k$, que tende ao infinito com k. A área preenchida, por outro lado, diminui a cada etapa, o triângulo invertido removido representa precisamente $\frac{1}{4}$ da área maior, então a área preenchida fracionária de S_k é $(\frac{3}{4})^k$, que tende a zero quando k vai para o infinito. Assim, a forma limite S ocupa área zero no plano, é limitada, mas tem perímetro infinito, os detalhes podem ser observados na Figura 4.3.

Os três exemplos mencionados, apresentam algumas caraterísticas em comum.

i) Auto-semelhança em diferentes escalas. Isso significa que, ao aumentar ou diminuir a escala em que o fractal é observado, as mesmas características são encontradas em diferentes níveis de detalhe.

Figura 4.3: Triângulo de Sierpinski



Fonte: [8]

ii) Estrutura Fina: A estrutura fina se refere à sua complexidade e detalhamento em diferentes níveis de escala. Ao observar os três exemplos em uma escala maior, podemos notar a presença de várias estruturas menores que se repetem em diferentes níveis de detalhamento. Isso torna a estrutura do fractal fina, com muitos detalhes em diferentes escalas.

iii) Irregularidade: A irregularidade pode ser explicada pela geometria não-euclidiana. A geometria euclidiana, que é a geometria clássica, descreve objetos que têm formas regulares, como triângulos, círculos e quadrados. No entanto, os exemplos não podem ser descritos pela geometria euclidiana, pois sua forma é altamente irregular e complexa, com bordas irregulares e formas variadas.

iv) Finito: Vimos que os três exemplos, são finitos, embora possam parecer infinitos. Têm aparência infinita, mas as suas estruturas são finitas e definidas por uma função. Cada iteração dessa função produz um novo nível de detalhamento no fractal, mas, eventualmente, o processo se estabiliza e o fractal é completo.

Portanto, nos tês exemplos, os fractais que vimos têm uma estrutura fina, são irregulares para serem explicados pela geometria euclidiana, têm a forma de auto-semelhança e são finitos devido à sua definição matemática. Tendo em conta os exemplos ilustrados, segue a definição de fractal segundo [13]

Definição 44. *(Fractal segundo Falconer): Um fractal F é um conjunto que apresenta muitas das seguintes propriedades:*

(i) F tem uma estrutura fina, ou seja, detalhes em escalas arbitrariamente pequenas.

(ii) F é muito irregular para ser descrito na linguagem geométrica euclidiana, tanto local e globalmente.

(iii) Frequentemente, F tem alguma forma de auto-semelhança, talvez aproximada ou estatística.

(iv) Normalmente, a "dimensão fractal" de F (definida de alguma forma) é menor do que sua dimensão topológica.

(v) Na maioria dos casos de interesse, F é definido de forma muito simples, talvez recursivamente.

Com base na definição de [13], pode-se observar que os exemplos mencionados anteriormente são fractais.

4.2 Sistemas dinâmicos Complexos

Nesta seção, apresentaremos alguns conceitos de sistemas dinâmicos complexos, onde o foco principal é o Conjunto de Julia, e definiremos o conjunto de Julia de uma função polinomial complexa de grau $n \geq 2$ como a fronteira da bacia de atração de um ponto fixo atrator de f. Definiremos também o conjunto de Julia preenchido, cuja fronteira é igual ao conjunto de Julia.

Definição 45. *(Sistema dinâmico): Um sistema dinâmico consiste de um conjunto X de estados possíveis, juntamente com uma regra que descreve como o sistema evolui ao longo do tempo. Quando o tempo assume valores naturais ou inteiros, o sistema dinâmico é dito discreto. Quando o tempo assume valores reais, o sistema dinâmico é dito contínuo.*

Definição 46. *Dado um conjunto X e uma função $f : X \to X$, a k- ésima iterada de f é definida por*
$f^0(x) = x$,
$f^k(x) = f(f^{k-1}(x))$, *para* $k \geq 1$

Definição 47. *O comportamento de qualquer ponto $x \in \mathbb{R}$ sob iteração de f é dado pela sequência $x, f(x), f(f(x)), f(f(f(x))), ...$, que são chamados de órbita de x. Denotamos os pontos dessa orbita como $x, f(x), f^2(x), f^3(x)$ e assim sucessivamente.*

Definição 48. *Dada $f : X \to X$ e $x_0 \in \mathbb{C}$. A órbita de x_0 por f é limitada se existir $M > 0$, tal que $|f^k(x_0)| \leq M$, para todo $k \in \mathbb{N}$.*

Abordaremos sistemas dinâmicos discretos onde f é uma função complexa. Especificamente função polinomial complexa de grau $n \geq 2$ com coeficientes complexos, ou seja, $f : \mathbb{C} \to \mathbb{C}$ dada por

$$f(z) = a_n z^n + a_{n-1} z^{n-1} + ... + a_2 z^2 + a_1 z + a_0. \tag{4.4}$$

Definição 49. *Sejam $z \in \mathbb{C}$ e f uma função polinomial complexa. Dizemos que z é um ponto fixo de f se $f(z) = z$.*

Definição 50. *Sejam $z \in \mathbb{C}$ e f uma função polinomial complexa. Dizemos que z é um ponto periódico de f se existir um inteiro $p \geq 1$, tal que $f^p(z) = z$. O menor inteiro positivo p, tal que $f^p(z) = z$ é chamado de período de z*

Definição 51. *Sejam z um ponto fixo de f e $\lambda = f'(z)$. O ponto z é denominado:*
(a) Superatrator se $|\lambda| = 0$;
(b) Atrator se $0 \leq |\lambda| < 1$;
(c) Indiferente se $|\lambda| = 1$;
(d) Repulsor se $|\lambda| > 1$.

Definição 52. *(Bacia de atração): Seja w um ponto fixo atrator de f. A bacia de atração de w, denotada por $A_f(w)$, é o conjunto de todos os pontos cujas órbitas tendem à w, ou seja, $A_f(w) = \{z \in \mathbb{C} : f^k(z) \to w$ quando $k \to \infty\}$. A bacia de atração do infinito, $A_f(\infty)$, é definida do mesmo modo, ou seja, é definida por $A_f(\infty) = \{z \in \mathbb{C} : f^k(z) \to \infty$ quando $k \to \infty\}$.*

Exemplo 23. *Como exemplo, vamos considerar a função $f(z) = z^2$, e vamos determinar as bacias de atração. A função $f(z) = z^2$ é uma função polinomial que leva um número complexo z ao seu quadrado. Para calcular os pontos fixos da função, devemos encontrar os valores de z que satisfazem a equação $f(z) = z$. Substituindo $f(z) = z^2$ na equação acima, obtemos:*

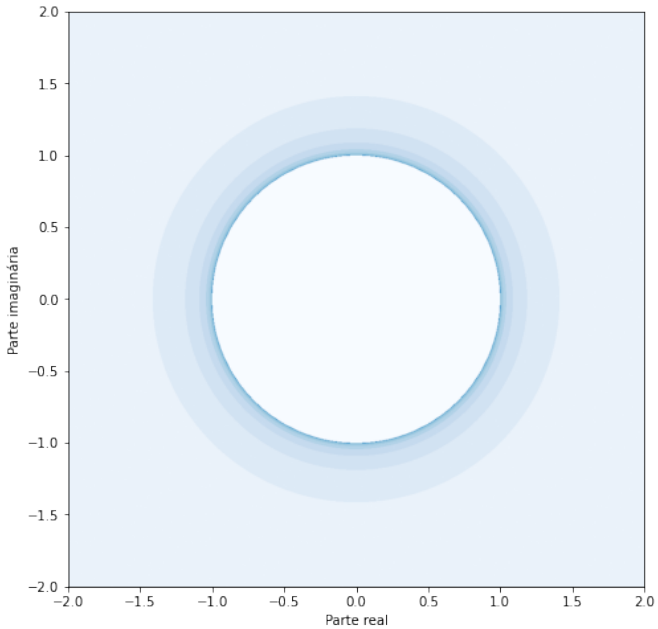
$$z^2 = z. \tag{4.5}$$

Isso implica que $z^2 - z = 0$, o que pode ser fatorado como $z(z-1) = 0$. Portanto, os pontos fixos da função são $z = 0$ e $z = 1$. Agora, para calcular as bacias de atração da função $f(z) = z^2$, precisamos examinar o comportamento das órbitas de pontos distintos no plano complexo sob a aplicação da função. A órbita de um ponto z_0 é a sequência de pontos gerada pela repetição da aplicação da função: $z_0, z_1 = f(z_0), z_2 = f(f(z_0)), z_3 = f(f(f(z_0)))$, e assim por diante. Vamos analisar cada ponto fixo da função $f(z) = z^2$ separadamente:

- *O ponto fixo $z = 0$: Se $|z_0| < 1$, então a sequência $z_0, z_1 = z_0^2, z_2 = (z_0^2)^2, \ldots$ converge para 0. Caso contrário, a sequência diverge para o infinito. Portanto, a bacia de atração para $z = 0$ é o disco unitário aberto $B(0,1) = z \in \mathbb{C} : |z| < 1$;*
- *O ponto fixo $z = 1$: Se $z_0 \neq -1$, então a sequência $z_0, z_1 = z_0^2, z_2 = (z_0^2)^2, \ldots$ converge para 1. Caso contrário, a sequência oscila indefinidamente entre 1 e -1. Portanto, a bacia de atração para $z = 1$ é o complemento do disco unitário fechado $\bar{B}(0,1) = z \in \mathbb{C} : |z| \leq 1$, ou seja, o conjunto $\mathbb{C} \setminus \bar{B}(0,1) = z \in \mathbb{C} : |z| > 1 \cup \infty$.*

Podemos visualizar essas bacias de atração no plano complexo colorindo o plano de acordo com a convergência das órbitas de pontos individuais. O resultado é mostrado na Figura 4.4, onde a região colorida em azul é a bacia de atração para $z = 1$ e a região colorida em branco é a bacia de atração para $z = 0$.

Figura 4.4: Bacia de Atração



Fonte: Autor

Teorema 13. *(Ponto fixo atrator): Sejam $A \subset \mathbb{C}$ e w um ponto fixo atrator da função $f : A \to A$. Então existe uma bola aberta de centro em w e raio $\delta > 0$(arbitráriamente pequeno), na qual a seguinte condição é satisfeita: Se $z \in B_\delta(w) \cap A$, então $f^k(z) \in B_\delta(w) \cap A$ e $f^k(z) \to w$ quando $k \to \infty$.*

Demonstração. Como $|f'(w)| < 1$, vamos tomar $r = \dfrac{1 + |f'(w)|}{2}$. Desta forma, $|f'(w)| < r < 1$. Por definição, $f'(w) = \lim\limits_{z \to w} \dfrac{f(z) - f(w)}{z - w}$. Portanto, $\forall\ \epsilon > 0$, $\exists \delta > 0$ tal que se $0 < |z - w| < \delta$ então $\left| \dfrac{f(z) - f(w)}{z - w} - f'(w) \right| < \epsilon$.

Pela desigualdade triangular temos que:
$\left| \dfrac{f(z) - f(w)}{z - w} \right| - |f'(w)| \leq \left| \dfrac{f(z) - f(w)}{z - w} - f'(w) \right| < \epsilon$, logo, $\left| \dfrac{f(z) - f(w)}{z - w} \right| < \epsilon + |f'(w)|$.

Tomemos $\epsilon = r - |f'(w)| > 0$. Com isto, teremos que:
$\left| \dfrac{f(z) - f(w)}{z - w} \right| < r$, $\forall\ z \in A \setminus \{w\}$ com $|z - w| < \delta$.
Consequentemente, para $z \in A$ com $|z - w| < \delta$, temos

$$|f(z) - w| \leq r|z - w| < r\delta, \text{ onde}, 0 < r < 1. \tag{4.6}$$

Isto significa que $f(z)$ está mais próximo de w do que z. Portanto, $f(z) \in B_\delta \cap A$. Podemos observar que que $f^2(z) = f(f(z))$, com $f(z) \in B_\delta(w) \cap A$. Aplicando a desigualdade (4.6) duas vezes teremos

$$|f^2(z) - w| = |f(f(z)) - w| \leq r|f(z) - w| \leq r.r|z - w| < r^2\delta,$$

onde $0 < r < 1$. Logo, $f^2(z) \in B_\delta(w) \cap A$. Por indução em k, para $z \in A$ com $|z - w| < \delta$, temos que $|f^k(z) - w| < r^r\delta$, onde $0 < r < 1$. Portanto, $f^k(z) \in B_\delta(w) \cap A$. Quando $k \to \infty$, $r^k \to 0$, e consequentemente, $f^k(z) \to w$

■

O **Teorema 13** não vale para qualquer função f. A condição para que o teorema seja válido é que w seja um ponto fixo atrator de f em A, ou seja, $f(w) = w$ e a sequência $\{f^k(z)\}_{k=0}^{\infty}$ converge para w para todo $z \in A$. Por exemplo, considere a função $f(z) = z+1$ em $A \subseteq \mathbb{C}$. Nesse caso, não existe ponto fixo atrator para a função f, pois a sequência $\{f^k(z)\}_{k=0}^{\infty}$ não converge para nenhum ponto em $\mathbb{C}$ para todo $z \in \mathbb{C}$. Portanto, o teorema não se aplica nesse caso.

Teorema 14. *(Ponto fixo Repulsor) Sejam $A \subset \mathbb{C}$ e w um ponto fixo repulsor da função $f : A \to A$. Então existe uma bola aberta de centro em w e raio δ na qual a seguinte condição é satisfeita: Se $z \in B_\delta(w) \cap A$ e $z \neq w$, então existe $n_0 \in \mathbb{N} \setminus \{0\}$, tal que $f^n(z) \notin B_\delta(w) \cap A$, para todo $n \geq n_0$.*

Demonstração. Como w é um ponto fixo repulsor de f, existe uma vizinhança U de w tal que $|f'(w)| > 1$ e $|f(z) - w| < |z - w|$ para todo $z \in U$ diferente de w. Escolha $\delta > 0$ tal que $B_\delta(w) \subset U$. Suponha, por contradição, que existe $z \in B_\delta(w) \cap A$ e uma sequência (n_k) tal que $f^{n_k}(z) \in B_\delta(w) \cap A$ para todo k. Como $z \in B_\delta(w)$, temos que $|z - w| < \delta$, e como $f^{n_k}(z) \in B_\delta(w)$, temos que $|f^{n_k}(z) - w| < \delta$. Pela desigualdade triangular, temos que

$$|z - w| \leq |z - f^{n_k}(z)| + |f^{n_k}(z) - w| \tag{4.7}$$

Usando a desigualdade $|f(z) - w| < |z - w|$ e iterando n_k vezes, obtemos

$$|z - w| \leq \sum_{j=0}^{n_k-1} |f^j(z) - f^{j+1}(z)| + |f^{n_k}(z) - w| < \sum_{j=0}^{n_k-1} |f^j(z) - f^{j+1}(z)| + \delta \tag{4.8}$$

Usando a desigualdade triangular novamente, temos que

$$|f^j(z) - f^{j+1}(z)| \leq |f^j(z) - w| + |w - f^{j+1}(z)| < 2|z - w| \tag{4.9}$$

para todo $j \in \mathbb{N}$. Substituindo a equação (4.9) na equação (4.8), obtemos

$$|z - w| < 2n_k|z - w| + \delta, \tag{4.10}$$

o que implica que $n_k > \frac{1}{2}|z - w|^{-1}(\delta - |z - w|)$. Como $\delta > |z - w|$, isso implica que n_k cresce indefinidamente, o que contradiz o fato de w ser um ponto repulsor de f. Portanto, a suposição inicial está errada, e o teorema é verdadeiro. ■

Proposição 5. *$A_f(w)$ é um conjunto aberto*

Demonstração. Como w é um ponto fixo atrator de f, pelo Teorema do ponto Fixo Atrator (**Teorema 13**) , existe um conjunto aberto V contendo w, tal que $V \subset A_f(w)$ (Se $w = \infty$, podemos tomar $\{z : |z| > r\}$, para r suficientemente grande). Isto implica que $A_f(w)$ é aberto. Com efeito, se $z \in A_f(w)$, então $f^k(z) \in V$ para algum k, logo $z \in f^{-k}(V)$, o qual é aberto. ■

4.2.1 Conjunto de Julia

O conjunto de Julia surgiu com o estudo do comportamento de iterações de funções complexas. Este conceito foi introduzido pelo matemático francês Gaston Maurice Julia (1893-1978), publicado na sua obra prima "Mémoire Sur I´iteration des fonctions rationnelles".

Definição 53. *(Conjunto de Julia): Seja f uma função polinomial complexa de grau $n \geq 2$ e w um ponto fixo atrator de f. O Conjunto de Julia de f denotado por $J(f)$, é a fronteira da bacia de atração de w.*

A **Definição 53** descreve o Conjunto de Julia de uma função polinomial complexa f de grau $n \geq 2$, com um ponto fixo atrator w. O Conjunto de Julia, representado por $J(f)$, é a fronteira da bacia de atração de w, que consiste nos pontos que tendem a convergir para w ao serem iterados por f infinitamente. A fronteira da bacia de atração é formada pelos pontos limites das órbitas que convergem para w. O Conjunto de Julia exibe formas complexas e fractais, mesmo para funções aparentemente simples.

Definição 54. *(Conjunto de Fatou): O complementar do Conjunto de Julia é chamado de Conjunto de Fatou, representado por $F(f)$, e consiste nos pontos que convergem para algum ponto atrator sob a ação de f.*

O Conjunto de Fatou é composto pelos pontos que pertencem à bacia de atração de um ponto fixo atrator de f.

Exemplo 24. *(Implementação do conjunto de Julia pelo MATLAB). A implementação do conjunto de Julia no MATLAB, começa definindo a função iterativa $f(z, c) = z^2 + c$, como uma função anônima usando a sintaxe @(z, c). Em seguida, definimos as dimensões da imagem, a largura e altura da imagem que serão geradas. Também definimos os limites*

do eixo x e y. Esses limites definem a região do plano complexo que será representada na imagem. Depois, criamos uma matriz de pontos no plano complexo usando a função meshgrid. Linspace é usada para criar pontos igualmente espaçados dentro dos limites x e y definidos. Em seguida, definimos o número máximo de iterações (max_iter) e o número complexo constante c. Nesse caso, $c = a + bi$ é um valor arbitrário. Então, a matriz de cores é inicializada com zeros. Para cada ponto na matriz, é feita uma iteração da função iterativa até que a magnitude do ponto exceda o limite de 2 ou o número máximo de iterações seja atingido. A cor do ponto é determinada pelo número de iterações necessárias para que a magnitude do ponto exceda o limite de 2. Os pontos que não divergem após o número máximo de iterações são coloridos de preto. Por fim, o conjunto de Julia é plotado usando a função image. A cor do conjunto é definida usando a função colormap. Os eixos são desligados usando a função axis off. A seguir, o código genérico do algoritmo:

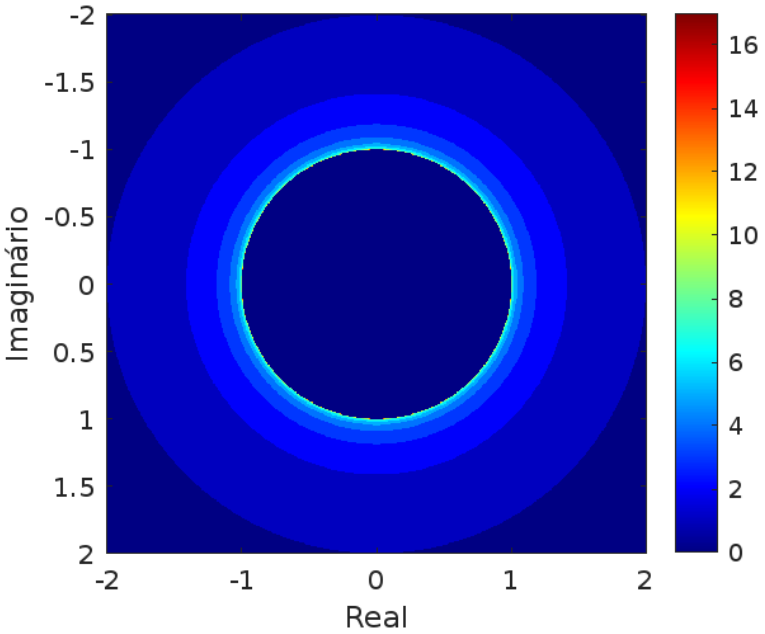
```
1
2  % Definindo a fun   o iterativa
3  f = @(z, c) z^2 + c;
4
5  % Definindo o plano complexo
6  w = 800; % largura da imagem
7  h = 800; % altura da imagem
8  x_min = -2;
9  x_max = 2;
10 y_min = -2;
11 y_max = 2;
12
13 % Criando uma malha de pontos no plano complexo
14 [X, Y] = meshgrid(linspace(x_min, x_max, w), linspace(y_min,
       y_max, h));
15
16 % Definindo o n mero m ximo de itera   es
17 max_iter = 100;
18
19 % Definindo o n mero complexo constante c
20 c = complex(a, b);
21
22 % Inicializando a matriz de cores
23 colors = zeros(h, w);
24
25 % Looping para cada ponto no plano complexo
26 for i = 1:h
27     for j = 1:w
28         z = complex(X(i, j), Y(i, j)); % convertendo o ponto
               em n mero complexo
29         iter = 0; % contador de itera   es
30
31         % Looping at   atingir o n mero m ximo de
               itera   es ou divergir
32         while (abs(z) < 2) && (iter < max_iter)
33             z = f(z, c); % aplicando a fun   o iterativa
```

```
34              iter = iter + 1; % atualizando o contador de
                    itera  es
35          end
36
37          % Colorindo o ponto de acordo com o n mero de
                itera  es necess rias
38          if iter == max_iter
39              colors(i, j) = 0; % preto
40          else
41              colors(i, j) = iter; % outra cor
42          end
43      end
44  end
45
46  % Plotando o conjunto de Julia
47  figure;
48  imagesc([x_min, x_max], [y_min, y_max], colors);
49  colormap(jet);
50  colorbar;
51  xlabel('Real');
52  ylabel('Imagin rio');
53  axis image;
```

Listing 4.1: Implementação do Conjunto de Julia no MATLAB

Exemplo 25. *Para o presente exemplo, vamos considerar a função* $f : \mathbb{C} \to \mathbb{C}$ *dada por* $f(z) = z^2 + c$*, onde* $c = 0$*. Assim sendo, a* ***Figura 4.5*** *ilustra o conjunto de Julia.*

Figura 4.5: Conjunto de Julia c=0

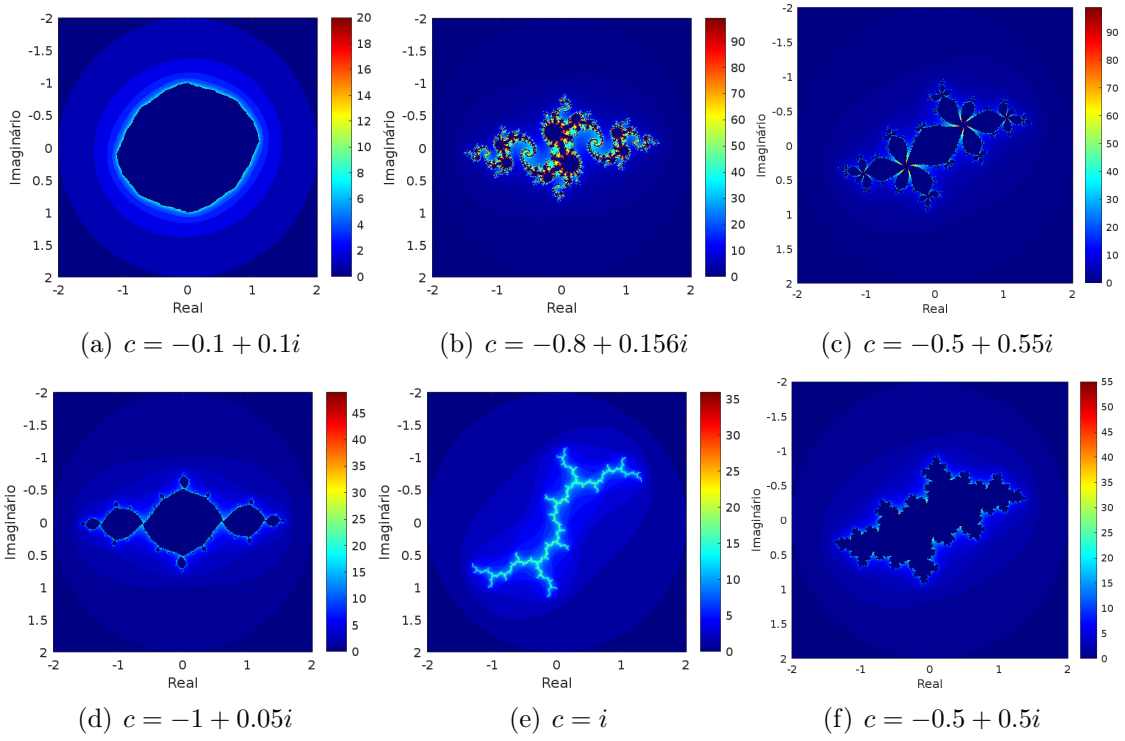


Fonte: Autor

Temos que o conjunto de Julia é a circunferência de centro em 0 e raio 1, ao passo que o conjunto de Fatou é a união disjunta entre o interior e o exterior desta circunferência,

como ilustrado na **Figura 4.5**. No **Exemplo 25**, temos um fractal que é um conjunto de Julia cuja constante $c = 0$. O conjunto de Julia gera uma família de subconjuntos denominados fractais de Julia, considerando a família de funções $f : \mathbb{C} \to \mathbb{C}$ dada por $f(z) = z^2 + c$, onde $z, c \in \mathbb{C}$. À medida que variamos o valor de c, podemos obter vários fractais de Julia. Na Figura 4.6, temos diversos exemplos de fractais de Julia obtidos ao variar o valor de c.

Figura 4.6: Fractais de Julia



(a) $c = -0.1 + 0.1i$ (b) $c = -0.8 + 0.156i$ (c) $c = -0.5 + 0.55i$

(d) $c = -1 + 0.05i$ (e) $c = i$ (f) $c = -0.5 + 0.5i$

Definição 55. *(Conjunto Prisioneiro). Seja f uma função polinomial complexa de grau $n \geq 2$. O conjunto prisioneiro de uma função f é definido como o conjunto de pontos no plano complexo que permanecem limitados, ou "presos", dentro de uma região finita do plano quando iterados sob a função f. Formalmente, o conjunto prisioneiro é dado por:*

$$K(f) = \left\{ z \in \mathbb{C} : |f^k(z)| \nrightarrow \infty, \text{ quando } k \to \infty \right\} \tag{4.11}$$

onde $f^k(z)$ representa a k-ésima iteração de z sob a função f.

Definição 56. *(Conjunto de Escape). Seja f uma função polinomial complexa de grau $n \geq 2$. O conjunto de escape é definido como o conjunto de pontos no plano complexo que divergem infinitamente com o aumento do número de iterações sob a função f. Formalmente, o conjunto de escape é dado por:*

$$E(f) = \left\{ z \in \mathbb{C} : |f^k(z)| \to \infty, \text{ quando } k \to \infty \right\} \tag{4.12}$$

onde $f^k(z)$ representa a k-ésima iteração de z sob a função f. Em outras palavras, o conjunto de escape é composto por pontos que se afastam indefinidamente do ponto de partida com o aumento do número de iterações sob a função f.

Definição 57. *(Conjunto de Julia Preenchido): Seja f uma função polinomial complexa de grau $n \geq 2$. o conjunto de Julia preenchido é um conjunto fractal que é obtido a partir do conjunto prisioneiro de uma função complexa f e é denotado por $K(f)$. O conjunto*

de Julia preenchido é definido como o conjunto de pontos no plano complexo que não pertencem ao conjunto de escape de f e que possuem vizinhanças que se comportam de forma semelhante, ou seja, que têm a mesma estrutura topológica e geométrica.

Formalmente, o conjunto de Julia preenchido é dado por:

$$K(f) = \overline{\{z \in \mathbb{C} \backslash E(f) : |f^k(z)| \leq M\}}, \text{ para todo, } k \in \mathbb{N}, \text{ e, algum, } M > 0 \tag{4.13}$$

onde $E(f)$ é o conjunto de escape de f e $f^k(z)$ representa a k-ésima iteração de z sob a função f.

Em outras palavras, o conjunto de Julia preenchido é composto pelos pontos que não divergem com o aumento do número de iterações sob a função f, e que estão rodeados por pontos que têm a mesma estrutura topológica e geométrica.

Observação 1. *Como $J(f) = \partial A_f(\infty) = \partial E(f) = \partial K(f)$, segue que o conjunto de Julia de f é a fronteira do conjunto de Julia preenchido.*

A **Observação 1** afirma que o conjunto de Julia de uma função complexa f é igual à fronteira do conjunto de Julia preenchido $K(f)$. Isto é, $J(f) = \partial K(f)$. Isso significa que os pontos que compõem a fronteira do conjunto de Julia preenchido são aqueles que possuem comportamento semelhante aos pontos do conjunto de Julia, mas que não pertencem a ele nem ao conjunto de escape da função f. Tendo em conta as definições 53 e 57, podemos afirmar:

Corolário 3. *O conjunto de Julia e o conjunto de Julia preenchido são fechados*

Demonstração. O conjunto de Julia é fechado por ser fronteira de um conjunto. Temos que $A_f(\infty)$ é aberto. Como $A_f(\infty) = E(f)$ e $K(f)$ são complementares, segue que o conjunto de Julia preenchido é fechado. ■

Exemplo 26. *Como exemplo, considerando a função $f : \mathbb{C} \to \mathbb{C}$ dada por $f(z) = z^2$, temos que $J(f) = \{z \in \mathbb{C} : |z| = 1\}$ e $K(f) = \{z \in \mathbb{C} : |z| \leq 1\}$*

Corolário 4. *O Conjunto de Julia é simétrico em relação à origem*

Demonstração. Devemos mostrar que $z \in J(f)$ se, e somente se, $-z \in J(f)$. Seja $z \in \mathbb{C}$. De $f_c^k(z) = f_c^{k-1}(f_c(z)) = f_c^{k-1}(z^2 + c) = f_c^{k-1}((-z)^2 + c) = f_c^{k-1}(f_c(-z)) = f_c^k(-z)$, segue que $f_c^k(z) \to \infty$ se, e somente se, $f_c^k(-z) \to \infty$. Logo, $z \in K(f_c)$ se, e somente se, $-z \in K(f_c)$. Potanto, o conjunto de Julia preenchido de f_c e, consequentemente, sua fronteira, $J(f)$, são simétricos em relação à origem. ■

Definição 58. *(Conjunto de Mandelbrot): Seja* **F** *a família das funções quadráticas da forma $f_c : \mathbb{C} \to \mathbb{C}$ dadas por $f_c(z) = z^2 + c$, onde c é uma constante complexa. O conjunto de Mandelbrot* **M** *é o conjunto dos parâmetros c tais que a órbita do ponto crítico de f_c é limitada, pode ser escrito como:*

$$\mathbf{M} = \{c \in \mathbb{C} : \{f_c^k(0)\}_{k \geq 1} \text{ é limitado}\}$$

Lema 2. *Sejam $z \in \mathbb{C}$ e $f_c(z) = z^2 + c$ onde c é uma constante complexa. Se $|z| \geq |c|$ e $|z| > 2$, então existe numero real positivo ϵ, tal que $|f_c^n(z)| \geq (1 + \epsilon)^n |z|$.*

Demonstração. [24], página 15. ■

Proposição 6. *Sejam $z \in \mathbb{C}$ e $f_c(z) = z^2 + c$, onde c é uma constante complexa. Se $|z| \geq |c|$ e $|z| > 2$, então $\lim_{n \to \infty} |f_c^n(z)| = \infty$ (ou seja, a órbita de z tende ao infinito)*

Demonstração. De acordo com o **Lema 2**, existe um número real positivo ϵ tal que $|f_c^n(z)| \geq (1+\epsilon)^n|z|$. Como $\lim_{n\to\infty}(1+\epsilon)^n = \infty$, teremos

$$\lim_{n\to\infty}|f_c^n(z)| \geq \lim_{n\to\infty}(1+\epsilon)^n|z| = |z| \lim_{n\to\infty}(1+\epsilon)^n = \infty$$

Portanto, a órbita de z tende a infinito se $|z|$ e $|z| > 2$. ■

Corolário 5. *O conjunto de Julia f_c está contido no círculo de centro 0 e raio $r_c = \max\{|c|, 2\}$*

Demonstração. Para demonstrar que o conjunto de Julia f_c está contido no círculo de centro 0 e raio $r_c = \max\{|c|, 2\}$, precisamos verificar se cada ponto no conjunto de Julia satisfaz a condição de pertencer a esse círculo. Tendo em conta que o conjunto de Julia f_c é um conjunto formado por pontos do plano complexo que não tendem ao infinito sob iterações sucessivas da função $f_c(z) = z^2+c$, onde c é uma constante complexa, assumindo que um ponto z pertence ao conjunto de Julia, significa que ele não tende a infinito sob iterações sucessivas de f_c. Assim sendo, podemos escrever:

$$|f_c(z)| \leq R, \tag{4.14}$$

onde R é algum número real positivo. Observe que, se $|z|$ for maior que R, então $|z|^2 > R|z|$ e, portanto,

$$|f_c(z)| = |z|^2 + |c| \geq |z|^2 - |c| > R|z| - |c|$$

Usando a desigualdade triangular para $|z|^2 + |c|$, temos:

$$|f_c(z)| \geq |z|^2 - |c| > R|z| - |c|$$

Para que um ponto z esteja contido no círculo de centro 0 e raio $r_c = \max\{|c|, 2\}$, precisamos ter $|z| \leq r_c$. Então, se escolhermos $R = r_c$, obtemos:

$$|f_c(z)| \geq |z|^2 - |c| > R|z| - |c| \geq r_c|z| - |c| \tag{4.15}$$

A partir de (4.15), podemos concluir que, se $|z| > r_c$, então $|f_c(z)| > r_c|z| - |c|$, e portanto, z não pertence ao conjunto de Julia. Portanto, todos os pontos no conjunto de Julia estão contidos no círculo de centro 0 e raio $r_c = \max\{|c|, 2\}$. ■

4.2.2 Algumas Propriedades Topológicas do Conjunto de Julia

Nesta subsecção veremos algumas propriedades importantes do conjunto de Julia, tais propriedades são de capital importância para o nosso trabalho. Podemos definir o conjunto de Julia de diversas formas equivalentes, a seguir veremos a seguinte definição equivalente à **Definição 53**.

Definição 59. *O conjunto de Julia $J(f)$ é o fêcho do conjunto dos pontos periódicos repulsores de f. O complementar do conjunto de Julia é denominado conjunto de Fatou $F(f)$*

Exemplo 27. *Seja $f : \mathbb{C} \to \mathbb{C}$ definida por $f(z) = z^2$. Logo, $f^k(z) = z^{2^k}$ e $|(f^k)'(z)| = 2^k|z|^{2^k-1}$. Os pontos periódicos de f satisfazem $f^p(z) = z$, ou seja, $z^{2p} = z$ com $p \in \mathbb{N} \setminus \{0\}$. Se $z = 0$, então $|(f^p)'(0)| = 0$. Logo, $z = 0$ é um ponto fixo atrator, consequentemente, não pertence ao conjunto de Julia. Se $z \neq 0$, podemos escrever $z^{2^p-1} = 1$, de onde segue que z é uma raiz a unidade e os pontos periódicos não nulos de f são*

$\{cos(\frac{2k\pi}{2^p-1}) + isen(\frac{2k\pi}{2^p-1}); 0 \leq k \leq 2^p-2\} = \{e^{i.\frac{2k\pi}{2^p-1}}; 0 \leq k \leq 2^p-2\}$. *Como* $|z| = 1$, *temos que* $|(f^p)'(z)| = 2^p > 1$. *Logo, as* $(2^p - 1)$ *raízes da unidade são pontos periódicos repulsores de* f. *Quando* $p \to \infty$, *obtemos um conjunto denso de pontos periódicos repulsores em* $C = \{z \in \mathbb{C}; |z| < 1 \text{ ou } |z| > 1\}$.

Na definição do conjunto de Julia $J(f)$, usamos a função f para definir esse conjunto. Por isso, é comum usar a notação $J(f)$ para representar o conjunto de Julia associado à função f. No entanto, quando a função f é clara ou não é importante no contexto, podemos simplesmente usar a notação J para representar o conjunto de Julia. No **Exemplo 27**, temos que $f(J) = J$ e $f^{-1}(J) = J$, ou seja, o conjunto de Julia é invariante sob a ação da função f e de sua inversa. Isso significa que, se um ponto pertence ao conjunto de Julia J, então sua imagem sob a função f também pertence a J, e o mesmo ocorre para a pré-imagem sob f^{-1}. Como J é invariante sob a ação de f e f^{-1}, temos que $J = f(J) = f^{-1}(J)$. Dessa forma, podemos usar a notação J para representar o conjunto de Julia neste exemplo, já que a função f é clara e sua ação no conjunto de Julia é invariante.

Definição 60. *(Função Analítica Complexa.) Uma função* $f(z)$ *é dita analítica em um domínio* D *do plano complexo se ela é diferenciável em todo ponto de* D. *Isso significa que, para cada ponto* $z_0 \in D$, *a derivada de* $f(z)$ *existe em* z_0 *e em todos os pontos suficientemente próximos a* z_0.

Definição 61. *(Família Normal). Seja* U *um conjunto aberto em* $\mathbb{C}$, *e seja* $f_k : U \to \mathbb{C}$ *uma família de funções analíticas complexas. Dizemos que a família* $\{f_k\}$ *é normal em* U *se toda sequência de funções selecionadas de* $\{f_k\}$ *possui uma subsequência que converge uniformemente em cada subconjunto compacto de* U, *ou para uma função analítica limitada ou para* ∞. *A família* $\{f_k\}$ *é normal no ponto* w *de* U *se existe algum subconjunto aberto* V *de* U *contendo* w, *tal que* $\{f_k\}$ *é uma família normal em* V.

A **Definição 61** significa que, se escolhermos qualquer sequência de funções da família $\{f_k\}$, podemos sempre extrair uma subsequência que converge uniformemente em subconjuntos compactos de U. Em outras palavras, a família f_k é "bem definida"em relação à convergência uniforme em subconjuntos compactos de U. Além disso, a definição estabelece o conceito de normalidade em um ponto w de U. Nesse caso, a família $\{f_k\}$ é normal no ponto w se existe um subconjunto aberto V de U contendo w tal que a família é normal em V. Isso significa que, em torno de um ponto w de U, a família f_k é "bem definida"em relação à convergência uniforme em subconjuntos compactos de V.

Teorema 15. *(Teorema de Montel): Seja* $\{f_n\}$ *uma família de funções analíticas complexas definidas num domínio* U. *Suponha que existem os números* $a, b \in \mathbb{C}$, $a \neq b$, *tal que* $f_n(z) \neq a$ *ou* b *para algum* n *e algum* $z \in U$. *Então* $\{f_n\}$ *é uma família normal em* U.

O **Teorema 15** significa que toda sequência de funções selecionadas de f_n possui uma subsequência que converge uniformemente em cada subconjunto compacto de U, ou para uma função analítica limitada ou para ∞. É muito importante destacar aqui a importância do teorema de Montel para qualquer função analitica. A consequência deste teorema segue o **Corolário 6** que é a base para o método de Newton-Raphson de natureza fractal que será estudado no **Capítulo 5**.

Definição 62. *Seja* K *um conjunto compacto de* $\mathbb{C}$ *e seja* $\{f_n\}$ *uma família de funções analíticas complexas definidas em* K. $\{f_n\}$ *diz-se equicontínua se Para todo* $\epsilon > 0$ *existe um* $\delta > 0$ *tal que, para todo* n *e para todo* $z, w \in K$ *com* $|z-w| < \delta$, *temos* $|f_n(z)-f_n(w)| < \epsilon$. *Em outras palavras, as funções da família* $\{f_n\}$ *variam suavemente em* K.

Definição 63. *Seja K um conjunto compacto de $\mathbb{C}$ e seja $\{f_n\}$ uma família de funções analíticas complexas definidas em K. $\{f_n\}$ diz-se uniformemente limitada se existe uma constante $M > 0$ tal que para todo n e todo $z \in K$, temos $|f_n(z)| \leq M$. Em outras palavras, as funções da família $\{f_n\}$ são limitadas uniformemente em K. Essa propriedade é importante para provar que uma sequência de funções analíticas converge uniformemente em subconjuntos compactos de U.*

Teorema 16. *(Teorema de Arzelà-Ascoli). Seja K um conjunto compacto de $\mathbb{C}$ e seja $\{f_n\}$ uma família de funções analíticas complexas definidas em K. Então, as seguintes afirmações são equivalentes:*

1. *$\{f_n\}$ é equicontínua e uniformemente limitada em K;*
2. *$\{f_n\}$ possui uma subsequência que converge uniformemente em K para uma função analítica complexa definida em K.*

Demonstração. Primeiro, vamos mostrar que (2) implica (1). Seja $\{f_{n_k}\}$ uma subsequência de $\{f_n\}$ que converge uniformemente em K para uma função analítica complexa f. Então, para todo $\epsilon > 0$, existe $N \in \mathbb{N}$ tal que $|f_{n_k}(z) - f(z)| < \epsilon$ para todo $z \in K$ e todo $k \geq N$. Em particular, temos que $|f_{n_k}(z) - f_{n_k}(w)| < \epsilon$ para todo $z, w \in K$ e todo $k \geq N$. Portanto, $\{f_n\}$ é equicontínua em K. Além disso, como $\{f_{n_k}\}$ converge uniformemente em K para f, então para todo $\epsilon > 0$ existe $N \in \mathbb{N}$ tal que $|f_{n_k}(z) - f(z)| < \epsilon$ para todo $z \in K$ e todo $k \geq N$. Como $\{f_n\}$ é uniformemente limitada em K, existe $M > 0$ tal que $|f_n(z)| < M$ para todo $z \in K$ e todo $n \in \mathbb{N}$. Então, para todo $z \in K$ e todo $k \geq N$, temos:

$$|f(z)| \leq |f(z) - f_{n_k}(z)| + |f_{n_k}(z)| \ < \epsilon + M.$$

Portanto, $\{f_n\}$ é uniformemente limitada em K. Agora, vamos mostrar que (1) implica (2). Pela equicontinuidade de $\{f_n\}$ em K, para todo $\epsilon > 0$, existe $\delta > 0$ tal que $|f_n(z) - f_n(w)| < \epsilon$ para todo $z, w \in K$ com $|z - w| < \delta$ e todo $n \in \mathbb{N}$. Como K é compacto, podemos escolher um conjunto finito de pontos $z_1, \ldots, z_m \in K$ tal que $K \subseteq \bigcup_{j=1}^{m} B(z_j, \delta/2)$, onde $B(z_j, \delta/2)$ é a bola aberta de centro z_j e raio $\delta/2$. Como f_n é uniformemente limitada em K, existe $M > 0$ tal que $|f_n(z)| < M$ para todo $z \in K$ e todo $n \in \mathbb{N}$. Então, para cada $j = 1, \ldots, m$ e cada $n \in \mathbb{N}$, podemos escrever a função f_n na forma de sua série de Taylor centrada em z_j:

$$f_n(z) = \sum_{k=0}^{\infty} c_k^{(n)} (z - z_j)^k,$$

onde $c_k^{(n)} = \frac{f_n^{(k)}(z_j)}{k!}$ é o coeficiente de Taylor de f_n de ordem k centrado em z_j. Como $K \subseteq \bigcup_{j=1}^{m} B(z_j, \delta/2)$, temos que para todo $z \in K$ existe $j \in 1, \ldots, m$ tal que $z \in B(z_j, \delta/2)$. Vamos fixar um ponto $z \in K$ e escolher j de forma que $z \in B(z_j, \delta/2)$. Então, para todo $n \in \mathbb{N}$ e todo $k \in \mathbb{N}$, temos:

$$|c_k^{(n)}| = \left| \frac{f_n^{(k)}(z_j)}{k!} \right| = \frac{|f_n^{(k)}(z_j)|}{k!} \leq \frac{M}{(\delta/2)^k},$$

usando a desigualdade de Cauchy para derivadas. Como a soma $\sum_{k=0}^{\infty} \frac{M}{(\delta/2)^k}$ converge, então a família $c_k^{(n)}{}_{n \in \mathbb{N}}$ é uniformemente limitada, ou seja, existe $B > 0$ tal que $|c_k^{(n)}| < B$

para todo $n \in \mathbb{N}$ e todo $k \in \mathbb{N}$. Portanto, a família de funções $\{f_n\}$ é uniformemente limitada em cada bola $B(z_j, \delta/2)$. Agora, para cada $j = 1, \ldots, m$, considere a família $\{f_n|\{B(z_j, \delta/2)\}\}$, que é uma família de funções analíticas definidas em $B(z_j, \delta/2)$. Como $\{f_n\}$ é equicontínua em K, então $\{f_n|\{B(z_j, \delta/2)\}\}$ é equicontínua em $B(z_j, \delta/2)$. Usando o teorema de Cauchy para funções analíticas, sabemos que $\{f_n|\{B(z_j, \delta/2)\}\}$ é uniformemente limitada em $B(z_j, \delta/2)$. Portanto, aplicando o lema de Arzelà-Ascoli (que é uma versão mais fraca do teorema de Arzelà-Ascoli), concluímos que $\{f_n|\{B(z_j, \delta/2)\}\}$ possui uma subsequência que converge uniformemente em $B(z_j, \delta/2)$ para uma função analítica f_j ou para ∞. Para cada $j = 1, \ldots, m$, denotamos por f a função analítica em U dada por

$$f(z) = \begin{cases} f_j(z), & \text{se } z \in B(z_j, \delta/2) \\ & 0, \text{ se } z \in U \setminus \bigcup_{j=1}^{m} B(z_j, \delta/2). \end{cases}$$

Então, f é uma função analítica em U e, para cada $j = 1, \ldots, m$, a sequência $\{f_n|_{B(z_j,\delta/2)}\}$ converge uniformemente para f_j, que é igual a f em $B(z_j, \delta/2)$. Portanto, $\{f_n\}$ converge uniformemente em cada subconjunto compacto de U para a função f ou para ∞. Isso implica que $\{f_n\}$ é uma família normal em U, o que conclui a prova. ■
Assim, podemos demonstrar o Teorema de Montel, pelo que, segue a demonstração.

Demonstração. (Teorema de Montel). Suponha que $\{f_n\}$ é uma família de funções analíticas complexas definidas em um domínio U, e que existem os números $a, b \in \mathbb{C}$, $a \neq b$, tais que $f_n(z) \neq a$ ou b para algum n e algum $z \in U$. Seja K um subconjunto compacto de U. Precisamos mostrar que toda sequência de funções selecionadas de $\{f_n\}$ possui uma subsequência que converge uniformemente em K. Para isso, consideremos a sequência de funções $\{f_n|_K\}$, que é uma sequência de funções analíticas complexas definidas em K. Como K é um subconjunto compacto de U, então $\{f_n|_K\}$ é uma família de funções analíticas complexas definidas em um conjunto compacto e, portanto, é uma família equicontínua. Pelo Teorema de Arzelà-Ascoli, toda família equicontínua de funções analíticas complexas definidas em um conjunto compacto é uma família normal. Logo, $\{f_n|_K\}$ é uma família normal em K, o que implica que toda sequência de funções selecionadas de $\{f_n|_K\}$ possui uma subsequência que converge uniformemente em K. Portanto, toda sequência de funções selecionadas de $\{f_n\}$ possui uma subsequência que converge uniformemente em cada subconjunto compacto de U. Portanto, $\{f_n\}$ é uma família normal em U, como afirmado no enunciado do teorema. ■

Corolário 6. *Dada f um mapa analítico. Seja z_0 pertencente a $J(f)$ e seja U uma vizinhança de z_0. Então as iterações $f^n(U)$ omitem, no máximo, um ponto em $\mathbb{C}$.*

Demonstração. Suponha, por contradição, que $f^n(U)$ omite dois pontos distintos z_1 e z_2. Considere a função auxiliar $g : \mathbb{C} \to \mathbb{C}$ dada por $g(z) = \frac{z-z_1}{z-z_2}$. Observe que $g(z_1) = 0$, $g(z_2) = \infty$ e $g(z) \neq 1$ para todo $z \in \mathbb{C}$. Agora, fixe $\epsilon > 0$ e escolha n suficientemente grande de modo que $f^n(U)$ esteja contido em um disco $D(z_0, r)$ com $z_0 \in U$ e $r < \frac{\epsilon}{2}$. Como $z_1, z_2 \notin f^n(U)$, temos que $f^n(U)$ está contido em $\mathbb{C} \setminus z_1, z_2$, e portanto $g \circ f^n$ é uma função analítica em U. Por outro lado, a hipótese de que $z_0 \in J(f)$ implica que f^n não é uma função de ramificação em z_0, significa que para cada z_0 escolhido, há pelo menos um ponto $z \neq z_0$ tal que $f^n(z) = f^n(z_0)$, ou seja, a função f^n não é injetiva em uma vizinhança de z_0. Assim, podemos aplicar o Teorema de Montel para a família $g \circ f^n$ em U. Em particular, temos que $g \circ f^n$ é equicontínua e uniformemente limitada em U. Isso significa que, para todo $\delta > 0$, existe um $M > 0$ tal que $|g(f^n(z)) - g(f^n(w))| \leq M|f^n(z) - f^n(w)|$ para todo $z, w \in U$ com $|z - w| < \delta$. Tomando $\delta = \frac{\epsilon}{2M}$, obtemos que $|g(f^n(z)) - g(f^n(w))| < \frac{\epsilon}{2}$ para todo $z, w \in U$ com $|z - w| < \delta$. Em outras palavras, $g \circ f^n$ é equicontínua em U. Mas isso é uma contradição, pois a única função analítica e equicontínua em U é a função

constante, e $g \circ f^n$ não é constante, pois $g(z_1) = 0$ e $g(z_2) = \infty$. Portanto, concluímos que as iterações $f^n(U)$ omitem, no máximo, um ponto em $\mathbb{C}$. ■

No **Corolário 6** , as iterações $f^n(U)$ omitem, no máximo, um ponto em $\mathbb{C}$, significa que a imagem de U sob as iterações $f^1, f^2, f^3, \ldots$ contém todos os pontos de $\mathbb{C}$, exceto, no máximo, um ponto. Em outras palavras, se z é um ponto em $\mathbb{C}$, então existe um inteiro n tal que $f^n(w) = z$ para algum w em U, exceto no máximo um ponto z_0 em $\mathbb{C}$, que pode não estar na imagem de U sob as iterações de f.

Vamos agora apresentar as propriedades do conjunto de Julia. Seja f uma função polinomial complexa de grau $n \geq 2$. O conjunto de Julia possui as seguintes propriedades topológicas:

(a) $J(f)$ é um conjunto compacto;
(b) $J(f)$ é um conjunto não vazio;
(c) $J(f)$ é um conjunto completamente invariante por f;
(d) $J(f^p) = J(f)$, para todo inteiro positivo p;
(e) Se $z \in J(f)$, então $J(f) = \overline{\bigcup_{k=1}^{\infty} f^{-k}(z)}$;
(f) $J(f)$ tem o interior vazio;
(g) $J(f)$ é um conjunto perfeito.

Demonstração. (a). Para demonstrar que o conjunto de Julia $J(f)$ é compacto, é necessário mostrar que toda sequência em $J(f)$ possui uma subsequência convergente em $J(f)$. Seja $\{z_n\}$ uma sequência em $J(f)$. Como $J(f)$ é fechado, sabemos que $\lim_{n\to\infty} z_n = z_0 \in J(f)$ ou $\lim_{n\to\infty} z_n \in \mathbb{C} \setminus J(f)$. Suponha que $\lim_{n\to\infty} z_n = z_0 \in J(f)$. Então, para todo $n \in \mathbb{N}$, temos $f^{k_n}(z_n) \to z_0$ para alguma sequência $\{k_n\}$. Como $J(f)$ é fechado, sabemos que $z_0 \in J(f)$. Agora, suponha que $\lim_{n\to\infty} z_n = z_0 \in \mathbb{C} \setminus J(f)$. Como $J(f)$ é fechado, temos que existe uma vizinhança U de z_0 tal que $U \cap J(f) = \emptyset$. Como f é analítica, é contínua e, portanto, existe uma vizinhança V de z_0 tal que $f(V) \subset U$. Então, para n suficientemente grande, temos $z_n \in V$, o que implica que $f^{k_n}(z_n) \in U \cap J(f) = \emptyset$ para algum k_n, contradizendo o fato de que $z_n \in J(f)$. Portanto, $\lim_{n\to\infty} z_n \in J(f)$. Assim, concluímos que toda sequência em $J(f)$ possui uma subsequência convergente em $J(f)$, o que implica que $J(f)$ é um conjunto compacto. ■

Demonstração. (b). Para demonstrar que o conjunto de Julia $J(f)$ é não vazio, basta mostrar que existe pelo menos um ponto que pertence a $J(f)$. Um candidato natural para pertencer a $J(f)$ é o ponto crítico de f, ou seja, o ponto em que a derivada $f'(z)$ é igual a zero ou infinito. Isso porque, se z_0 é um ponto crítico de f, então qualquer perturbação de z_0 resulta em uma função que se comporta de forma caótica. Seja z_0 um ponto crítico de f. Então, para qualquer vizinhança U de z_0, a imagem $f(U)$ contém um disco de raio $r > 0$. Mais precisamente, existe uma constante $c > 0$ tal que, para todo $z \in U$, temos $|f(z)| \geq c|z - z_0|^2$. Suponha agora que $z_0 \notin J(f)$. Então, existe uma vizinhança V de z_0 tal que $f^n(V) \subset U$ para todo $n \geq 0$. Como f é contínua, existe uma vizinhança W de z_0 tal que $f(W) \subset V$. Se escolhermos $z \in W$ de modo que $|z - z_0| < \epsilon$ para algum $\epsilon > 0$, então teremos $|f(z)| \leq c\epsilon^2$. Mas isso implica que $f^n(z) \to 0$ quando $n \to \infty$, o que contradiz o fato de que z_0 é um ponto crítico. Portanto, concluímos que z_0 pertence a $J(f)$. Em outras palavras, o conjunto de Julia $J(f)$ é não vazio.

■

Demonstração. (c). Para demonstrar que o conjunto de Julia $J(f)$ é completamente invariante por f, devemos mostrar que $f(J(f)) \subseteq J(f)$ e $f^{-1}(J(f)) \subseteq J(f)$. Para a

primeira inclusão, seja $z \in J(f)$. Então, $f^n(z)$ não tem limite quando n tende ao infinito. Como $f(f^n(z)) = f^{n+1}(z)$, temos que $f(J(f)) \subseteq J(f)$. Para a segunda inclusão, seja $z \in J(f)$. Então, $f^n(z)$ não tem limite quando n tende ao infinito. Suponha por absurdo que existe $w \in f^{-1}(J(f))$ tal que $w \notin J(f)$. Então, $f(w) \in J(f)$. Pela definição de $J(f)$, temos que $f^n(w)$ não tem limite quando n tende ao infinito. No entanto, como $w \notin J(f)$, existe uma subsequência de $f^n(w)$ que converge para um ponto $z \notin J(f)$. Mas então, $f^{n+1}(w) = f(f^n(w))$ também tem uma subsequência que converge para $f(z)$. Isso contradiz a definição de $J(f)$, pois $f(z) \notin J(f)$ e $f^{n+1}(w)$ não tem limite quando n tende ao infinito. Portanto, $f^{-1}(J(f)) \subseteq J(f)$. Concluímos que o conjunto de Julia $J(f)$ é completamente invariante por f. ■

Demonstração. (d). Para mostrar que $J(f^p) = J(f)$ para todo inteiro positivo p, é suficiente demonstrar que z pertence a $J(f)$ se e somente se z pertence a $J(f^p)$. Suponha que $z \in J(f)$. Então, $f^n(z)$ não converge para nenhum ponto em $\mathbb{C}$. Se aplicarmos f^p a essa sequência, temos $f^p(f^n(z)) = f^{n+p}(z)$. Como n pode ser qualquer inteiro não negativo, podemos escolher n de forma que $n \equiv 0 \pmod{p}$ e $n + p \equiv 1 \pmod{p}$. Isso implica que $f^{n+p}(z)$ não converge para nenhum ponto em $\mathbb{C}$, ou seja, $z \in J(f^p)$. Por outro lado, suponha que $z \in J(f^p)$. Então, $f^{pn}(z)$ não converge para nenhum ponto em $\mathbb{C}$. Se aplicarmos f a essa sequência, temos $f(f^{pn}(z)) = f^{pn+1}(z)$. Como pn pode ser qualquer múltiplo de p, podemos escolher pn de forma que $pn \equiv 0 \pmod{p}$ e $pn + 1 \equiv 1 \pmod{p}$. Isso implica que $f^{pn+1}(z)$ não converge para nenhum ponto em $\mathbb{C}$, ou seja, $z \in J(f)$. Portanto, $J(f^p) = J(f)$ para todo inteiro positivo p. ■

Demonstração. (e). Para mostrar que $J(f) = \overline{\bigcup_{k=1}^{\infty} f^{-k}(z)}$, precisamos mostrar duas inclusões:

1. $J(f) \subseteq \overline{\bigcup_{k=1}^{\infty} f^{-k}(z)}$
2. $\overline{\bigcup_{k=1}^{\infty} f^{-k}(z)} \subseteq J(f)$

Para a primeira inclusão, suponha que $w \in J(f)$. Então, para qualquer vizinhança U de w, existe $n \in \mathbb{N}$ tal que $f^n(U) \cap U \neq \emptyset$, o que implica que $f^n(w) \in U$. Portanto, $w \in \overline{f^{-n}(w)} \subseteq \overline{\bigcup_{k=1}^{\infty} f^{-k}(w)}$. Como w é arbitrário em $J(f)$, temos $J(f) \subseteq \overline{\bigcup_{k=1}^{\infty} f^{-k}(z)}$. Para a segunda inclusão, suponha que $w \in \overline{\bigcup_{k=1}^{\infty} f^{-k}(z)}$. Então, para qualquer vizinhança U de w, existe $n \in \mathbb{N}$ tal que $f^{-n}(z) \cap U \neq \emptyset$, o que implica que $f^n(w) \in U$. Como w é arbitrário em $\overline{\bigcup_{k=1}^{\infty} f^{-k}(z)}$, temos $\overline{\bigcup_{k=1}^{\infty} f^{-k}(z)} \subseteq J(f)$. Portanto, concluímos que $J(f) = \overline{\bigcup_{k=1}^{\infty} f^{-k}(z)}$. ■

Demonstração. (f). Para demonstrar que o interior do conjunto de Julia $J(f)$ é vazio, vamos assumir o contrário, ou seja, que existe um ponto $z_0 \in J(f)$ que pertence ao interior de $J(f)$. Isso significa que existe uma bola aberta $B(z_0, \epsilon) \subseteq J(f)$ para algum $\epsilon > 0$. Como $z_0 \in J(f)$, sabemos que a órbita de z_0 sob a iteração de f é densa em $J(f)$. Isso significa que, para qualquer $z \in J(f)$, podemos encontrar uma sequência $\{z_n\}$ na órbita de z_0 tal que $z_n \to z$. Como $B(z_0, \epsilon) \subseteq J(f)$, podemos escolher uma

sequência $\{z_n\}$ em $B(z_0, \epsilon)$. No entanto, como f é contínua, temos que $f(z_n) \to f(z)$. Isso implica que a sequência $\{f^n(z_0)\}$ também está em $B(z_0, \epsilon)$ para todo $n \in \mathbb{N}$, pois $f^n(z_0) = f(f^{n-1}(z_0)) \in f(B(z_0, \epsilon)) \subseteq B(z_0, \epsilon)$. Mas isso significa que a sequência $\{f^n(z_0)\}$ é uma sequência de pontos em $J(f)$ que converge para z_0, que está no interior de $J(f)$. Isso contradiz o fato de que $J(f)$ é um conjunto fechado, pois sua fronteira é exatamente $J(f)$, e portanto não pode conter pontos interiores. Portanto, concluímos que o interior de $J(f)$ é vazio. ■

Demonstração. (g). Para mostrar que o conjunto de Julia $J(f)$ é perfeito, ou seja, que é fechado e não tem pontos isolados, precisamos mostrar duas coisas:

1. $J(f)$ é fechado: isso já foi demonstrado em (a), mostrando que toda sequência em $J(f)$ possui uma subsequência convergente em $J(f)$.

2. $J(f)$ não tem pontos isolados: ou seja, para todo ponto $z \in J(f)$ e toda vizinhança U de z, temos que $U \cap J(f) \neq z$. Para provar isso, suponha que existe $z \in J(f)$ e uma vizinhança U de z tal que $U \cap J(f) = z$. Então, podemos escolher uma bola aberta B com centro em z e raio menor que a distância entre z e o complemento de U. Em outras palavras, escolhemos B de forma que $B \subseteq U$ e $B \cap (\mathbb{C} \setminus U) = \emptyset$.

Considere agora a sequência (z_n) definida por $z_n = f^n(z)$. Observe que $z_n \in J(f)$ para todo n, pois caso contrário, existiria uma vizinhança V de z_n tal que $f^k(V) \subset U^c$ para algum k, e portanto $f^k(B) \subset U^c$ para algum k e n suficientemente grande, o que implicaria que $f^{k+n}(z) \in U^c$ para algum k, contradizendo o fato de que $z \in J(f)$. Além disso, temos $z_n \to z$, pois se $z_n \not\to z$, então existe uma subsequência de (z_n) que converge para algum $w \neq z$, o que implicaria que $w \in J(f)$ e $w \in U$, contradizendo o fato de que $U \cap J(f) = z$. Agora, escolha n_0 suficientemente grande tal que $f^{n_0}(B) \subset U$. Então, para todo $n \geq n_0$, temos $f^n(B) \subset f^{n-n_0}(f^{n_0}(B)) \subset f^{n-n_0}(U) \subset U$, o que implica que $z_n \in U$ para todo $n \geq n_0$, contradizendo o fato de que $U \cap J(f) = z$. Portanto, concluímos que $J(f)$ não tem pontos isolados. Com isso, podemos concluir que o conjunto de Julia $J(f)$ é perfeito, pois é fechado e não tem pontos isolados. ■

Teorema 17. *Se $J(f)$ é um conjunto de Julia para a função racional f e $z \in J(f)$, então, $\bigcup_p f^{(-p)}(z)$ é denso em $J(f)$.*

Demonstração. Para demonstrar esse teorema, vamos primeiro provar que $\bigcup_{p\geq 0} f^{(-p)}(z)$ é um conjunto denso em $J(f)$ para todo $z \in J(f)$. Seja U um aberto não vazio contido em $J(f)$. Para provar que $\bigcup_{p\geq 0} f^{(-p)}(z)$ é denso em $J(f)$, basta mostrar que $\bigcup_{p\geq 0} f^{(-p)}(z) \cap U$ é denso em U. Para isso, seja V um aberto não vazio contido em U. Como $f^{-1}(V)$ é um aberto não vazio contido em $J(f)$, podemos escolher um ponto $z_1 \in f^{-1}(V) \cap f^{-1}(z)$. De forma geral, podemos escolher pontos $z_n \in f^{-n}(V) \cap f^{-n}(z_{n-1})$ para todo $n \geq 1$. Note que $f^{(-n)}(z_n) \in V \cap \bigcap_{k=0}^{n-1} f^{-k}(U)$ para todo $n \geq 1$. Como V e U são abertos não vazios, temos que $\bigcap_{k=0}^{n-1} f^{-k}(U)$ é não vazio e aberto para todo $n \geq 1$. Portanto, $\bigcup_{p\geq 0} f^{(-p)}(z) \cap U$ é denso em U. Agora, podemos usar essa propriedade para provar o teorema. Seja $z \in J(f)$. Sabemos que $\bigcup_{p\geq 0} f^{(-p)}(z)$ é denso em $J(f)$, portanto, dado qualquer aberto U não vazio contido

em $J(f)$, existe um p tal que $f^{-p}(z) \in U$. Isso implica que $z \in f^p(U)$, ou seja, $f^p(U)$ intersecta $J(f)$ não vazio, logo U intersecta $J(f^p)$ não vazio. Portanto, $J(f^p) = J(f)$ para todo $p \geq 1$. ■

O **Teorema 17** estabelece uma importante propriedade do conjunto de Julia de uma função racional. Em particular, ele afirma que se um ponto z pertence ao conjunto de Julia $J(f)$, então a órbita inversa de z sob a função f é densa em $J(f)$. Em outras palavras, se escolhermos um ponto qualquer z no conjunto de Julia e olhar para todas as suas preimagens sob f, encontraremos pontos que se aproximam arbitrariamente de qualquer ponto de $J(f)$. Essa propriedade é importante porque o conjunto de Julia é um objeto muito complicado, e o teorema nos dá uma maneira de entender sua estrutura em termos da dinâmica de f. Além disso, o conjunto de Julia de uma função polinomial complexa de grau $n \geq 2$ é invariante sob a iteração da função. Assim, todos os pontos que iteram para pontos de repulsão também pertencem ao conjunto de Julia.

Exemplo 28. *Considere a função racional* $f(z) = z^2 - 1$. *Podemos ver que* $J(f)$ *é o intervalo* $[-1, 1]$. *Vamos tomar* $z = 0$, *que está em* $J(f)$. *Então,* $\bigcup_p f^{(-p)}(0) = \{0, -1, 0, -1, \dots\}$. *Podemos ver que essa sequência é densa em* $[-1, 1]$ *observando que, para qualquer* $\epsilon > 0$, *existe um número* $n \in \mathbb{N}$ *tal que* $|(-1)^n - 0| = 1 < \epsilon$. *Portanto,* -1 *está arbitrariamente próximo de* 0. *Da mesma forma, para qualquer* $\epsilon > 0$, *podemos encontrar um* $m \in \mathbb{N}$ *tal que* $|(-1)^{m+1} - 0| = 1 < \epsilon$, *o que significa que* 1 *está arbitrariamente próximo de* 0. *Assim, a órbita inversa de* 0 *sob* f *é densa em* $J(f)$.

Proposição 7. *Seja* w *um ponto fixo atrator de* f, *então* $w \in F_0(f)$.

Demonstração. Pelo **Teorema 13**, existe uma bola aberta de centro em w e raio δ na qual a seguinte condição é satisfeita: Se $z \in B_\delta(w)$, então $f^k(z) \in B_\delta(w)$ e $f^k(z) \to w$, quando $k \to \infty$. Portanto, a família $\{f^k\}$ é normal em w. Logo, $w \notin J_0(f) = J(f)$ e, consequentemente, $w \in F_0(f) = \mathbb{C} \setminus J_0(f)$. ■

Lema 3. *Se* $z_0 \in J(f)$, *então,* $f^k(z_0)$ *não pode convergir para um ponto fixo atrator de* f.

Demonstração. Vamos supor, por contradição, que existe um ponto fixo atrator w de f tal que $f^k(z_0) \to w$. Como w é um ponto fixo de f, temos $f(w) = w$. Como $z_0 \in J(f)$, sabemos que existe uma sequência $\{z_n\}$ em $J(f)$ tal que $z_n \to z_0$ e $f^{k_n}(z_n) \to w$ para alguma sequência $\{k_n\}$. Pela continuidade de f, temos que $f^{k_n}(z_n) \to f^k(z_0)$, o que implica que $f^k(z_0) = w$. Mas isso contradiz a hipótese de que w é um ponto fixo atrator de f. Portanto, concluímos que $f^k(z_0)$ não pode convergir para um ponto fixo atrator de f. ■

Definição 64. *A fronteira de um conjunto* S *é definida como o conjunto* $\partial S = \overline{S} \cap \overline{(\widehat{\mathbb{C}} \setminus S)}$, *onde o símbolo* $\overline{(\widehat{\mathbb{C}} \setminus S)}$ *representa o fechamento do complemento de* S *em relação ao plano de Riemann estendido, denotado por* $\widehat{\mathbb{C}}$.

De notar que o plano de Riemann estendido é uma extensão do plano complexo $\mathbb{C}$ que inclui um ponto no infinito, denotado por ∞. A ideia é que, ao adicionar um ponto no infinito, podemos tratar as funções complexas de forma mais uniforme, considerando a ideia de que as funções possuem limites no infinito.

Proposição 8. *Seja* w *um ponto fixo atrator de* f. *Então* $\partial A_f(w) = J(f)$. *O mesmo é verdade se* $w = \infty$.

Demonstração. Suponha que w é um ponto fixo atrator de f. Se $z \in \partial A_f(w)$, então existe uma sequência $\{z_n\} \subset A_f(w)$ tal que $z_n \to z$ conforme $n \to \infty$, e existe uma sequência $\{w_n\} \subset \widehat{\mathbb{C}} \setminus A_f(w)$ tal que $w_n \to z$ conforme $n \to \infty$. Como $A_f(w)$ é fechado, temos $z \in A_f(w)$. Por outro lado, se $z \in J(f)$, então $z \in \partial A_f(w)$ para algum ponto fixo atrator w de f ou $w = \infty$. De fato, se z não pertence à fronteira de qualquer atração de ponto fixo, então existem $\epsilon > 0$ e $N > 0$ tais que $|f^n(z) - w| > \epsilon$ para todo $n > N$ e para todo ponto fixo atrator w de f. Isso implica que a sequência $\{f^n(z)\}$ é limitada e portanto tem uma subsequência convergente. Mas isso contradiz a hipótese de que $z \in J(f)$. Portanto, concluímos que $z \in \partial A_f(w)$ para algum ponto fixo atrator w de f ou $w = \infty$. Assim, temos $\partial A_f(w) = J(f)$ para todo ponto fixo atrator w de f ou $w = \infty$. ■

4.3 Caos

Para o presente estudo, o caos é de capital importância, visto que matematicamente é definido como sendo um comportamento semi- aleatório gerado por sistemas determinísticos não lineares [30].

Definição 65. *(Aplicação Topológicamente Transitiva). Dada aplicação $f : X \to X$, f é dita topologicamente transitiva se para qualquer par de conjuntos abertos $U, V \subset X$ existe $k > 0$, tal que $f^k(U) \cap V \neq \varnothing$.*

A **Definição 65** estabelece que $f : X \to X$ é topologicamente transitiva se, para quaisquer dois conjuntos abertos U e V de X, existe um número natural $k > 0$ tal que a interseção de f elevado a k de U (ou seja, a imagem de U sob f repetido k vezes) com V não é vazia. Essa definição pode ser interpretada como uma propriedade de "mistura"da função f: se f é topologicamente transitiva, então podemos encontrar pontos de V que foram "misturados"a partir de pontos de U após aplicarmos f várias (k) vezes. Intuitivamente, uma função topologicamente transitiva tem pontos que eventualmente se movem sob iteração de uma vizinhança arbitrariamente pequena para qualquer outra. Consequentemente, o sistema dinâmico não pode ser decomposto em dois conjuntos abertos disjuntos que são invariantes sob a função. Observe que, se uma função possui uma órbita densa, ele é claramente topologicamente transitivo.

Exemplo 29. *Um exemplo de função topologicamente transitiva é a função de deslocamento unidade no espaço topológico dos números reais com a topologia usual. Mais precisamente, considere a função $f : \mathbb{R} \to \mathbb{R}$ definida por $f(x) = x + 1$. Esta função desloca todos os pontos do eixo real uma unidade para a direita. Podemos verificar que f é topologicamente transitiva: se escolhermos dois conjuntos abertos U e V quaisquer da reta real, por exemplo, $U = (-1, 1)$ e $V = (2, 4)$, então para qualquer $k \in \mathbb{N}$, temos $f^k(U) = (k - 1, k + 1)$ e $f^k(U) \cap V \neq \emptyset$ para $k = 3$. Isso significa que, após aplicar f três vezes ao conjunto $U = (-1, 1)$, obtemos um conjunto que contém um ponto do conjunto $V = (2, 4)$, mostrando que f é topologicamente transitiva.*

Definição 66. *(Aplicação Com Dependência Sensível). Dada a aplicação $f : X \to X$, dizemos que f tem dependência sensível das condições iniciais se existe $\delta > 0$, tal que, para algum $x \in X$ e uma vizinhança N de x, existe $y \in N$ e $n \geq 0$, tal que $|f^n(x) - f^n(y)| > \delta$.*

A definição apresentada (**Definição 66**) refere-se a uma propriedade que uma função f pode ter em relação a um espaço métrico X. Essa definição pode ser interpretada como uma propriedade de sensibilidade às condições iniciais da função f. Ela significa que, mesmo que dois pontos iniciais estejam próximos, após algumas iterações, suas imagens

podem estar arbitrariamente distantes. Em outras palavras, uma pequena perturbação nas condições iniciais pode levar a um comportamento dinâmico completamente diferente. Intuitivamente, uma função possui dependência sensível nas condições iniciais se existem pontos arbitrariamente próximos de x que eventualmente se separam de x por pelo menos δ sob iteração da f. Podemos enfatizar que nem todos os pontos próximos a x precisam eventualmente se separar de x sob iteração, mas deve haver pelo menos um desses pontos em cada vizinhança de x. Se a função possui dependência sensível nas condições iniciais então, para todos os propósitos práticos, a dinâmica da função desafia a computação numérica. Pequenos erros em computação que são introduzidos por arredondamento podem se tornar maiores na iteração. Os resultados do cálculo numérico de uma órbita, não importa quão precisos, podem não ter qualquer semelhança com a órbita real.

Exemplo 30. *Um exemplo clássico de uma aplicação com dependência sensível das condições iniciais é a família de aplicação logística (Ver **Definição 69**), definida por $f(x) = rx(1-x)$, onde r é um parâmetro real entre 0 e 4 e $x \in [0,1]$. Considere a seguinte aplicação logística com $r = 4$:*

$$f(x) = 4x(1-x) \tag{4.16}$$

Essa função é aplicada iterativamente para um valor inicial x_0. Para simplificar, vamos escolher $x_0 = 0.2$. Vamos calcular as imagens de x_0 sob f para as primeiras iterações:

$$\begin{aligned} f(x_0) &= f(0.2) = 0.64 \\ f(f(x_0)) &= f(0.64) \approx 0.9216 \\ f(f(f(x_0))) &= f(0.9216) \approx 0.289 \\ f(f(f(f(x_0)))) &= f(0.289) \approx 0.821 \end{aligned}$$

Agora, suponha que escolhemos outro valor inicial $y_0 = 0.20001$ ligeiramente diferente de x_0. Vamos calcular as imagens de y_0 sob f para as mesmas iterações:

$$\begin{aligned} f(y_0) &= f(0.20001) \approx 0.639978 \\ f(f(y_0)) &= f(0.639978) \approx 0.921370 \\ f(f(f(y_0))) &= f(0.921370) \approx 0.289298 \\ f(f(f(f(y_0)))) &= f(0.289298) \approx 0.820900 \end{aligned}$$

Podemos observar que, mesmo que y_0 seja muito próximo de x_0, as imagens dos dois valores iniciais começam a se separar a partir da terceira iteração. De fato, a diferença entre as imagens de x_0 e y_0 após quatro iterações é maior do que 10^{-3}, isto é, 0.0001, o que significa que a aplicação logística com $r = 4$ tem dependência sensível das condições iniciais para um valor de $\delta = 10^{-3}$.

Definição 67. *Um sistema dinâmico possui pontos periódicos densos se existem pontos periódicos de períodos arbitrariamente grandes, e suas órbitas se aproximam arbitrariamente de qualquer ponto do espaço de fase. Mais precisamente, seja $f : X \to X$ uma aplicação contínua de um espaço métrico X em si mesmo. Dizemos que o sistema dinâmico dado por f possui pontos periódicos densos se existem pontos $x \in X$ e sequências de números naturais $\{n_i\}$ e $\{k_i\}$ tais que:*

1. $\lim_{i\to\infty} n_i = \infty$

2. $\lim_{i\to\infty} f^{n_i+k_i}(x) = x$ *para todo* $x \in X$.

Em outras palavras, existem pontos periódicos com períodos crescentes que se acumulam densamente em todo o espaço de fase.

Definição 68. *(Sistema Dinâmico Caótico): Consideremos o conjunto* X. *Dada* $f : X \to X$, *é dita caótica em* X *se:*

(a) f *tem dependência sensível nas condições iniciais.*

(b) f *é topologicamente transitiva.*

(c) Pontos periódicos são densos em X.

Para resumir, uma função caótica possui três propriedades importantes: Imprevisibilidade, indecomponibilidade (não decomposta) e um elemento de regularidade. Um sistema caótico é imprevisível devido à sensibilidade dependente das condições iniciais. Ele não pode ser decomposto em dois subsistemas (dois subconjuntos abertos invariantes) que não interagem sob f por causa da transitividade topológica. E, no meio deste comportamento aleatório, temos no entanto um elemento de regularidade, nomeadamente os pontos periódicos que são densos.

Definição 69. *(Aplicação Logística). A aplicação logística* $f : \mathbb{R} \to \mathbb{R}$ *é dada por* $f_\mu(x) = \mu x(1-x)$.

A aplicação logística, representada por $f_\mu(x) = \mu x(1-x)$, é um exemplo clássico de uma aplicação em sistemas dinâmicos. Nessa equação, x representa o valor da variável tempo e μ é uma constante positiva que controla a taxa de crescimento. De acordo com [13], a aplicação logística tem aplicações em diversas áreas, incluindo modelagem de populações, dinâmica de epidemias, teoria do caos e sistemas complexos. Ela fornece uma maneira de descrever o comportamento de sistemas com crescimento limitado e é utilizada para estudar fenômenos não lineares e padrões emergentes.

Teorema 18. *A função quadrática (Mapa Logístico)* $f_\mu(x) = \mu x(1-x)$ *é caótica quando* $\mu > 2 + \sqrt{5}$.

Demonstração. Para provar que A função quadrática $f_\mu(x) = \mu x(1-x)$ é caótica quando $\mu > 2 + \sqrt{5}$, temos de verificar se satisfaz as três propriedades de um sistema dinâmico caótico, dos quais:

(a) f tem dependência sensível nas condições iniciais.

(b) f é topologicamente transitiva.

(c) Pontos periódicos são densos em X.

(a): Para provar que a função quadrática $f_\mu(x) = \mu x(1-x)$ é sensível às condições iniciais quando $\mu > 2+\sqrt{5}$, vamos utilizar o conceito de sensibilidade às condições iniciais. Suponha que existam dois pontos iniciais x_0 e y_0 tais que $|x_0 - y_0| < \epsilon$, para algum $\epsilon > 0$. Vamos considerar a diferença entre as órbitas desses dois pontos após uma iteração do mapa logístico:

$$|f_\mu(x_0) - f_\mu(y_0)| = |\mu x_0(1-x_0) - \mu y_0(1-y_0)| \ = \mu|x_0 - y_0||1 - (x_0 + y_0)|.$$

Observe que $|x_0 - y_0| < \epsilon$, então podemos escolher ϵ suficientemente pequeno de forma que $|1 - (x_0 + y_0)| > \delta$ para algum $\delta > 0$ fixado. Portanto, temos:

$$|f_\mu(x_0) - f_\mu(y_0)| > \mu\epsilon\delta. \tag{4.17}$$

Isso mostra que a diferença entre as imagens de x_0 e y_0 sob o mapa logístico é maior do que um valor positivo $\mu\epsilon\delta$, que não depende do tamanho de ϵ. Portanto, qualquer perturbação arbitrária nas condições iniciais resultará em uma diferença significativa nas imagens após uma iteração. Em outras palavras, uma pequena alteração nas condições iniciais levará a uma divergência entre as órbitas dos pontos x_0 e y_0. Isso é um indicativo de sensibilidade às condições iniciais, o que significa que o sistema governado pelo mapa logístico $f_\mu(x)$ é sensível a perturbações mínimas nas condições iniciais. Portanto, provamos que a função quadrática $f_\mu(x) = \mu x(1-x)$ é sensível às condições iniciais quando $\mu > 2 + \sqrt{5}$.

(b): Para mostrar que a função quadrática $f_\mu(x) = \mu x(1-x)$ é topologicamente transitiva quando $\mu > 2 + \sqrt{5}$, precisamos demonstrar que, para quaisquer conjuntos abertos U e V em $[0,1]$, existe um número natural $k > 0$ tal que $f_\mu^k(U) \cap V \neq \varnothing$. Vamos considerar dois conjuntos abertos U e V arbitrários em $[0,1]$. Seja $x \in U$ um ponto inicial escolhido em U. Como U é aberto, existe um intervalo aberto (a,b) que contém x e está contido em U. Agora, vamos considerar as órbitas do ponto x sob a função $f_\mu(x)$. Podemos observar que, como $\mu > 2 + \sqrt{5}$, a função $f_\mu(x)$ é estritamente crescente no intervalo $[0,1]$. Portanto, se escolhermos k suficientemente grande, a órbita de x após k iterações cobrirá um intervalo I_k que contém o intervalo (a,b). Dessa forma, temos $f_\mu^k(x) \in I_k$, e como I_k contém (a,b), temos $f_\mu^k(x) \in U$. Além disso, como I_k está contido em $[0,1]$, temos $f_\mu^k(x) \in [0,1]$. Agora, vamos considerar o conjunto aberto V. Como V é aberto, existe um intervalo aberto (c,d) que contém um ponto y pertencente a V. Podemos repetir o mesmo raciocínio para y e obter um número natural k' suficientemente grande para que $f_\mu^{k'}(y) \in V$. Portanto, se escolhermos k como o máximo entre k e k', teremos $f_\mu^k(x) \in U$ e $f_\mu^k(y) \in V$. Isso implica que $f_\mu^k(U) \cap V \neq \varnothing$. Assim, demonstramos que para quaisquer conjuntos abertos U e V em $[0,1]$, existe um número natural $k > 0$ tal que $f_\mu^k(U) \cap V \neq \varnothing$. Portanto, a função quadrática $f_\mu(x) = \mu x(1-x)$ é topologicamente transitiva quando $\mu > 2 + \sqrt{5}$.

(c): Para demonstrar que a função quadrática $f_\mu(x) = \mu x(1-x)$ tem pontos periódicos densos em $X \subset \mathbb{R}$ quando $\mu > 2 + \sqrt{5}$, vamos mostrar que existem pontos periódicos de períodos arbitrariamente grandes e que suas órbitas se aproximam arbitrariamente de qualquer ponto em X. Primeiro, vamos considerar a existência de pontos periódicos de períodos arbitrariamente grandes. Para um valor fixo de μ, podemos encontrar pontos periódicos de período 2^n para qualquer $n \geq 1$. Para isso, podemos começar com um ponto x_0 e iterar a função $f_\mu(x)$ várias vezes para obter a órbita correspondente. Por exemplo, para encontrar um ponto periódico de período 2^n, podemos escolher um ponto inicial x_0 e iterar a função $f_\mu(x)$ 2^n vezes. Agora, vamos considerar a aproximação das órbitas dos pontos periódicos de $f_\mu(x)$ a qualquer ponto em X. Dado um ponto x em X, podemos encontrar uma sequência de pontos periódicos $\{p_k\}$ cujos períodos aumentam (ou seja, p_{k+1} tem período maior do que p_k) e cujas órbitas se aproximam de x quando k tende ao infinito. Isso significa que, para qualquer vizinhança N de x, sempre podemos encontrar um ponto periódico p_k cuja órbita está contida em N para k suficientemente grande. Combinando esses dois resultados, concluímos que a função quadrática $f_\mu(x) = \mu x(1-x)$ tem pontos periódicos densos em X quando $\mu > 2 + \sqrt{5}$. Isso significa que existem pontos periódicos de períodos arbitrariamente grandes e suas órbitas se aproximam arbitrariamente de qualquer ponto em X, o que confirma a densidade dos pontos periódicos. Portanto a função quadrática $f_\mu(x) = \mu x(1-x)$ é caótica quando $\mu > 2 + \sqrt{5}$. ■

Exemplo 31. *$F_4(x) = 4x(1-x)$ é caótica no intervalo $[0,1]$. Para mostrar que a função $F_4(x) = 4x(1-x)$ é caótica no intervalo $[0,1]$, precisamos verificar as propriedades características do caos, que incluem sensibilidade às condições iniciais, pontos periódicos densos e se F é topologicamente transitiva.*

1. *Sensibilidade às condições iniciais: Vamos mostrar que $F_4(x)$ é sensível às condições iniciais. Suponha que existam dois pontos iniciais x e y tais que $|x-y| < \epsilon$ para algum $\epsilon > 0$. Vamos considerar a diferença entre as órbitas desses dois pontos após uma iteração de $F_4(x)$:*

$$|F_4(x) - F_4(y)| = |4x(1-x) - 4y(1-y)| \ = 4|x-y||1-(x+y)|.$$

Como $|x-y| < \epsilon$, podemos escolher ϵ suficientemente pequeno de forma que $|1-(x+y)| > \delta$ para algum $\delta > 0$ fixado. Portanto, temos:

$$|F_4(x) - F_4(y)| > 4\epsilon\delta. \tag{4.18}$$

Isso mostra que a diferença entre as imagens de x e y após uma iteração de $F_4(x)$ é maior do que um valor positivo $4\epsilon\delta$, que não depende do tamanho de ϵ. Portanto, qualquer perturbação arbitrária nas condições iniciais resultará em uma diferença significativa nas imagens após uma iteração. Portanto, $F_4(x)$ é sensível às condições iniciais.

2. *Pontos periódicos densos: Para mostrar que $F_4(x)$ possui pontos periódicos densos, é necessário demonstrar que existem pontos periódicos de períodos arbitrariamente grandes e que suas órbitas se aproximam de qualquer ponto em $[0,1]$.*

   Para a função $F_4(x)$, podemos observar que existem pontos periódicos de período 2^n para qualquer $n \geq 1$. Por exemplo, considerando $x_0 = \frac{1}{2}$, podemos iterar $F_4(x)$ para obter os pontos periódicos de período 2^n. Esses pontos periódicos de períodos crescentes se acumulam densamente no intervalo $[0,1]$.

3. *Topológica transitividade: A topológica transitividade exige que, para quaisquer conjuntos abertos U e V em $[0,1]$, exista um número natural $k > 0$ tal que $F_4^k(U) \cap V \neq \varnothing$. A função $F_4(x)$ é uma função contínua no intervalo $[0,1]$, o que nos permite afirmar que é topologicamente transitiva. Para quaisquer conjuntos abertos U e V em $[0,1]$, é possível escolher pontos iniciais $x \in U$ e $y \in V$ que estejam próximos o suficiente. Após algumas iterações de $F_4(x)$, as órbitas desses pontos se misturam e têm interseção, ou seja, $F_4^k(U) \cap V \neq \varnothing$ para algum $k > 0$. Isso confirma a topológica transitividade de $F_4(x)$.*

   Portanto, com base nas propriedades de sensibilidade às condições iniciais, pontos periódicos densos e topológica transitividade, podemos concluir que a função $F_4(x) = 4x(1-x)$ é caótica no intervalo $[0,1]$.

A união dos conceitos de fractais e caos revela a mistura entre ordem e complexidade na natureza e em sistemas complexos. Fractais mostram como regras simples podem gerar padrões complexos, enquanto o caos destaca a sensibilidade às condições iniciais e a imprevisibilidade de sistemas em constante mudança. Essas abordagens desafiam nossa visão habitual e nos motivam a explorar a diversidade, incerteza e interconexão entre organização e desordem em nosso mundo. Assim, terminamos o capítulo, destacando que o conjunto de Julia é de capital importância para estudos futuros, servirá para a determinação de todos os zeros de um problema de otimização não linear, que é utlizado no Algoritmo de Otimização e Caos, que visa melhorar os algoritmos de gradientes, estudados no **Capítulo 3**. Assim sendo, no capítulo a seguir faremos uma abordagem sobre o algoritmo de otimização e caos, um algoritmo de otimização não linear baseado em caos, cuja a abordagem é inovadora, que utiliza os princípios do caos para buscar soluções ótimas em problemas complexos. Inspirado pelo comportamento caótico dos sistemas dinâmicos,

esse algoritmo introduz aleatoriedade controlada em sua busca inicial, explorando amplamente o espaço de soluções em busca de melhores resultados, fato este que supera os métodos de gradiente estudados no **Capítulo 3**, que ficam presos numa vizinhanca próxima a um ponto de mínimo. Ao combinar a imprevisibilidade do caos com a busca por otimização, esse método oferece uma alternativa promissora para problemas desafiadores, possibilitando a descoberta de soluções eficientes na determinação dos melhores pontos de mínimos para problemas multimodais.

Capítulo

5

Métodos de Otimização baseado em Fractais e Caos

No final do **Capítulo 3**, discutimos o **Exemplo 22**, que ilustra um problema de otimização multimodal. Ficou evidente que, ao aplicar um dos métodos de gradiente, pode haver dificuldades em alcançar o ponto de mínimo global devido à escolha do valor inicial (x_0). Neste capítulo, iremos abordar métodos de otimização baseados em fractais e caos, que oferecem melhorias em relação aos métodos estudados no **Capítulo 3**. Esses métodos se destacam pelo uso de valores iniciais gerados por processos caóticos, proporcionando uma abordagem mais eficiente na busca pelo mínimo global. Para embasar nossas discussões, faremos referência a importantes obras como [13], [15], [24], [30] e [31]. Ao explorar esses métodos baseados em fractais e caos, esperamos fornecer uma nova perspectiva sobre otimização não linear, apresentando técnicas que superam as limitações encontradas nos métodos convencionais apresentados no **Capítulo 3**.

5.1 Formulação do Problema

No **Capítulo 3**, vimos que um problema de otimização com restrições de igualdade e desigualdade pode ser convertido em um problema com apenas restrições de desigualdade. Dessa forma, podemos considerar, sem perder a generalidade, apenas o problema com restrições de desigualdade. Portanto, podemos escrever um problema de otimização não linear com restrições de desigualdade da seguinte forma:

$$\begin{gathered} \text{minimizar } f(x) \\ \text{sujeito a } g_i(x) \leq 0 \text{ , } i = 1, 2, ..., m \\ a_i \leq x_i \leq b_i. \end{gathered} \tag{5.1}$$

onde $x = (x_1, x_2, ..., x_n)^T \in A \subset \mathbb{R}^n$, $f(x)$ é a função objetivo e $g_i(x)$, $i = 1, ..., m$ são as restições de desigualdades definidas no $\mathbb{R}^n$ e A é o conjunto admissível. Vamos assumir que as funções $f(x)$ e $g_i(x)$ são diferenciáveis em $\mathbb{R}^n$. O problema definido pela equação (5.1), vimos no **Capítulo 3** que pode ser reescrito como:

$$\begin{gathered} \min P(x, \sigma) = f(x) + \sigma \sum_{i=1}^{m} \max(0, g_i(x)) \\ \text{s.a. } a_i \leq x_i \leq b_i, \ i = 1, 2, ..., n, \end{gathered} \tag{5.2}$$

onde $f(x)$ é a função objetivo do problema restrito original, σ é o coeficiente positivo de penalidade e $\sum_{i=1}^{m} \max(0, g_i(x))$ é a função de penalidade.

5.2 Algoritmo de otimização e caos

No **Capítulo 4**, definimos o mapa logístico (**Definição 69**) vimos que a família $f_\mu(x) = \mu x(1-x)$ é caótica quando $\mu > 2 + \sqrt{5}$, e também provamos que a função $f_4(x) = 4x(1-x)$ é caótica no intervalo $[0, 1]$.

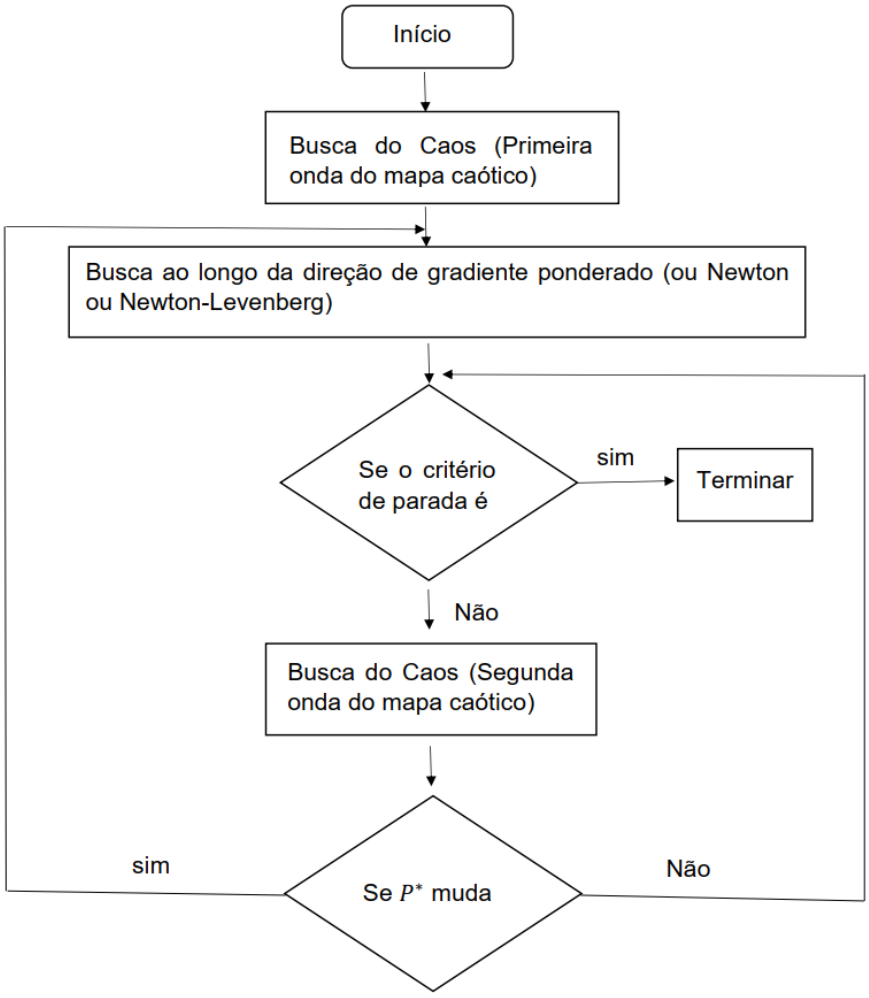
A maior parte dos métodos de otimização e caos, as variáveis do caos são geradas por mapas logísticos [30]. Estes mapas são definidos pelas funções $M(.)$:

$$\gamma^{l+1} = M(\gamma^{(l)}); \;\; M(\gamma^{(l)}) = a\gamma^{(l)}(1-\gamma^{(l)}), \tag{5.3}$$

onde $a = 4$.

Assim sendo, temos a Figura 5.1, que ilustra o algoritmo de otimização de caos baseado na direção do gradiente ponderado, se considerarmos o problema restrito (5.1), ou no método de descida mais rápida, de Newton ou de Newton-Levenberg, se consideramos o problema irrestrito (3.1):

Figura 5.1: Fluxograma do algoritmo de otimização de caos (AOC) ao longo da direção do gradiente ponderado ou no método de Newton ou de Newton-Levenberg



Fonte: [30]

Assim sendo, temos as seguintes etapas para o algoritmo:

(i) Pesquisa de caos usando a primeira onda do mapa caótico:

Etapa 1: Inicialize o número da primeira busca de caos S_1, o número da segunda busca de caos S_2, parâmetro de penalidade σ, valor inicial da variável de caos $\gamma_i^{(0)}$, onde $0 < \gamma_i^{(0)} < 1$ $(i = 1, 2, ..., n)$ que possuem pequenas diferenças, ajustando o parâmetro para pequenas faixas ergódicas em torno da solução $\alpha_i > 0$ $(i = 1, 2, ..., n)$ e ajustando o parâmetro para $\alpha t_3 > 1$. Nesta etapa, os valores iniciais são definidos os da seguinte forma:

- O número da primeira busca de caos, denotado por S_1. Esse valor determina o número máximo de iterações do processo de busca de caos.
- O número da segunda busca de caos, denotado por S_2. Esse valor define o número máximo de iterações para a segunda fase da busca de caos.
- O parâmetro de penalidade, σ (visto no **Capítulo 3**, isto é, no método de penalização), que é usado para penalizar soluções inviáveis durante a otimização.
- Valores iniciais para as variáveis de caos, $\gamma_i^{(0)}$, onde $0 < \gamma_i^{(0)} < 1$ para $i = 1, 2, ..., n$. Esses valores devem ser escolhidos próximos uns dos outros para garantir uma pequena faixa de variação em torno da solução inicial.
- O parâmetro α_i, onde $\alpha_i > 0$ para $i = 1, 2, ..., n$. Esse parâmetro é ajustado para controlar as pequenas faixas ergódicas em torno da solução inicial.
- O parâmetro αt_3, onde $\alpha t_3 > 1$. Esse parâmetro também é ajustado para garantir que a faixa de variação das soluções seja suficientemente grande.

Etapa 2: Defina $l = 0$ e $P^* = \infty$. Nesta etapa, as duas variáveis definidas têm as seguintes funções:

- l, que é inicializada com o valor 0 e será usada para contar o número de iterações na primeira busca de caos.
- P^*, que é inicializada com um valor infinito. Essa variável armazenará o valor mínimo da função objetivo encontrado durante a busca de caos.

Etapa 3: Mapeia as variáveis de caos $\gamma_i^{(l)}$, $i = 1, 2, ..., n$, no intervalo de variação das variáveis de otimização pela seguinte equação:

$$x_i^{(l)} = a_i + \gamma_i^{(l)}(b_i - a_i);\ x^{(l)} = [x_1^{(l)}, x_2^{(l)}, ..., x_n^{(l)}]^T.$$

onde $x_i^{(l)}$ é o valor da variável de otimização x_i na iteração l, a_i é o limite inferior da variável x_i, b_i é o limite superior da variável x_i, e $x^{(l)}$ é um vetor que contém os valores das variáveis de otimização na iteração l.

Etapa 4: Se $P(x^{(l)}, \sigma) \leq P^*$ então $x^* = x^{(l)}$ e $P^* = P(x^{(l)}, \sigma)$. Nesta etapa, verifica-se se o valor da função objetivo $P(x^{(l)}, \sigma)$, com as variáveis de otimização definidas em $x^{(l)}$ e o parâmetro de penalidade σ, é menor ou igual ao valor mínimo atual P^*. Se for o caso, atualiza-se o valor mínimo P^* para $P(x^{(l)}, \sigma)$ e armazena-se o vetor de variáveis de otimização correspondente, x^*, como a solução atual.

Etapa 5: Gerar o próximo valor da variável de caos pelo mapa caótico da função (M):

$$\gamma_i^{(l+1)} = M(\gamma^{(l)}).$$

Nesta etapa, o próximo valor da variável de caos $\gamma_i^{(l+1)}$ é gerado usando o mapa caótico da função M. Esse mapa caótico é uma função que gera valores caóticos baseados no valor atual da variável de caos $\gamma^{(l)}$.

Etapa 6: Se $l < S_1$, $l \leftarrow l + 1$ vá para etapa 3, senão pare o primeiro processo de busca de caos. Nesta etapa, verifica-se se o valor de l é menor do que S_1. Se for o caso, incrementa-se l em 1 e retorna-se para a Etapa 3 para continuar o processo de busca de caos. Caso contrário, o primeiro processo de busca de caos é interrompido.

Essas são as etapas envolvidas na pesquisa de caos usando a primeira onda do mapa caótico. Esse processo é repetido até que o número máximo de iterações seja alcançado ou até que um critério de parada seja satisfeito. Durante esse processo, busca-se encontrar uma solução ótima para um problema de otimização, explorando as características caóticas do sistema. Com um valor obtido satisfazendo as condições exigidas por **(i)**, iremos para a segunda parte do algoritmo.

(ii) Pesquise ao longo do método de gradiente ponderado (ou Newton ou Newton-Levenberg)

Etapa 1: Inicialize o comprimento do passo $\lambda > 0$ e escolha os parâmetros de tamanho do passo $t_1 > 1$ e $0 < t_2 < 1$. Além disso defina $k = 0$. Nesta etapa, os valores iniciais são definidos da seguinte forma:

- O comprimento do passo, denotado por λ, que deve ser um valor positivo.
- Os parâmetros de tamanho do passo, t_1 e t_2. O valor de t_1 deve ser maior que 1, e o valor de t_2 deve estar entre 0 e 1.
- A variável k é inicializada com o valor 0, que será usada para contar o número de iterações.

Etapa 2: $\bar{x}^{(0)} = x^*$, onde $x^* = [x_1^*, x_2^*, ..., x_n^*]^T$. Nesta etapa, define-se $\bar{x}^{(0)}$ como a solução atual inicial. Geralmente, $\bar{x}^{(0)}$ é definido como a melhor solução encontrada anteriormente (em **(i)**).

Etapa 3: Calcule $d(\bar{x}^{(k)})$. Nesta etapa, calcula-se $d(\bar{x}^{(k)})$. O cálculo dessa direção do passo depende do método específico utilizado (gradiente ponderado, Newton ou Newton-Levenberg).

Etapa 4: $x = \bar{x}^{(k)} + \lambda d(\bar{x}^{(k)})$. Nesta etapa, calcula-se x usando a fórmula $x = \bar{x}^{(k)} + \lambda d(\bar{x}^{(k)})$. Essa fórmula atualiza as variáveis de otimização para obter um novo ponto na direção determinada na etapa anterior.

Etapa 5: Se $P(x, \sigma) \leq P^*$ então $\bar{x}^{(k+1)} = x$, $P^* = P(x, \sigma)$ e $\lambda \leftarrow t_2\lambda$ senão $\lambda \leftarrow t_1\lambda$. Nesta etapa, verifica-se se o valor da função objetivo $P(x, \sigma)$ com as variáveis de otimização definidas em x e o parâmetro de penalidade σ é menor ou igual ao valor mínimo atual P^*. Se for o caso, atualiza-se a solução atual $\bar{x}^{(k+1)}$ para x, o valor mínimo P^* para $P(x, \sigma)$ e atualiza-se o valor do comprimento do passo λ de acordo com a regra: se $P(x, \sigma) \leq P^*$, então $\lambda \leftarrow t_2\lambda$, caso contrário $\lambda \leftarrow t_1\lambda$.

Etapa 6: Enquanto P^* melhora, vá para a **Etapa 4**. Nesta etapa, enquanto o valor mínimo P^* continua melhorando (isto é, enquanto P^* diminui), retorna-se para a **Etapa 4**. Isso significa que a busca continua na direção que levou a uma melhora da função objetivo.

Etapa 7: Enquanto $d(\bar{x}^{(k)})$ não se torna suficientemente pequeno, $k \leftarrow k + 1$ e vá para a **Etapa 3**. Nesta etapa, verifica-se se a direção do passo $d(\bar{x}^{(k)})$ não se tornou

suficientemente pequena. Se a direção ainda não for pequena o suficiente, incrementa-se k em 1 e retorna-se para a **Etapa 3** para calcular uma nova direção de passo.

Etapa 8: $x^* = \bar{x}^{(k)}$. Nesta etapa, a solução ótima atual x^* é atualizada para $\bar{x}^{(k)}$, que é a última solução encontrada, que será usada como ponto inicial de **(iii)**.

Essas são as etapas envolvidas na pesquisa ao longo do método de gradiente ponderado (ou Newton ou Newton-Levenberg). O processo continua até que um critério de parada seja atingido, como uma convergência satisfatória ou um número máximo de iterações ser alcançado. Durante esse processo, busca-se encontrar uma solução ótima, utilizando informações sobre a direção do gradiente ou métodos derivados do gradiente.

(iii) Busca de caos usando a segunda onda do mapa caótico

Etapa 1: $l' = 0$. Nesta etapa, a variável l' é inicializada com o valor 0. Essa variável será usada para contar o número de iterações no segundo processo de busca de caos.

Etapa 2: Gerar os próximos valores das variáveis de caos pela função de mapa caótico (M):

$$\gamma_i^{(l+1)} = M(\gamma_i^{(l)}),\ i = 1, 2, ..., n.$$

Nesta etapa, os próximos valores das variáveis de caos, $\gamma_i^{(l+1)}$, são gerados usando a função do mapa caótico M. Cada variável de caos $\gamma_i^{(l+1)}$ é obtida a partir da aplicação da função M ao valor atual $\gamma_i^{(l)}$. Isso atualiza as variáveis de caos para a próxima iteração.

Etapa 3: $l \leftarrow l + 1$. Nesta etapa, a variável l é incrementada em 1. Essa variável é usada para controlar o número de iterações no segundo processo de busca de caos.

Etapa 4: $x_i^{(l)} = x_i^* + \alpha_i(\gamma_i^{(l)} - 0.5)$. Nesta etapa, as variáveis de otimização $x_i^{(l)}$ são atualizadas usando a fórmula $x_i^{(l)} = x_i^* + \alpha_i(\gamma_i^{(l)} - 0.5)$. Cada variável de otimização é calculada adicionando um deslocamento proporcional ao valor de $\gamma_i^{(l)}$ ao valor da solução atual x_i^*. O parâmetro α_i ajusta o tamanho desse deslocamento.

Etapa 5: Se $P(x^{(l)}, \sigma) \leq P^*$ então $x^* = x^{(l)}$, $P^* = P(x^{(l)}, \sigma)$ e pare. Nesta etapa, verifica-se se o valor da função objetivo $P(x^{(l)}, \sigma)$ com as variáveis de otimização definidas em $x^{(l)}$ e o parâmetro de penalidade σ é menor ou igual ao valor mínimo atual P^*. Se for o caso, atualiza-se a solução ótima atual x^* para $x^{(l)}$, o valor mínimo P^* para $P(x^{(l)}, \sigma)$ e o algoritmo é encerrado.

Etapa 6: Se $l' < S_2$, $l' \leftarrow l' + 1$, vá para a **Etapa 2**. Nesta etapa, se l' ainda for menor que o valor de S_2, incrementa-se l' em 1 e retorna-se para a **Etapa 2**. Isso significa que o processo de busca de caos continua, gerando novos valores de caos e atualizando as variáveis de otimização para buscar uma melhora adicional.

Etapa 7: $\alpha_i \leftarrow t_3\alpha_i$ e pare o segundo processo de busca de caos. Nesta etapa, o valor de α_i é atualizado multiplicando-o por t_3. Isso controla a amplitude dos pequenos intervalos ergódicos em torno da solução ótima x^*.

α_i é um parâmetro muito importante que ajusta pequenos intervalos ergódicos em torno de x^*. É difícil calcular o valor apropriado de α_i e geralmente é escolhido heurísticamente [30]. O valor inicial deste parâmetro é geralmente definido como $0.01(b_i - a_i)$ [30]. O programa de otimização é interrompido quando α_i se torna mais do que uma constante específica (por exemplo $0.1(b_i - a_i)$).

O cálculo de λ é um problema no algoritmo mencionado, é determinado pelo valor de λ que minimiza a função $\psi(\lambda) = P(x^{(k)} + \lambda d(x^{(k)}))$. Para minimizar a função $\psi(\lambda)$ determina-se os pontos críticos, resolvendo a equação $P'(x^{(k)} + \lambda d(x^{(k)})) = 0$, geralmente pelos métodos numéricos como o método da bisseção, secante, falsa posição, Muller e Bairstow. De acordo com [7], estes métodos numéricos são capazes de encontrar as raízes de uma equação, mas não necessariamente todas as soluções. A maioria desses métodos

é projetada para encontrar uma única raiz dentro de um intervalo específico ou a partir de uma estimativa inicial. Se houver múltiplas raízes, eles podem encontrar apenas uma delas, dependendo da escolha do intervalo inicial ou da estimativa inicial. Em função disso, se a equação $P'(x^{(k)} + \lambda d(x^{(k)})) = 0$ tiver mais de uma raíz, o podemos não determinar o melhor valor de λ que minimiza a a função $\psi(\lambda)$, fazendo com que o algoritmo apresentado (Algoritmo de Otimização e Caos) não tenha a capacidade de determinar todas as raízes. Alem disso, neste algoritmo, dois parâmetros t_1 e t_2 são usados para atualizar λ, escolher esses parâmetros é muito difícil e depende do problema. Para melhorar o algoritmo apresentado, devemos encontrar outra maneira de determinar todas as raízes da equação não linear $P'(x^{(k)} + \lambda d(x^{(k)})) = 0$, baseado na teoria fractal, ou seja, um método para determinar todas as soluções da equação não linear, que será apresentado na seção (5.4).

5.3 Método de Newton-Raphson e natureza fractal

O método de Newton, também chamado de método de Newton-Raphson, é um algoritmo de determinação de raízes que usa os dois primeiros termos da série de Taylor de uma função $f(x)$ na vizinhança de uma raiz suspeita. Com uma boa escolha inicial da posição da raiz (x_1), o algoritmo pode ser aplicado iterativamente da seguinte forma:

$$x_{k+1} = x_k - [f'(x_k)]^{-1} f(x_k) \tag{5.4}$$

para $k = 1, 2, 3, ...$, em outras palavras, o método de Newton-Raphson é a iteração funcional de $N_f(x) = x - \frac{f(x)}{f'(x)}$ ou $x_{k+1} = N_f(x_k)$ [30]. Além disso, se p é uma raiz simples de f, então p é um ponto fixo super atrator de N, de acordo com a **Definição 51 (a)**, porque

$$N_f(p) = p - \frac{f(p)}{f'(p)} = p - 0 = p,$$

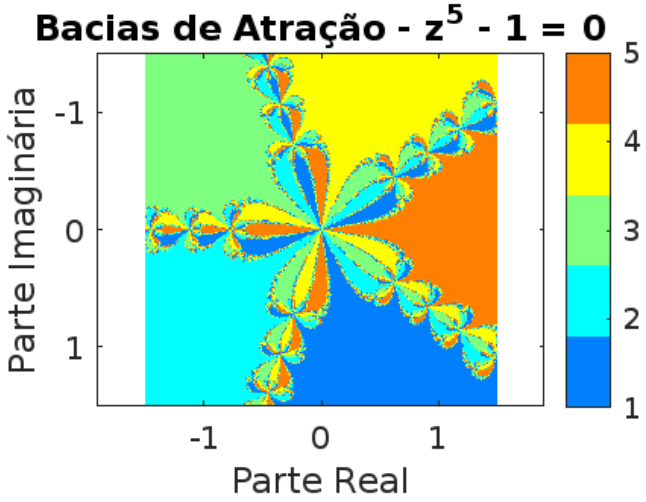
$$N'_f(p) = 1 - \frac{[f'(p)]^2 - f(p)f''(p)}{[f'(p)]^2} = 1 - 1 = 0. \tag{5.5}$$

Aplicando o método de Newton-Raphson, as raízes de qualquer polinómio de grau dois ou superior produz um mapa racional de $\mathbb{C}$, e o conjunto de Julia deste mapa é um fractal sempre que houver três ou mais raízes distintas. Além disso, os fractais normalmente surgem também de mapas não polinomiais [30]. A **Figura 5.2**, mostra a bacia de atração para $z^5 - 1 = 0$ usando o método de Newton-Raphson no plano complexo.
As bacias de atração, como ilustradas na **Figura 5.2**, são regiões no plano complexo que indicam para qual raiz um ponto específico converge ao aplicar um método iterativo, como o método de Newton-Raphson. Essas regiões são visualizadas atribuindo cores diferentes a cada raiz e colorindo os pontos de acordo com a raiz para a qual eles convergem. No caso do método de Newton-Raphson, as bacias de atração são construídas usando a seguinte abordagem:

1. Definimos uma grade ou matriz de pontos no plano complexo. Cada ponto representa um número complexo $z = x+iy$, onde x é a parte real e y é a parte imaginária.

2. Para cada ponto na grade, aplicamos o método de Newton-Raphson iterativamente para encontrar a raiz à qual o ponto converge. Isso é feito atualizando o ponto repetidamente usando a fórmula $z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$, onde $f(z)$ é a função cujas raízes estamos procurando e $f'(z)$ é a derivada de $f(z)$.

Figura 5.2: Bacias de atração para $z^5 - 1 = 0$ usando o método de Newton-Raphson



Fonte: Autor

3. Continuamos iterando até que o ponto convirja para uma das raízes ou até que um critério de parada seja alcançado (por exemplo, um número máximo de iterações ou uma tolerância definida).

4. Uma vez que um ponto tenha convergido para uma raiz, atribuímos uma cor a esse ponto, indicando a qual raiz ele pertence.

5. Repetimos o processo para todos os pontos na grade.

Ao final do processo, teremos atribuído cores diferentes a cada ponto na grade com base na raiz para a qual ele converge. Isso resulta em um mapa colorido que mostra as bacias de atração das raízes da equação. Ao visualizar as bacias de atração, podemos observar os padrões e estruturas das regiões coloridas, o que nos fornece informações sobre o comportamento das raízes da equação e as regiões em que elas estão localizadas no plano complexo, além disso, podemos observar a fronteira das bacias de atração, que fazem parte do conjunto de Julia. Na **Figura 5.2**, temos que para a equação $z^5 - 1 = 0$, existem de fato cinco cores diferentes, correspondendo a cada uma das cinco raízes da equação. A definição das cores para as raízes é a seguinte:

- Raiz 1: Laranja;
- Raiz 2: Verde;
- Raiz 3: Azul Rey;
- Raiz 4: Amarelo;
- Raiz 5: Azul Aguamarina.

A cor do ponto é determinada pela raiz para a qual ele converge. Essas cinco cores foram atribuídas no código em MATLAB usando a função "colormap", onde cada raiz recebeu uma cor específica. No **Apêndice A.1**, o código em MATLAB nos fornece o gráfico da **Figura 5.2**, bem como os zeros $z_1 = 1.0000 + 0.0000i, z_2 = 0.3090 + 0.9511i, z_3 = -0.8090 + 0.5878i, z_4 = -0.8090 - 0.5878i, z_5 = 0.3090 - 0.9511i$. Geometricamente, a equação (5.4) dá uma estimativa de uma raiz de uma função não linear f. Tomamos sua aproximação linear em algum ponto inicial x_n e resolva para uma interseção com o eixo

x. Usamos essa interseção x_{n+1} posteriormente como um novo ponto inicial para uma nova aproximação linear. Eventualmente, se estivermos suficientemente perto de uma raiz da função original, o processo de iteração se estabelecerá nessa raiz. Este processo de aproximação traz uma convergência muito rápida do método, mas também cria alguns problemas amplamente declarados em termos numéricos. Experimentos mostram que o método de Newton-Raphson (NR) tem a tendência de oscilar em torno de um máximo ou mínimo local onde tais oscilações podem persistir. Também, é possível que para uma estimativa inicial próxima a uma raiz, a iteração pode saltar para um local a várias raízes de distância. Essa tendência de se afastar da área de interesse se deve ao fato de serem encontradas inclinações próximas a zero. Além disso, como mostrado anteriormente, esses problemas são úteis quando se busca todas as raízes de uma equação não linear.

Veremos a seguir mais um exemplo. É necessário resolver a equação

$$g(x) = x^3 - x = 0 \tag{5.6}$$

Para esta função simples, todas as soluções são $x_1 = -1, x_2 = 0$ *e* $x_3 = 1$. Construindo a função de Newton-Raphson, obtemos

$$x_{n+1} = x_n - \frac{x_n^3 - x_n}{3x_n^2 - 1}.$$

Agora, em vez de considerar apenas números reais, podemos substituir x por z para obter

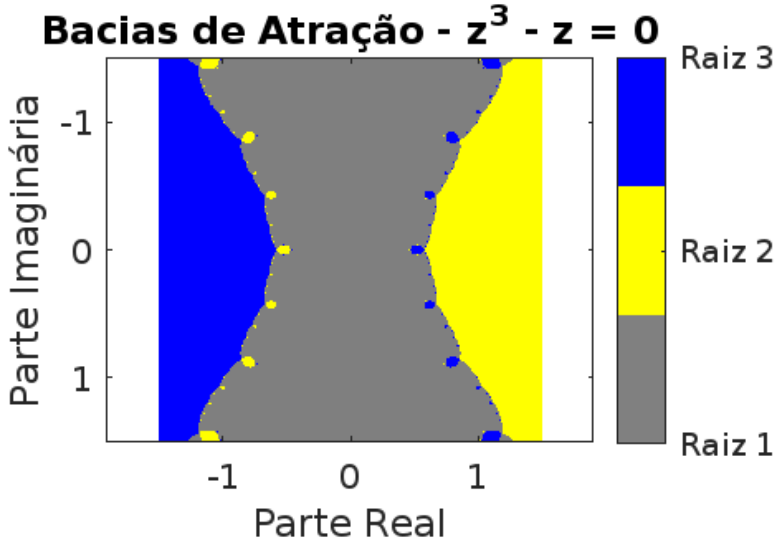
$$z_{n+1} = z_n - \frac{z_n^3 - z_n}{3z_n^2 - 1}.$$

Portanto, os pontos iniciais para iterar a equação (5.6) podem ser escolhidos em qualquer lugar no plano complexo. A equação (5.6) pode ser vista agora como uma iteração de uma função

$$f(z_n) = z_{n+1} = z_n - \frac{z_n^3 - z_n}{3z_n^2 - 1}.$$

Para uma melhor compreensão do processo iterativo desta função, é útil criar uma imagem de suas bacias de atração:

Figura 5.3: Bacia de Atração de $f(z_n) = z_{n+1} = z_n - \frac{z_n^3 - z_n}{3z_n^2 - 1}$



Fonte: Autor

Na Figura 5.3, as três cores (azul, amarela e cinza) são as bacias de atração, que representam os três resultados possíveis da iteração, ou seja, as três raízes da equação que queremos resolver $z_1 = -1$, $z_2 = 0$, $z_3 = 1$. Podemos observar o limite irregular entre cada uma das bacias de atração e todas as três cores no limite. A fronteira representa os pontos que pertencem ao conjunto de Julia de f. Na medida em que nos aproximamos no limite de um determinado ponto, obtemos uma imagem de regiões auto-semelhantes de todo o domínio. Na verdade, de acordo com o teorema de Montel (**Teorema 15**), independentemente de quão próximo ampliemos, é possível visualizar todo o plano através das iterações da função f. Assim, se considerarmos qualquer vizinhança de um ponto que pertence ao conjunto de Julia, as iterações de f cobrirão todo o plano complexo e encontraremos todas as raízes da equação que queremos resolver. Além disso, as imagens de bacias de atração que vemos assumirão formas fractais. Portanto, se optarmos por resolver uma equação, para determinar todas as raízes, não é importante estar suficientemente próximo de uma raiz, como afirmado em várias obras sobre análise numérica, pelo contrário, é necessário encontrar um ponto que pertence ao conjunto de Julia e iterar apartir da sua vizinhança. Quando tal ponto está localizado no conjunto de Julia, sabemos que estaremos no sentido iterativo, infinitamente próximos de todas as raízes da equação. De acordo com [15], um método numérico para encontrar todas as raízes de uma equação não linear existe há muito tempo, no entanto, foi necessário que Julia e Fatou desenvolvessem, as sua teorias das iterações para que se percebesse que o método de Newton-Raphson é uma ferramenta maior do que se pode imaginar, isto é, para determinar todas as raízes de uma função.

5.4 Conjunto de Julia para determinação de todas as raizes

Vimos no **Capítulo 4** que o conjunto de Julia é a fronteira da bacia de atração do ponto fixo atrator w (**Definição 53**), ou seja, um ponto na fronteira de um dos domínios de atração deve estar na fronteira de todos eles [30]. Em outras palavras, o conjunto de Julia é o limite do domínio de atração de cada raiz. Portanto, como estudado na seção anterior o conjunto de Julia contém todas as portas para a determinação de todas as raízes de uma equação não linear. Em outras palavras, encontrando um ponto que pertence ao conjunto de Julia da função Newton-Raphson (NR) e iterando em uma pequena vizinhança desse ponto, todas as raízes são alcançáveis. Assim sendo, usando a teoria da iteração, temos a garantia de encontrar todas as raízes, porque o teorema de Montel afirma que as vizinhanças dos pontos em $J(f)$ estão espalhadas pelo plano complexo por iterações da função NR. A questão que surge agora é como encontrar um ponto que pertence ao conjunto de Julia. Para responder a essa pergunta, devemos ter em conta que o conjunto de Julia é o fecho de todos os pontos periódicos repulsores de f (**Definição 59**). Portanto, devemos procurar um dos pontos periódicos repulsores de qualquer período, ou aínda que o conjuto de Julia é a fronteira da bacia de atração. Tendo em conta as duas definições equivalentes de conjunto de Julia, teremos dois critérios para determinar pontos que pertencem ao conjunto de Julia, nas subseções seguintes, veremos métodos para determinar pontos que pertencem ao conjunto de Julia.

5.4.1 Encontrar pontos de Julia através de um ponto repulsor (instável) de período 2

O método de Newton-Raphson oferece pontos que pertencem a um conjunto de Julia e pode ser facilmente encontrado. Para entender isso, vimos que uma das propriedades

do conjunto de Julia de uma função, é de ser invariante sob iteração da função, neste caso, a função NR. Isso significa que todos os pontos repulsores de NR também pertencem ao conjunto de Julia. O objetivo deste método é encontrar um ponto repulsor com período 2. A escolha de buscar um ponto repulsor com período 2 se relaciona diretamente com a natureza dos conjuntos de Julia e a propriedade dos pontos repulsores em sistemas dinâmicos. Nos sistemas dinâmicos, um ponto repulsor é um tipo de ponto fixo que "repele"trajetórias que estão próximas dele. Em outras palavras, uma pequena perturbação de um ponto repulsor resultará em uma trajetória que se afasta desse ponto. No contexto do método de Newton-Raphson, estamos particularmente interessados em pontos repulsores porque eles tendem a definir as regiões mais "interessantes"ou "complexas"do conjunto de Julia. O "período 2"refere-se ao fato de que estamos buscando pontos que retornam ao seu valor original após duas iterações da função. Esse é um caso específico de um ponto periódico, que é um ponto que retorna ao seu valor original após um número fixo de iterações. No caso de um ponto com período 2, ele retorna ao seu valor original após duas iterações da função. Portanto, a busca por um ponto repulsor de período 2 é uma maneira de identificar pontos que são particularmente interessantes ou significativos no conjunto de Julia. Esses pontos podem ser usados para entender melhor a estrutura e a complexidade desses conjuntos. Encontrar um ponto repulisor de período 2, é matematicamente igual a encontrar uma solução para a seguinte equação com restrição [30]:

$$N_f^2(x) = x \tag{5.7}$$

$$\text{Sujeita a } \left|(N_f^2)'(x)\right| > 1, \tag{5.8}$$

onde a equação 5.7 procura por um ponto que, após duas iterações da função de Newton-Raphson (dada por $N_f(x_k) = x_k - \frac{f(x_k)}{f'(x_k)}$), retorna ao seu próprio valor, portanto, para um ponto ser fixo sob duas iterações da função NR, ele deve satisfazer a equação $N_f^2(x) = x$ e a restrição dada pela equação 5.8 garante que x é um ponto repulsor. Assim, primeiro devemos determinar uma das raízes de $f(x)$ (por exemplo, pelo método de Newton-Raphson), partimos de um valor inicial x_0 e iteramos a equação $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$ até a convergência, buscando um valor x para o qual a função $f(x)$ é igual a zero. Em seguida, construir $N_f^{*2}(x_k)$ e $\varphi^*(x_k)$ como:

$$N_f^{*2}(x_k) = x_k - \mu(x_k - N_f^2(x_k)), \tag{5.9}$$

$$\varphi^*(x_k) = x_k + \frac{x_k - N_f^{*2}(x_k)}{(N_f^{*2})'(x_k)}, \tag{5.10}$$

onde $0 < \mu < 1$ é um parâmetro de controle. Agora iterar $\varphi^*(x_k)$ com a raiz determinada como valor inicial. Para as equações (5.9) e (5.10), primeiramente, vale lembrar que o método de Newton-Raphson é uma maneira de encontrar as raízes de uma função. Neste caso, estamos interessados em encontrar os pontos do conjunto de Julia que são repulsores com período 2. Para isso, criamos uma nova função $N_f^2(x)$ que corresponde a duas iterações da função de Newton-Raphson, ou seja:

$$N_f^2(x) = x - \frac{f(x - \frac{f(x)}{f'(x)})}{f'(x - \frac{f(x)}{f'(x)})}. \tag{5.11}$$

A equação 5.7, é então usada para encontrar os pontos que, após duas iterações, retornam ao seu valor original.

A equação 5.9, é uma modificação da função $N_f^2(x)$ onde introduzimos um parâmetro de controle μ, com $0 < \mu < 1$, que serve para controlar a velocidade de convergência do método.

A função $\varphi^*(x_k)$ da equação 5.10, é a função que vamos iterar para encontrar o ponto fixo repulsor com período 2. Esta função é uma versão modificada da função de Newton-Raphson que, em vez de procurar a raiz da função $f(x)$, está procurando a raiz da função $N_f^{*2}(x)$. A fórmula específica para $\varphi^*(x_k)$ é derivada da fórmula original do método de Newton-Raphson, substituindo $f(x)$ por $N_f^{*2}(x)$ e ajustando o sinal para procurar um ponto repulsor em vez de um ponto atrator. Portanto, as fórmulas dadas pelas equações 5.9 e 5.10 são modificações do método de Newton-Raphson original projetadas para encontrar pontos repulsores com período 2 no conjunto de Julia.

5.4.2 Encontrar pontos de Julia por vizinhança

Quando o método de Newton-Raphson é aplicado a polinômios, pode-se mostrar que infinito é um ponto fixo repulsor ($f(\infty) = \infty$) e, portanto, pertence ao conjunto de Julia. Agora, todos os pontos que iteram até o infinito também devem pertencer ao conjunto de Julia. Esses pontos são chamados de pontos críticos e ocorrem onde a inclinação do polinômio é zero. Considerando o exemplo da equação em (5.6) anterior, temos de encontrar os pontos críticos resolvendo $\frac{dg(z)}{dz} = 3z^2 - 1 = 0$, onde temos os zeros $z_1 = \frac{1}{\sqrt{3}}$ e $z_2 = -\frac{1}{\sqrt{3}}$, como

$$N_f(z_n) = z_n - \frac{z_n^3 - z_n}{3z_n^2 - 1}$$

Assim, teremos que

$$N_f(\pm\frac{1}{\sqrt{3}}) = \infty.$$

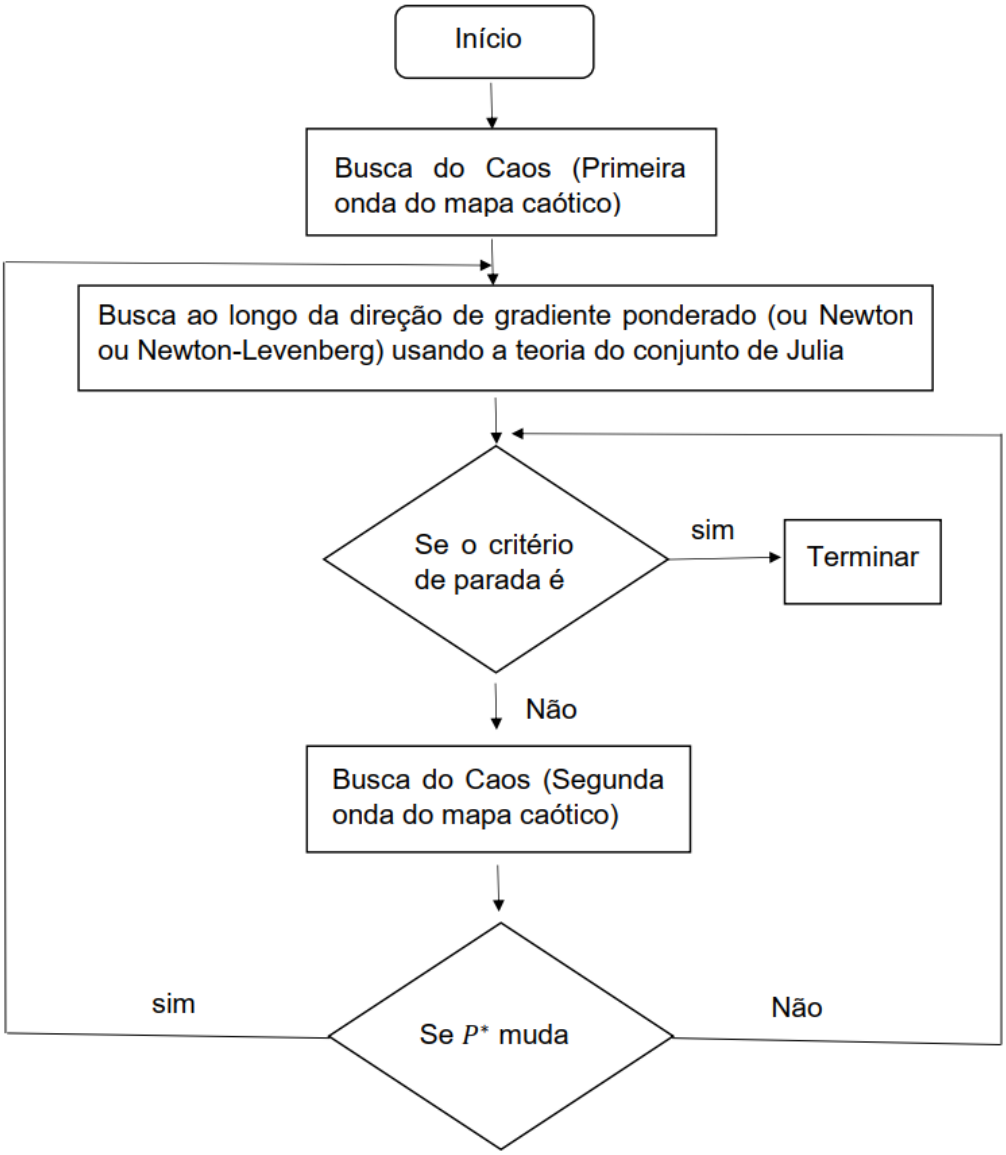
Portanto, ambos pontos pertencem ao conjunto de Julia. No caso de resolvermos funções não polinomiais, não se pode afirmar que o infinito é um ponto fixo repulsor, visto que se considerarmos a função tangente, temos que $tan(\infty)$ não existe. Portanto, para encontrar as raízes de $g(x) = 0$ precisamos apenas examinar a vizinhança de uma das raízes de $dg(x)/dx = 0$.

Resumidamente, para encontrar todos os zeros de uma equação não linear $f(x) = 0$, temos de encontrar uma solução x_c de $f'(x) = 0$ e usar valores na região $|x - x_c| \leq \epsilon$, onde ϵ é um número positivo arbitrário e pequeno, utilizado como valor inicial do processo iterativo.
A solução x_c estar no conjunto de Julia garante que todas as raízes de uma equação $f(x) = 0$ serão alcançáveis por este procedimento. Para encontrar uma solução de $f'(x) = 0$ estão disponíveis várias técnicas numéricas, por exemplo o próprio método de Newton-Raphson.

5.5 Algoritmo de otimização e caos via conjunto de Julia

O algoritmo de caos descrito na **Seção 5.2**, não é muito efetivo, visto que não tem ferramentas suficientes para determinar todos os zeros de $f(x) = 0$ ou $f'(x) = 0$. Assim sendo, nesta seção, mudamos o algoritmo de caos, usando a teoria do conjunto de Julia, que nos auxiliará a determinar todos os zeros de interesse. Assim sendo, o fluxograma a seguir (Figura 5.4) ilustra o funcionamento do Algoritmo.

Figura 5.4: Fluxograma do algoritmo de otimização de caos ao longo da direção do gradiente ponderado, ou no método de Newton ou de Newton-Levenberg via teoria de Julia



fonte: [30]

Assim sendo, temos as seguintes etapas para o algoritmo:

(i) Pesquisa de caos usando a primeira onda do mapa caótico:

Etapa1: Inicialize o número da primeira busca de caos S_1, o número da segunda busca de caos S_2, parâmetro de penalidade σ, valor inicial da variável de caos $0 < \gamma_i^{(0)} < 1$ $(i = 1, 2, ..., n)$ que possuem pequenas diferenças, ajustando o parâmetro para pequenas faixas ergódicas em torno da solução $\alpha_i > 0$ $(i = 1, 2, ..., n)$ e ajustando o parâmetro para $\alpha t_3 > 1$.

Etapa 2: Defina $l = 0$ e $P^* = \infty$.

Etapa 3: Mapeia as variáveis de caos $\gamma_i^{(l)}$, $i = 1, 2, ..., n$, no intervalo de variação das variáveis de otimização pela seguinte equação:

$$x_i^{(l)} = a_i + \gamma_i^{(l)}(b_i - a_i);\ \ x^{(l)} = [x_1^{(l)}, x_2^{(l)}, ..., x_n^{(l)}]^T.$$

Etapa 4: Se $P(x^{(l)}, \sigma) \leq P^*$ então $x^* = x^{(l)}$ e $P^* = P(x^{(l)}, \sigma)$.

Etapa 5: Gerar o próximo valor da variável de caos pelo mapa caótico da função (M):

$$\gamma_i^{(l+1)} = M(\gamma^{(l)})$$

.

Etapa 6: Se $l < S_1$, $l \leftarrow l + 1$ vá para o passo 3, senão pare o primeiro processo de busca de caos.

(ii) Pesquise ao longo do método de gradiente ponderado, ou no método de Newton ou de Newton-Levenberg usando a teoria de Julia

Etapa 1: Calcular $d(x^*)$ de (4.30)

Etapa 2: Determinar o valor λ que minimiza a função $P(\lambda) = P(x^* + \lambda d(x^*), \sigma)$. Em outras palavras, encontrar todas as soluções da equação $P'(\lambda) = 0$. Para encontrar as soluções, primeiro localiza-se um ponto do conjunto de Julia, e então, usa-se o método de Newton-Raphson para iterar numa vizinhança ϵ ao redor deste ponto. Se não haver mínimos, parar nesta etapa.

Etapa 3: Entre todas as soluções determinadas na etapa 2, encontrar λ que melhor minimiza $P(x^* + \lambda d(x^*), \sigma)$, em seguida definir $x^* = x^* + \lambda d(x^*)$ e $P^* = P(x^*, \sigma)$

Etapa 4: Ir para etapa 1 até que $d(x^*)$ se torna suficientemente pequeno.

(iii) Busca de caos usando a segunda onda do mapa caótico

Etapa 1: $l' = 0$.

Etapa 2: Gerar os próximos valores das variáveis de caos pela função de mapa caótico (M):

$$\gamma_i^{(l+1)} = M(\gamma_i^{(l)}),\ \ i = 1, 2, ..., n$$

.

Etapa 3: $l \leftarrow l + 1$.

Etapa 4: $x_i^{(l)} = x_i^* + \alpha_i(\gamma_i^{(l)} - 0.5)$.

Etapa 5: Se $P(x^{(l)}, \sigma) \leq P^*$ então $x^* = x^{(l)}$, $P^* = P(x^{(l)}, \sigma)$ e pare.

Etapa 6: Se $l' < S_2$, $l' \leftarrow l' + 1$ e vá para o passo 2.

Etapa 7: $\alpha_i \leftarrow t_3\alpha_i$ e pare o segundo processo de busca de caos.

Esse algoritmo objetiva melhorar o Algoritmo de Otimização e Caos.

Portanto, os algoritmos AOC e AOC via teoria de Julia, apresentam uma abordagem inovadora e poderosa para resolver problemas complexos multimodais. Ao unir os conceitos dos fractais, que demonstram como regras simples podem gerar padrões complexos, e do caos, que enfatiza a sensibilidade às condições iniciais e a imprevisibilidade dos sistemas, juntamente com a teoria de Julia, que é de capital importância para nosso estudo, esses métodos desafiam a compreensão tradicional da otimização. Eles oferecem uma alternativa promissora aos métodos convencionais de gradiente, permitindo explorar amplamente o espaço de soluções em busca de melhores resultados em problemas complexos multimodais. A inclusão da teoria de Julia abre novas possibilidades para a determinação de todos os zeros de um problema de otimização não linear, impulsionando na melhoria dos métodos de otimização. Essa abordagem inovadora abre caminho para soluções mais eficientes e promissoras, permitindo a descoberta de pontos de mínimo ótimos em problemas complexos multimodais.

No próximo capítulo (Capítulo 6), iremos explorar os resultados obtidos por meio do Algoritmo de Otimização e Caos. Serão apresentadas análises para dois exemplos simples que mostram a possibilidade de aplicar esse método na resolução de problemas complexos

multimodais e dentre outros. Além disso, faremos uma comparação dos resultados com a bibliografia consultada, bem como a trajetória de pontos durante a execução do algoritmo. Ao examinar os resultados numéricos, visualizamos a importância desses algoritmos e sua possível aplicabilidade em uma ampla gama de problemas de otimização não linear.

Capítulo

6

Resultados Numéricos

Neste capítulo são apresentados resultados numéricos de dois problemas, que foram implementados em MATLAB, resolvidos pelo Algoritmo de Otimização e Caos. O problema 1 é o problema do despacho econômico, com três geradores, cuja referência é [28], e o problema 2 é um problema de duas variáveis, cuja referência é [30].

6.1 Problema do Despacho Econômico (PDE)

O Problema do despacho econômico é formulado da seguinte forma:

$$\begin{aligned} \text{minimize} \quad & f(P_G) = \sum_{i=1}^{n}(a_i P_{G,i}^2 + b_i P_{G,i} + c_i) + \sum_{i=1}^{n} |e_i sen(f_i(P_{G,i}^{min} - P_{G,i}))| \\ \text{sujeito a} \quad & \sum_{i=1}^{n} P_{G,i} = P_d \\ & P_{G,i}^{min} \leq P_{G,i} \leq P_{G,i}^{max} \\ & i = 1, ..., n. \end{aligned} \tag{6.1}$$

O problema do despacho econômico é um problema de otimização que é amplamente estudado na área de engenharia elétrica. Ele consiste em determinar a melhor distribuição de potência gerada pelas unidades geradoras de energia elétrica, de forma a atender à demanda de energia de maneira eficiente, levando em consideração critérios econômicos. Em outras palavras, o objetivo do despacho econômico é encontrar a alocação ideal de potência para cada unidade geradora, de modo a minimizar os custos de operação do sistema elétrico. Isso envolve decidir quanta potência cada unidade geradora deve fornecer, considerando fatores como os custos de geração de energia, limites de potência e a demanda total de energia. Ao otimizar a alocação de potência, busca-se minimizar os custos de operação do sistema elétrico, como o consumo de combustível influenciado pelo desempenho dos geradores (limites de potências e carregamento de pontos de válvula), além de maximizar a eficiência geral do sistema. Dessa forma, o despacho econômico visa encontrar um equilíbrio entre a oferta e a demanda de energia elétrica, levando em conta os aspetos econômicos envolvidos. A formulação apresentada pela equação (6.1) descreve o problema do despacho econômico com algumas considerações específicas. A função objetivo, representada por $f(P_G)$, busca minimizar o custo total de operação do sistema elétrico. Essa função é composta por duas partes: A primeira parte é uma função quadrática que representa o custo de geração de energia das unidades geradoras, e

a segunda parte é uma função que considera a influência do carregamento de pontos de válvula associada a funções valores absolutos senoidais que considera os desvios entre a potência gerada e os limites mínimo e máximo de geração.

A primeira parte da função objetivo é uma soma ponderada dos custos de geração das unidades geradoras. Cada unidade geradora é representada pelo índice i e possui coeficientes a_i, b_i e c_i que modelam o custo de geração. O termo $P_{G,i}$ representa a potência gerada pela unidade geradora i. Portanto, o produto $a_i P_{G,i}^2 + b_i P_{G,i} + c_i$ representa o custo de geração da unidade geradora i. A segunda parte da função objetivo envolve uma função associada ao carregamento de pontos de válvula dos geradores. Ela é composta por uma soma ponderada de termos absolutos dos desvios entre a potência gerada e os limites mínimo e máximo de geração. Esses desvios são representados pelo termo $P_{G,i}^{min} - P_{G,i}$, onde $P_{G,i}^{min}$ é a potência mínima de geração permitida para a unidade geradora i. O coeficiente e_i representa a perda de eficiência quando cada uma das, normalmente, quatro válvulas de controle se encontra estrangulando parcialmente e o fluxo da i-ésima turbina. A expressão $sen(f_i(P_{G,i}^{min} - P_i))$ é uma função que modela o carregamento de pontos de válvula. As restrições do problema estão representadas pela equação 6.1. Elas impõem que a soma das potências geradas de todas as unidades geradoras deve ser igual à demanda de energia P_d. Além disso, impõem que a potência gerada por cada unidade geradora $P_{G,i}$ deve estar dentro dos limites mínimo e máximo $P_{G,i}^{min}$ e $P_{G,i}^{max}$, de potência, respetivamente. O objetivo do problema do despacho econômico é encontrar os pontos de carga ótimos das potências geradas Pg_i que minimizem o custo total de operação do sistema elétrico, sujeito às restrições de geração e demanda. O **Problema 1** definido a seguir é uma aplicação do PDE que considera o problema modelado para 3 geradores térmicos.

Problema 1: PDE com 3 geradores, problema este que foi retirado de [28], com os dados na Tabela:

Tabela 6.1: PDE - Três geradores

Gerador (i)	$P_{G,i}^{Min}$ (MW)	$P_{G,i}^{Max}$ (MW)	a_i $\$/(MW^2)$	b_i \$/(MW)	c_i \$	e_i \$	f_i $(1/MW)$
1	100	600	0,001562	7,92	561	300	0,0315
2	50	200	0,00482	7,97	78	150	0,063
3	100	400	0,00194	7,85	310	200	0,042
Demanda: D = 850 MW							

Fonte: [28]

Para a formulação do problema, tendo em conta o problema dado em (6.1), assim teremos o nosso problema formulado como:

$$
\begin{aligned}
\text{minimizar } f(P_G) = \quad & 0.001562P_{G,1}^2 + 7.92P_{G,1} + 561 + |300\sin(0.0315(100 - P_{G,1}))| \\
& + 0.00482P_{G,2}^2 + 7.97P_{G,2} + 78 + |150\sin(0.063(50 - P_{G,2}))| \\
& + 0.00194P_{G,3}^2 + 7.85P_{G,3} + 310 + |200\sin(0.042(100 - P_{G,3}))| \\
\text{sujeito a:} \quad & P_{G,1} + P_{G,2} + P_{G,3} = 850 \\
& 100 \leq P_{G,1} \leq 600 \\
& 50 \leq P_{G,2} \leq 200 \\
& 100 \leq P_{G,3} \leq 400.
\end{aligned}
\tag{6.2}
$$

Para a representação gráfica da função objetivo, é necessário considerar o **Problema 1** expresso em função de duas variáveis, assim sendo, a variável $P_{G,3}$ será redefinida pela diferença entre a demanda Pd e as variáveis $P_{G,1}$ e $P_{G,2}$, isto é, $P_{G,3} = 850 - P_{G,1} - P_{G,2}$, logo o problema pode ser reescrito como:

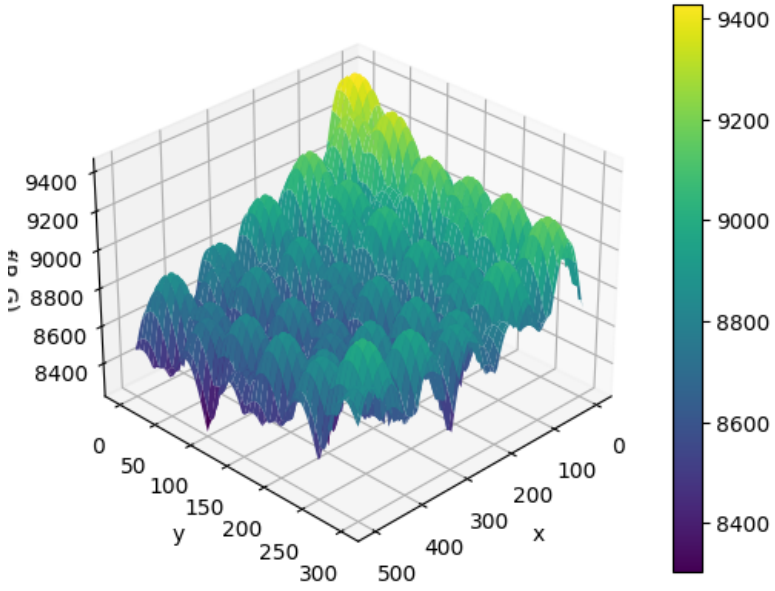
$$\begin{aligned}
\text{minimizar } f(P_G) = \quad & 0.001562P_{G,1}^2 + 7.92P_{G,1} + 561 + |300\sin(0.0315(100 - P_{G,1}))| \\
& + 0.00482P_{G,2}^2 + 7.97P_{G,2} + 78 + |150\sin(0.063(50 - P_{G,2}))| \\
& + 0.00194(850 - P_{G,1} - P_{G,2})^2 + 7.85(850 - P_{G,1} - P_{G,2}) + 310 \\
& + |200\sin(0.042(100 - (850 - P_{G,1} - P_{G,2})))| \\
\text{sujeito a:} \quad & P_{G,3} = 850 - P_{G,1} - P_{G,2} \\
& 100 \leq P_{G,1} \leq 600 \\
& 50 \leq P_{G,2} \leq 200 \\
& 100 \leq P_{G,3} \leq 400.
\end{aligned} \tag{6.3}$$

Para possibilitar a resolução de (6.3), é necessário que a função seja suavizada, para que seja diferenciável. Para isso consideramos a estratégia de suavização hiperbólica definida na Seção 3.2.4.1. Ressalta-se que a função depois de suavizada, os pontos de minimização do problema não se alteram e podem ser determinados à medida que o parâmetro de suavização η tende a zero. Assim, suavizando a função objetivo, teremos:

$$\begin{aligned}
\text{minimizar } f(P_G) = \quad & 0.001562P_{G,1}^2 + 7.92P_{G,1} + 561 + \sqrt{(300\sin(0.0315(100 - P_{G,1})))^2 + \eta} \\
& + 0.00482P_{G,2}^2 + 7.97P_{G,2} + 78 + \sqrt{(150\sin(0.063(50 - P_{G,2})))^2 + \eta} \\
& + 0.00194(850 - P_{G,1} - P_{G,2})^2 + 7.85(850 - P_{G,1} - P_{G,2}) + 310 \\
& + \sqrt{(200\sin(0.042(100 - (850 - P_{G,1} - P_{G,2}))))^2 + \eta} \\
\text{sujeito a:} \quad & P_{G,3} = 850 - P_{G,1} - P_{G,2} \\
& 100 \leq P_{G,1} \leq 600 \\
& 50 \leq P_{G,2} \leq 200 \\
& 100 \leq 850 - P_{G,1} - P_{G,2} \leq 400.
\end{aligned} \tag{6.4}$$

Assim sendo, teremos a Figura 6.1, que representa graficamente a função do PDE suavizado, considerando $\eta = 10$, implementado no python (Apêndice A.14).

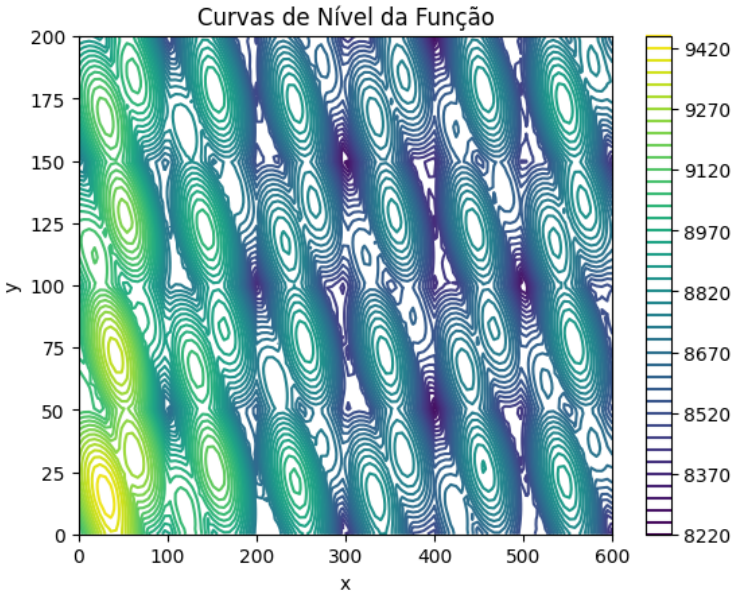
Figura 6.1: PDE suavizado - ($\eta = 10$)



Fonte: Autor

A seguir, temos as curvas de nível do PDE suavizado, ilustrado na Figura 6.2, cuja implementação em python, está no Apêndice A.15

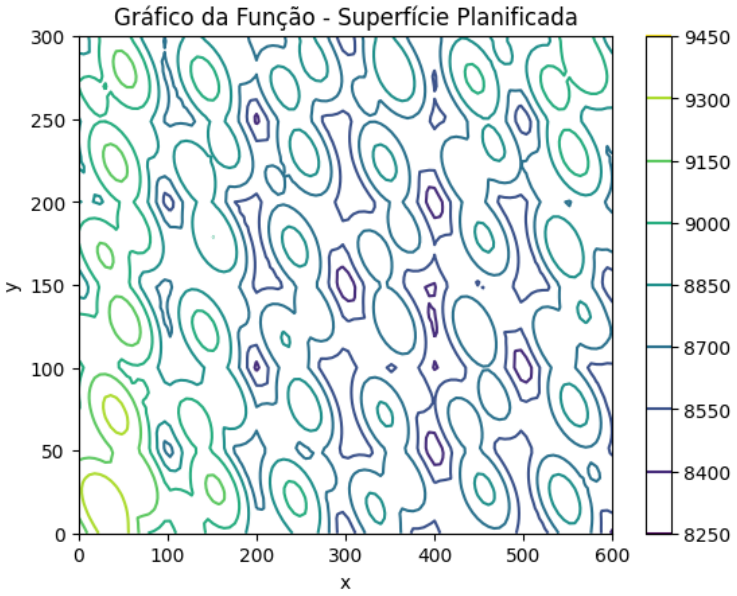
Figura 6.2: PDE suavizado - curvas de nível ($\eta = 10$)



Fonte: Autor

A seguir temos a Figura 6.3, implementada em python (Apêndice A.17), que representa a função planificada.

Figura 6.3: PDE Planificada ($\eta = 10$)



Fonte: Autor

Para resolver o problema usando o AOC, é necessário considerar o problema dado suavizado, e escolher um dos métodos de Barreira ou Penalização, para reescrever o problema como irrestrito. Para tal, vamos usar a estratégia de [30], isto é, usando o método de penalização. Assim sendo, considerando o problema (6.2) suavizado e com restrições de desigualdade tem-se:

$$
\begin{aligned}
\text{minimizar } f(P_G) = \quad & 0.001562P_{G,1}^2 + 7.92P_{G,1} + 561 + \sqrt{(300\sin(0.0315(100 - P_{G,1})))^2 + \eta} \\
& + 0.00482P_{G,2}^2 + 7.97P_{G,2} + 78 + \sqrt{(150\sin(0.063(50 - P_{G,2})))^2 + \eta} \\
& + 0.00194P_{G,3}^2 + 7.85P_{G,3} + 310 + \sqrt{(200\sin(0.042(100 - P_{G,3})))^2 + \eta} \\
\text{sujeito a:} \quad & P_{G,1} + P_{G,2} + P_{G,3} - 850 \leq 0 \\
& - P_{G,1} - P_{G,2} - P_{G,3} + 850 \leq 0 \\
& - P_{G,1} + 100 \leq 0 \\
& P_{G,1} - 600 \leq 0 \\
& - P_{G,2} + 50 \leq 0 \\
& P_{G,2} - 200 \leq 0 \\
& - P_{G,3} + 100 \leq 0 \\
& P_{G,3} - 400 \leq 0.
\end{aligned}
\tag{6.5}
$$

Assim, construindo a função de penalização quadrática, penalizando a função objetivo e reescrevendo o Problema (6.5) como um problema irrestrito tem-se:

$$
\begin{aligned}
\text{minimizar } P(P_G,\sigma) = \quad & 0.001562P_{G,1}^2 + 7.92P_{G,1} + 561 + \sqrt{(300\sin(0.0315(100-P_{G,1})))^2+\eta} \\
& + 0.00482P_{G,2}^2 + 7.97P_{G,2} + 78 + \sqrt{(150\sin(0.063(50-P_{G,2})))^2+\eta} \\
& + 0.00194P_{G,3}^2 + 7.85P_{G,3} + 310 + \sqrt{(200\sin(0.042(100-P_{G,3})))^2+\eta} \\
& + \sigma((P_{G,1}+P_{G,2}+P_{G,3}-850)^2 + (-P_{G,1}-P_{G,2}-P_{G,3}+850)^2 + \\
& (-P_{G,1}+100)^2 + (P_{G,1}-600)^2 + (-P_{G,2}+50)^2 + (P_{G,2}-200)^2 \\
& (-P_{G,3}+100)^2 + (P_{G,3}-400)^2) \\
\text{sujeito a:} \quad & P_{G,i} \in \mathbb{R}^3, i = 1,2,3.
\end{aligned}
\tag{6.6}
$$

Onde σ é o parâmetro de penalidade, que foi representado pela letra c no **Capítulo 3**. O problema (6.6) escrito como irrestrito, já pode ser resolvido pelo AOC. No caso do problema (6.2), poderíamos usar a barreira logaritmica, que teriamos de resolver o problema (6.7), descrito como:

$$
\begin{aligned}
\text{minimizar } P(P_G,\sigma) = \quad & 0.001562P_{G,1}^2 + 7.92P_{G,1} + 561 + \sqrt{(300\sin(0.0315(100-P_{G,1})))^2+\eta} \\
& + 0.00482P_{G,2}^2 + 7.97P_{G,2} + 78 + \sqrt{(150\sin(0.063(50-P_{G,2})))^2+\eta} \\
& + 0.00194P_{G,3}^2 + 7.85P_{G,3} + 310 + \sqrt{(200\sin(0.042(100-P_{G,3})))^2+\eta} \\
& - \frac{1}{\sigma}[\ln\ (-P_{G,1}-P_{G,2}-P_{G,3}+850) + \ln\ (P_{G,1}+P_{G,2}+P_{G,3}-850) + \\
& \ln\ (P_{G,1}-100) + \ln\ (-P_{G,1}+600) + \ln\ (P_{G,2}-50) + \ln\ (-P_{G,2}+200) \\
& + \ln\ (P_{G,3}-100) + \ln\ (-P_{G,3}+400)] \\
\text{sujeito a:} \quad & P_{G,i} \in \mathbb{R}, i = 1,2,3.
\end{aligned}
\tag{6.7}
$$

Para a resolução dos problemas irrestritos (6.6), pelo método AOC e método de gradientes ponderados ou (6.7), pelo método AOC e método de Newton (ou Newton-Levenberg), vistos no **Capítulo 3** e realizar a interpretação geométrica desses problemas em R^2, a solução em potência $P_{G,3}$ é substituída por $P_{G,3} = Pd - P_{G,1} - P_{G,2}$, em ambas as formulações. A implementação dos algoritmos considerados à resolução desses são apresentados no **Apêndice A**.

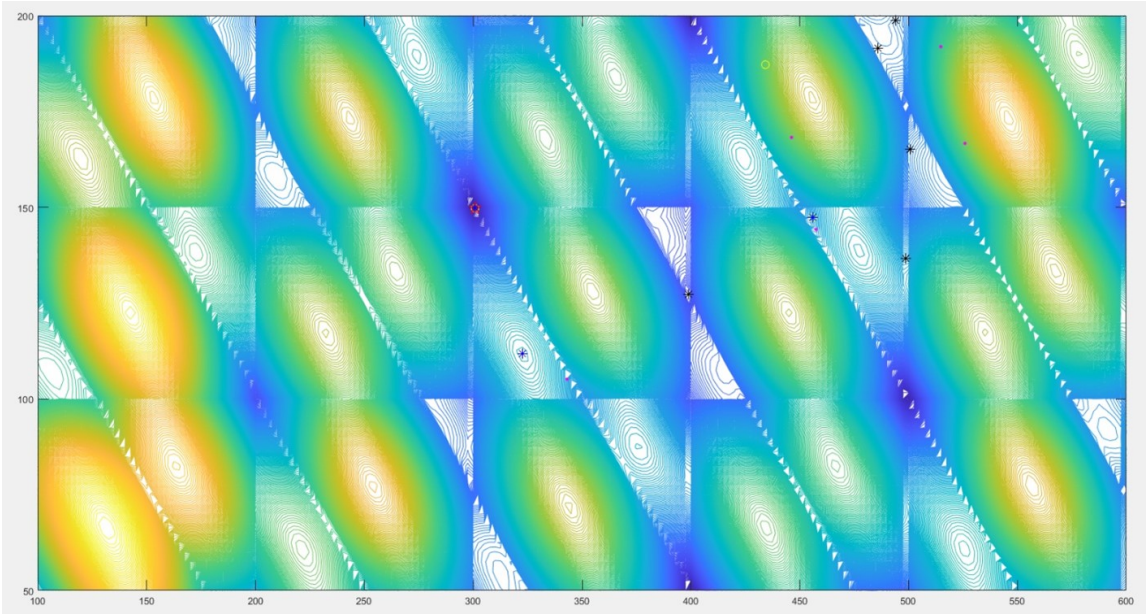
Na Figura 6.2, vemos os múltiplos pontos de mínimo do PDE, onde o azul mais carregado representa as zonas onde há minimizantes. Para a determinação dos pontos de mínimo do PDE com 3 geradores [28], foi usado o método misto AOC (Algoritmo de Otimização e Caos) e de Newton. Com este método, a solução ótima obtida é $P_{(G,1)} = 300; P_{(G,2)} = 150,02$ e $P_{(G,3)} = 399,8$ cujo melhor valor de mínimo é 8234.4, que será apresentado em detalhes na Tabela 6.3.

6.1.1 Trajetória de pontos para obtenção do melhor ponto de mínimo do PDE

Foram realizados 5 ciclos do método e atualizando os pontos e valores de mínimo obtidos até determinar a melhor solução ao PDE. Na trajetória de pontos, apresentados na Figura 6.4, temos:

- Em magenta os pontos iniciais obtidos na primeira onda do método pelo método de caos;
- Em preto as soluções obtidas pelo método de Newton próximas aos pontos obtidos na primeira onda;
- Em amarelo o ponto obtido na segunda Onda – método de caos;
- Em azul as soluções obtidas após a segunda Onda;
- Em estrela em vermelho a melhor solução obtida pelo método com custo de \$8234.5 e potências $P = (300.8225, 149.7591, 399.4184)$, com demanda de $850MW$.

Figura 6.4: Trajetória de pontos para obtenção do melhor ponto de mínimo do PDE



Fonte: Autor

6.1.2 Comparação Dos Resultados Com A Bibliografia Consultada

A tabela 6.2, mostra os diferentes valores de mínimos obtidos durante o processo iterativo, variando os diferentes parâmetros do AOC, como variáveis de caos e tamanho do passo. Com essa variação, foram obtidos os diferentes valores de mínimos como descritos.

Tabela 6.2: Resultados numéricos do Algoritmo de Otimização e Caos, para o Problema 1

Problema 1	Pior Valor Mínimo (\$)	Valor Mínimo Intermediário (\$)	Melhor Valor Mínimo	Melhor Solução (KW)
PDE	8375.6	8241.7	8234.4**	(300.8225, 149.7591, 399.4184)

Para o PDE, comparando-se a solução indicada por $**$ com a solução obtida em [28], que utilizou um método de pontos interiores/exteriores o valor mínimo obtido foi de 8234.5. A solução ótima obtida: $P_{(G,1)} = 300; P_{(G,2)} = 150,02$ e $P_{(G,3)} = 399,8$. Estes resultados

foram obtidos pela programação Matlab, cujos detalhes se encontram no Apêndice A.6. Nesse caso, além de valores mínimos piores e intermediários, o algoritmo da Seção 5.2 determinou a melhor solução de mínimo relativa ao PDE, também determinada em [28].

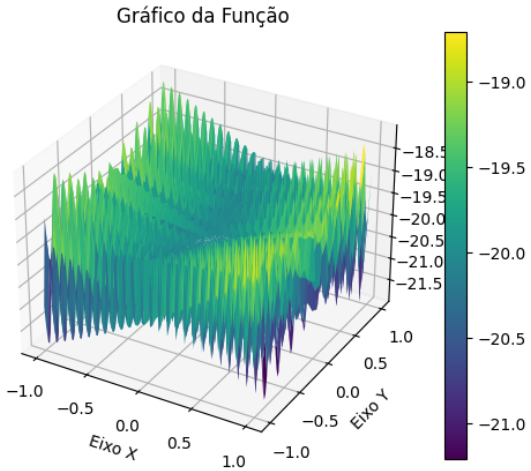
6.2 Problema 2

O problema (6.8), é um problema de minimização com restrição de desigualdade e variáveis canalizadas, que foi extraído de [30], cuja formulação é a seguinte:

$$\begin{gathered} \text{minimizar } f(x,y) = -[xsen(9\pi y) + ycos(25\pi x) + 20] \\ \text{Sujeito a: } x^2 + y^2 \leq 81 \\ x, y \in [-10, 10] \end{gathered} \tag{6.8}$$

Para melhorar a resolução do problema, a Figura 6.5 implementada no python (Apêndice A.9), representa o gráfico da função objetivo do Problema 2.

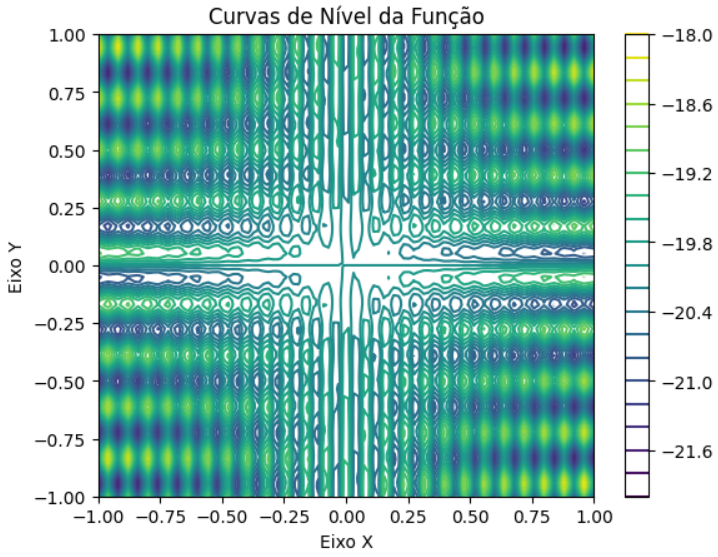
Figura 6.5: Gráfico da Função do Problema 2



Fonte: Autor

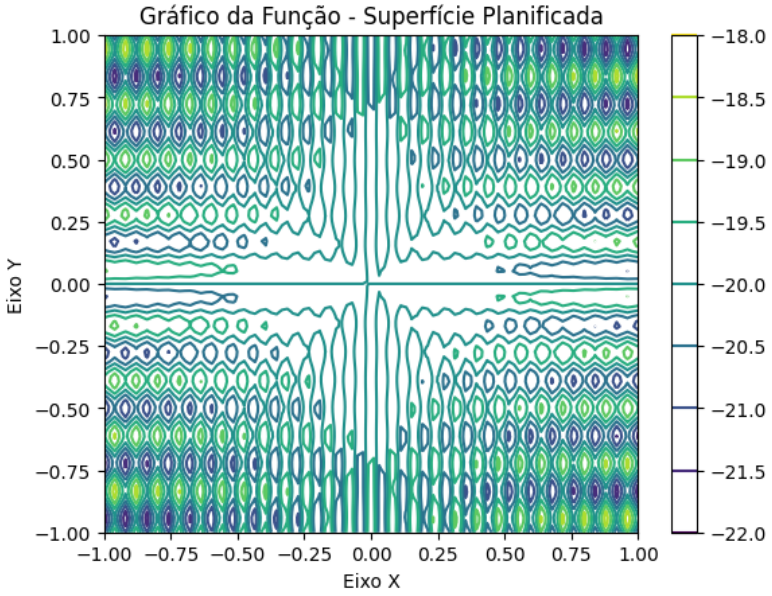
A seguir, temos a Figura 6.6, implementada no python (Apêndice A.11), que representa as curvas de nível da função objetivo e a Figura 6.7, implementada no python (Apêndice A.10), que representa a função planificada.

Figura 6.6: Curvas de nível do Problema 2



Fonte: Autor

Figura 6.7: Função Planificada do Problema 2



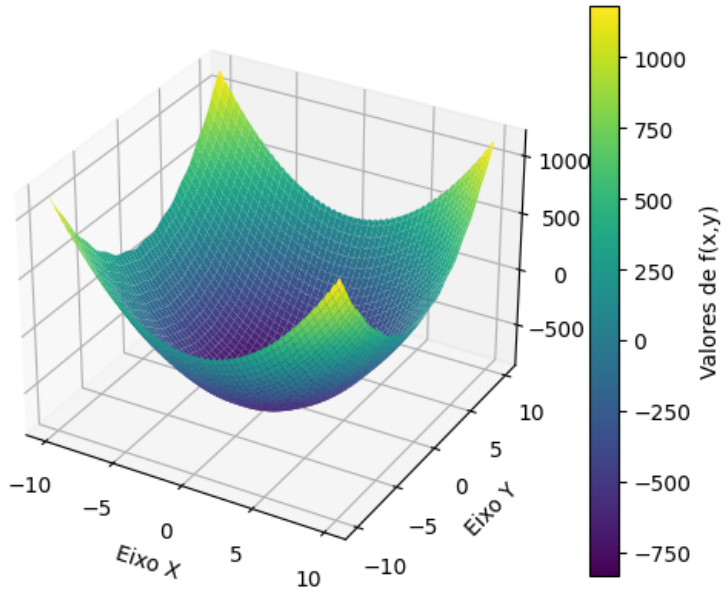
Fonte: Autor

Usando o método de penalização visto no **Capítulo 3**, o Problema (6.8) é reescrito como um problema irrestrito expresso em (6.9):

$$\begin{aligned} \text{minimizar } P((x,y),\sigma) = -[xsen(9\pi y) + ycos(25\pi x) + 20] + \sigma(x^2 + y^2 - 81) \\ x, y \in \mathbb{R}^2. \end{aligned} \tag{6.9}$$

Assim sendo, temos o gráfico do Problema 2 penalizado representado pela Figura 6.8 (Implementada em python, no Apêndice A.12), considerando o parâmetro de penalidade $\sigma = 10$.

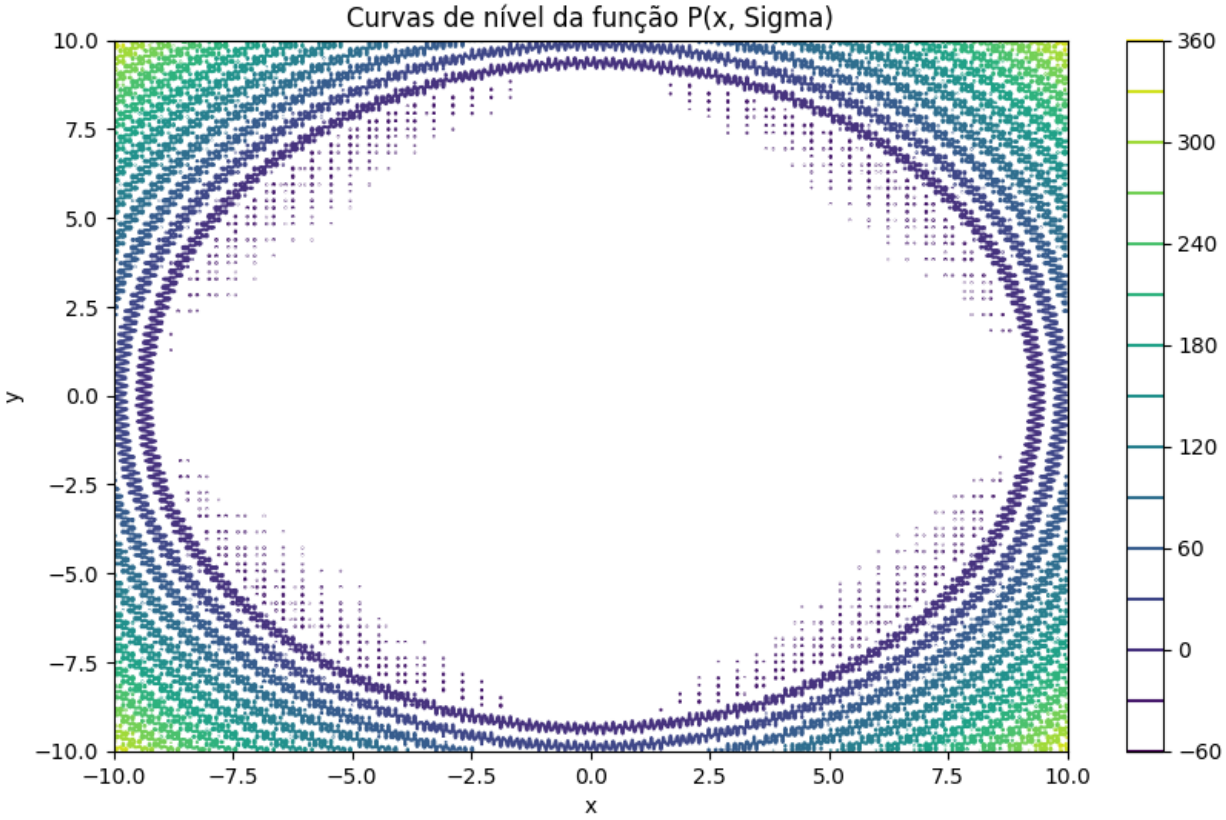
Figura 6.8: Representação gráfica do Problema 2 ($\sigma = 10$)



Fonte: Autor

A seguir, temos a Figura 6.9, implementada em python (Apêndice A.4), que representa as curvas de nível do Problema 2 penalizado.

Figura 6.9: Curvas de nível do Problema 2 ($\sigma = 3$)



Fonte: Autor

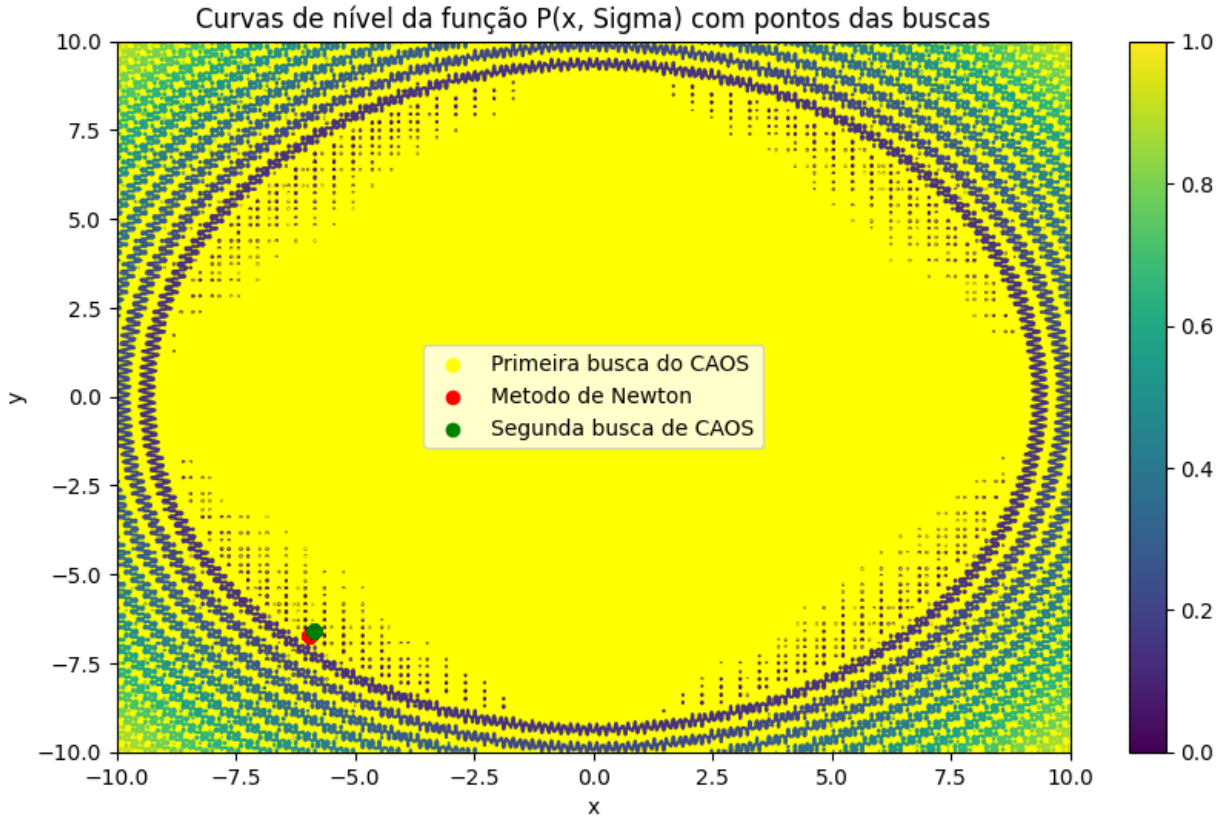
Com base nos contornos mais escuros, isto é, a região mais escura da Figura 6.9, podemos encontrar o ponto de mínimo o Problema 2, que foi resolvido pelo Método AOC com o método de Newton, que determinou como solução (-5.85749, -6.607253) e como valor de mínimo -32.63132945. Esta solução foi determinada pela programação em Python, cujos detalhes se encontram no Apêndice A.4.

6.2.1 Trajetória de pontos para obtenção do melhor ponto de mínimo do Problema 2

Foram realizadas as três etapas do AOC no problema, considerando o valor inicial $\gamma_0 = (0.64, 0.91), S_1 = 1000000, S_2 = 20000$ no método e atualizando os pontos e valores de mínimo obtidos até determinar a melhor solução ao Problema 2. Na trajetória de pontos, apresentados na Figura 6.10, implementada em python (Apêndice A.4), temos:

- Em Amarelo os pontos iniciais obtidos na primeira onda do método pelo método de caos;
- Em Verde os pontos obtidos na segunda onda pelo método de caos;
- Em vermelho as soluções obtidas pelo método de Newton;
- Na bola em vermelho, encontra-se a melhor solução obtida pelo método com função objetivo de -32.63132945 e solução $x = (-5.85749, -6.607253)$, em que $x_1^2 + x_2^2 = 77.966 < 81$. Esta solução foi obtida pela etapa do método de Newton.

Figura 6.10: Trajetória de pontos para obtenção do melhor ponto de mínimo do Problema 2 ($\sigma = 3$)



Fonte: Autor

Observando a Figura 6.10, pelas trajetórias dos pontos, nota-se que a convergência da solução ocorre na região mais escura, isto é, no contorno da circunferência centrada na origem e raio 9. É importante destacar que durante a execução do AOC, na primeira onda do mapa caótico, o número de iterações foi muito elevado devido a convergência na solução obtida, isto é, quanto menor foi o número de iterações, foram obtidos resultados piores.

6.2.2 Comparação dos Resultados Com a Bibliografia Consultada

A tabela 6.3, mostra os diferentes valores de mínimos obtidos durante o processo iterativo, variando os diferentes parâmetros do AOC, como variáveis de caos e tamanho do passo. Com essa variação, foram obtidos os diferentes valores de mínimos como descritos.

Tabela 6.3: Resultados numéricos do Algoritmo de Otimização e Caos, para o Problema 2

Problema	Pior Valor Mínimo	Valor Mínimo Intermediário	Melhor Valor Mínimo	Melhor Solução
2	-19.82274	-32.6128	-32.6313*	(-5.8575, -6.6073)

Com base na bibliografia consultada, comparando os resultados, teremos que:

Para o Problema 2, comparando-se com a solução indicada em (*) obtida em [30] é apresentado um valor mínimo de -32.7179, mas não consta o ponto mínimo $x^* \in \mathbb{R}^2$. Não foi possível obter a solução que determinou esse valor de mínimo em [30]. Possivelmente, com a implementação do algoritmo visto na Seção 5.2, quando a teoria de Julia será incorporada ao algoritmo da Seção 5.2, poderemos determinar uma melhor solução ao Problema 2.

Os resultados obtidos no Problema 1 (PDE) e problema 2, pelo método apresentado na Seçao 5.2 mostram que é necessário que esse método seja aperfeiçoado considerando o método visto na Seçao 5.2, pois esse depende de atualizações nos parâmetros de caos indicados pelos autores em [30], para realizar novos ciclos do método e obter novas e possíveis melhores soluções do que aquelas já obtidas. As Figuras 6.4 e 6.10 procuram mostrar a maleabilidade desse método.

A seguir apresentam-se as conclusões e considerações finais do trabalho desenvolvido, bem como a proposta de etapas possivelmente a serem realizadas como trabalhos futuros após a defesa de dissertação.

Capítulo

7

Conclusões e Considerações Finais

Tendo em conta o presente trabalho, chegamos as seguinte conclusões e considerações finais:

(i) Os problemas de otimização desempenham um papel crucial na sociedade contemporânea, pois a busca por estratégias eficientes que possam auxiliar empresas e outros setores sociais a minimizar custos ou maximizar lucros é uma necessidade constante. Nesse contexto, a disponibilidade de métodos de otimização robustos e eficazes, sejam eles determinísticos, bioinspirados ou híbridos, torna-se essencial para encontrar soluções precisas e de alta qualidade para os problemas de otimização encontrados na literatura.

(ii) O nosso estudo concentrou-se geralmente em problemas de otimização não linear e especificamente em problemas de otimização multimodais, nos quais a presença de múltiplos ótimos locais apresenta desafios adicionais. A abordagem adotada buscou explorar e investigar diferentes técnicas e algoritmos para lidar com essa complexidade, com o objetivo de encontrar soluções que se aproximem dos ótimos globais e superem as limitações dos métodos tradicionais.

(iii) O estudo de sistemas dinâmicos desempenha um papel essencial na compreensão e modelagem de fenômenos complexos ao longo do tempo. Sua aplicação abrange uma ampla gama de áreas do conhecimento e oferece ferramentas para a representação e análise de figuras complicadas, como os fractais, que são essenciais para avançar nosso entendimento em diversas disciplinas. O contínuo aprimoramento e desenvolvimento dessa área de estudo são fundamentais para enfrentar os desafios complexos do mundo contemporâneo e explorar novas fronteiras de conhecimento.

(iv) Num problema de otimização não linear, o Algoritmo de Otimização e Caos é eficiente, mas não é completo, visto que não determina todas as soluções minimizadoras do problema, reduzindo assim a melhor possibilidade de determinar o minimizante que seria o minimizador global do problema. É necessário que sejam feitas atualizações nos parâmetros de caos indicados pelos autores em [22] e realizar novos ciclos do método para obter novas e possíveis melhores soluções do que aquelas já obtidas.

(v) Os fractais de Julia constituem uma valiosa ferramenta para a resolução de problemas de otimização, permitindo a determinação de todas as soluções possíveis, independentemente de sua natureza ou localização no espaço de busca. Essa abordagem visual e detalhada proporciona uma compreensão abrangente do problema de otimização multimodal, auxiliando na seleção das soluções mais adequadas e fornecendo uma base sólida para a tomada de decisões otimizadas.

(vi) O Algoritmo de Otimização e Caos apresentado na Seção 5.2 oferece uma abordagem promissora para resolver problemas de otimização, no entanto, para aprimorar ainda mais sua eficiência e completude, é necessário incorporar a teoria de Julia, transformando-o em um Algoritmo de Otimização e Caos via Julia, onde a incorporação da teoria de Julia no algoritmo, conforme descrito na Seção 5.5, tem o potencial de melhorar significativamente seus resultados, isto é, a aplicação da Teoria de Julia permite uma exploração mais abrangente do espaço de soluções, levando à determinação de um maior número de soluções minimizadoras.

(vii) Ao implementar a Teoria de Julia no algoritmo, teremos uma representação visual detalhada do espaço de soluções, proporcionando uma compreensão mais profunda das estruturas e padrões presentes no problema de otimização. Essa abordagem nos permitirá identificar e analisar todas as soluções possíveis, incluindo o minimizante global, que representa a solução ótima do problema.

(viii) Para trabalhos futuros, uma das próximas etapas envolve a integração da estratégia baseada em fractais de Julia com o AOC e métodos de gradientes, visando aperfeiçoar a capacidade do algoritmo de encontrar soluções ótimas em problemas de otimização não lineares e multimodais de maior dimensão. Essa combinação tem como objetivo desenvolver um algoritmo mais robusto, eficiente e versátil, ampliando suas aplicações e contribuindo para avanços significativos na área. Além disso, a utilização dos fractais de Julia como ferramenta complementar permitirá uma exploração mais eficaz das regiões complexas do espaço de soluções, facilitando a identificação de soluções adicionais e proporcionando uma representação visual mais detalhada.

(ix) Além disso, em trabalhos futuros, pretende-se investigar métodos de gradientes mais eficientes para a resolução de problemas não lineares e multimodais. Nesse contexto, serão considerados os métodos propostos em referências como [28], que têm se mostrado promissores na abordagem de problemas com dimensões maiores. A utilização de tais métodos pode contribuir para aprimorar a convergência e a eficiência do algoritmo, possibilitando a resolução de problemas mais complexos e de maior escala.

Capítulo

8

Referências

[1] Marcos Andreani, Roberto; Raydan. Properties of the delayed weighted gradient method. *Computational Optimization and Applications 2020-sep 26*, sep 2020.

[2] A. T. Baraviera; F. M. Branco. *Sistemas Dinâmicos: uma primeira visão*. Campinas - SP: UNICAMP/IMECC, 2012.

[3] M. Barnsley. *Fractals Everywhere, 1a. edição*. Academic Press, Dublin, 1988.

[4] M. Barnsley; S. Damko. Iterated function system and the global construction of fractals. *Proc. R. Lond.*, A 399:243–275, 1985.

[5] Shetty, C.M Bazaraa, M. S, Sherali, H.D. *Nolinear Programming: Theory and Algorithms, third Edition*. Wiley Interscience- John Wiley & Sons Publishing Company, 2006.

[6] M. Brin; G. Stuck. *Introduction to Dynamical Systems*. Cambridge University Press, 2003.

[7] Richard L. Burden, J. Douglas Faires, and Annette M. Burden. *Numerical Analysis*. Cengage Learning, 2016.

[8] F Cooper. Introduction to fractals and julia sets. *University of Warwick*, 2011.

[9] R. L Devaney. *An Introduction to Chaotic Dynamical Systems, Second Edition*. Addison- Wesley Publishing Company, 1948.

[10] Robert L. Devaney. *An Introduction to Chaotic Dynamical Systems*. Westview Press, 2nd edition, 2003.

[11] Zdenek Dostál. *Optimal quadratic programming algorithms: with applications to variational inequalities*. Springer Optimization and Its Applications. Springer, 1 edition, 2009.

[12] A. P. Euletério. O espaço de hausdorff e a dimensão fractal: Estudo e abordagens no ensino fundamental. Master's thesis, Uberlândia- MG, 2021.

[13] K. Falconer. *Fractal Geometry*. John Wiley-Sons Ltd., 1990.

[14] M. Goemans. Pigeonhole principle and the probabilistic method. *MIT*, 2015. Lecture Notes.

[15] Vojin Jovanovic. Chaotic descent method and fractal conjecture. *International Journal for Numerical Methods in Engineering 2000-may 10 vol. 48 iss. 1*, 48, may 2000.

[16] C. T. Kelley. *Iterative Methods for Optimization.* SIAM, 1999.

[17] E. Kreyszig. *Introdutory Funcional Analysis With Applications,.* John Wiley & Sons, 1978.

[18] S. Lang. *Complex analysis.* Springer, 1999.

[19] G. C. Layek. *An Introduction to Dynamical Systems and Chaos.* Springer, 2015.

[20] E. L. Lima. *Curso de análise, vol.2, 1a. edição.* IMPA, 2014.

[21] B. Mandelbrot. *The Fractal Geometry of Nature.* W. H. Freeman, 1983.

[22] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[23] A. Rafael. Propriedades topológicas dos conjuntos de julia. Master's thesis, Universidade Estadual Paulista, "Julio de Mesquita Filho", 2008.

[24] H. M. de L. Ribeiro. Conjunto de julia. Master's thesis, Universidade Federal Rural de Pernambuco, 2014.

[25] W. Rudin. *Principles of Mathematical Analysis.* MacGraw-Hill Book Company, New York, 1953.

[26] W. Rudin. *Real and complex analysis.* McGraw-Hill Education, 1987.

[27] A. E. Schwertner. O método de levenberg-marquardt para problemas de otimização de menor valor ordenado. Master's thesis, Universidade Estadual de Maringá- PR, 2019.

[28] Diego Nunes Silva. Método primal-dual previsor-corretor de pontos interiores e exteriores com estratégias de correção de inércia e suavização hiperbólica aplicado ao problema de despacho econômico com ponto de carregamento de válvula e representação da transmissão. Master's thesis, Universidade Estadual Paulista, "Julio de Mesquita Filho", 2014.

[29] M. J. C Steinberg and T. H. SMITH. *Economic loading of power plants and eletric systems.* MacGraw-Hill, 1943.

[30] M Tavazoei, M. S. & Haeri. An optimization algorithm based on chaotic behavior and fractal nature. *Journal of Computational and Applied Mathematics 206 (2007) 1070 – 1081*, 2006.

[31] D. F. M Torres. *Programação Matemática.* UA, 2021.

Apêndice

A

Implementação dos Algoritmos e Figuras

Neste apêndice, serão apresentados alguns códigos usados para as figuras ou algoritmos desenvolvidos ao longo do trabalho. Foram usados códigos em MATLAB e Python.

A.1 Bacias de atração da equação $z^5 - 1 = 0$ (MATLAB)

```
1
2  % Fun  o para calcular a derivada da equa  o
3  f_prime = @(z) 5*z^4;
4
5  % Fun  o para calcular a equa  o
6  f = @(z) z^5 - 1;
7
8  % Definir o intervalo do plano complexo para plotagem
9  x = linspace(-1.5, 1.5, 1000);
10 y = linspace(-1.5, 1.5, 1000);
11 [X, Y] = meshgrid(x, y);
12 Z = X + 1i*Y;
13
14 % Definir as ra zes da equa  o
15 roots = [exp(2*pi*1i/5), exp(2*pi*2i/5), exp(2*pi*3i/5), exp
       (2*pi*4i/5), 1];
16
17 % Inicializar matriz para armazenar os  ndices  das ra zes
18 indices = zeros(size(Z));
19
20 % Loop para calcular os  ndices  das ra zes usando o m todo
       de Newton-Raphson
21 for i = 1:numel(Z)
22     z = Z(i);
23     iter = 0;
24     maxIter = 100;
25     tolerance = 1e-6;
26
```

```
    while abs(f(z)) > tolerance && iter < maxIter
        z = z - f(z) / f_prime(z);
        iter = iter + 1;
    end

    % Encontrar o  ndice  da raiz mais pr xima
    [~, index] = min(abs(roots - z));
    indices(i) = index;
end

% Plotar as bacias de atra  o
figure;
colormap(jet(5));  % Definir uma paleta de cores
imagesc(x, y, indices);
axis equal;
title('Bacias de Atra  o - z^5 - 1 = 0');
xlabel('Parte Real');
ylabel('Parte Imagin ria');
colorbar;

% Definir a fun  o e sua derivada, para determinar todos os
    zeros
f = @(z) z^5 - 1;
df = @(z) 5*z^4;

% Definir os par metros do m todo de Newton-Raphson
maxIter = 100; % N mero m ximo de itera  es
tol = 1e-6;    % Toler ncia de converg ncia

% Inicializar as ra zes, dependendo do n mero de ra zes
roots = zeros(1, 5);

% Aplicar o m todo de Newton-Raphson para encontrar as
   ra zes
for i = 1:5
    % Chute inicial pr ximo a cada raiz (pode ser ajustado)
    x0 = exp(2i*pi*(i-1)/5);

    % Itera  es do m todo de Newton-Raphson
    for iter = 1:maxIter
        x = x0 - f(x0) / df(x0);
        if abs(x - x0) < tol
            roots(i) = x;
            break;
        end
        x0 = x;
    end
end

% Imprimir as ra zes
```

```
disp('As ra zes  s  o:');
disp(roots);
```

Listing A.1: Bacias de atração da equação $z^5 - 1$

A.2 Grafico PDE- Três geradores e Curva de Nível (MATLAB)

```

%% Plota m nimos e maximos
X = a_min(1):2.5:b_max(1);
Y = a_min(2):2.5:b_max(2);
[x,y] = meshgrid(X,Y);
% com efeito do carregamento de v lvula
%Z = a(1).*X.^2 + b(1).*X + c(1) + abs(e(1).*sin(ft(1).*(
    V_min(1)-X))) + a(2).*Y.^2 + b(2).*Y + c(2) +abs(e(2).*sin
    (ft(2).*(V_min(2)-Y)))+ a(3).*X.^2 + b(3).*X + c(3) + abs(
    e(3).*sin(ft(3).*(V_min(3)-X)));
f1 = ((0.001562*(x.^2))+(7.92*x)+561+(abs((300*sin
    (0.0315*(100-x))))));
f2 = ((0.00482*(y.^2))+(7.97*y)+78+(abs(((150*sin(0.063*(50-y
    )))))));
f3 = ((0.00194*((Pd-x-y).^2))+(7.85*(Pd-x-y))+310+(abs(((200*
    sin(0.042*(100-(Pd-x-y)))))))));
f = f1+f2+f3;

mesh(x,y,f);
figure;
contour(x,y,f,300);
```

Listing A.2: Grafico PDE- Três geradores e Curva de Nível (MATLAB)

A.3 Implementação do AOC para o Problema 2 (MATLAB)

```
    %Implementa  o do m todo do Caos para o exemplo 2
%como descida m todo de gradientes ponderados

%tentativa com o m todo de busca unidimensional de Newton (n
    deu certo)
% function z=Newton(f,x0,tol)
% df = diff(sym(f));
% f1=subs(f,x0);
% df1 = subs(df,x0);
% c=1;
% while abs(f1) > tol
%     a= x0 - f1/df1;
```

```
%      f1 = subs(f,a);
%      df1= subs(df,a);
%      x0 = a;
%      c= c+1 ;
% end


close all
clear all
clc


%% Dados e Parametros do problema



a_min = [-10,-10];                % limites inferiores do
   intervalo
b_max = [10,10];                % limites superiorees

t1=1.1; t2 = 0.25; t3=1.1;
u = 1; % par metro inicial da fun   o penalidade de g(x)


%Gerando gr fico de f e curvas de n vel

syms x y% Define inc gnitas

f1=x*sin(9*pi*y);
f2=y*cos(25*pi*x);
f3=u*(x^2 + y^2 -81);
%f3=u*(sqrt(-y^2 +81);
f= -f1 - f2-20 + f3; %gr fico de f com a restri   o g(x)<=0


%% Plota minimos e maximos

X = a_min(1):2.0:b_max(1);
Y = a_min(2):2.0:b_max(2);
[x,y] = meshgrid(X,Y);

f1=x.*sin(9*pi.*y);
f2=y.*cos(25*pi.*x);
f3=(x.^2 + y.^2 -81);
f=-f1-f2-20+f3;

mesh(x,y,f);
figure;
contour(x,y,f,2500);

```

```
clear x y f
syms x y      %Define inc gnitas


%% Inicio do algoritimo
%
   ------------------------------------------------------------------------


%calcula fun   o gradiente da f e gradiente da g

f1=x.*sin(9*pi.*y);
f2=y.*cos(25*pi.*x);
g1= x.^2 + y.^2 -81;

f= -f1 - f2-20 ;

gradf = gradient(f);
gradg = gradient(g1);


hold on


for g=1:7   %executa o algoritmo completo por um n mero de
   vezes definida para g

%1a. onda - mapa do caos - 1o. m todo de otimiza   o do
   caos

%inicializa   o do m todo

l=0;
p_atual=10000;
Gama1=0.05;Gama2=0.05;
S1=8;
lambda=0.75;
u=1.0;
tol=10^-5;

%aplica   o do m todo

while l <= S1

 x = a_min(1) + Gama1*(b_max(1) - a_min(1));
 y = a_min(2) + Gama2*(b_max(2) - a_min(2));

 if S1==1
```

```
105  plot(x,y,'ob',"markersize",12); % Plota novo valor de x
        encontrado, indicando a trajetoria para encontar o minimo
         local (*)
106  end
107
108 f1=x.*sin(9*pi.*y);
109 f2=y.*cos(25*pi.*x);
110 f3=max(0,(x.^2 + y.^2 -81));
111 p0 = -f1 - f2 -20 + u*f3; %fun   o lagrangiana p(x) = f(x) +
        u*g(x)
112
113  x_inicial= [x;y];
114
115  if (p0 < p_atual)
116   p_atual=p0; x_atual=x_inicial;
117  end
118
119  Gama1= 4*Gama1*(1-Gama1);
120  Gama2= 4*Gama2*(1-Gama2);
121  l=l+1;
122 end
123
124 % Busca ao longo do m todo de gradiente ponderado
125
126 erro=1; cont=1;
127 x_min=x_atual;
128 %I=[1 0;0 1]; Beta = 0.2;
129
130
131 while ((erro>tol) &  (cont<50))
132
133
134     % Calcula os valores de x na fun   o gradiente
135      x = x_min(1);
136      y = x_min(2);
137
138
139     gradf0 = inline(gradf);
140     gradg0 = inline (gradg);
141
142     x_grad = gradf0 (x,y); % Resolve a equa   o
143     g_grad = gradg0 (x,y);
144
145     erro= max(abs(x_grad)); % Encontrar o maior valor
           absoluto do gradiente p(x_i)
146
147     G = (g_grad(1)^2) + (g_grad(2)^2);
148     %g1= max(0,x.^2 + y.^2 -81);
149     g1= x.^2 + y.^2 -81;
150     Delta1= 1-(g1/G);
```

```
151     Delta = 1/Delta1;
152     if g <= 0
153         w = 1;
154     else
155         w=Delta;
156     end
157     dx = x_grad + w*g_grad;
158 
159 %      for i=1:2
160 %      if x_min(i)>tol & dx(i)< 10^-6
161 %          h = min(-x_min(i)/dx(i),1)
162 %      end
163 %     end
164 
165     x_min= x_min - w*lambda*dx;
166   % end
167 %     if g<=2
168 %     x_min= x_min - h*x_grad;
169 %     end
170 
171    f1=x.*sin(9*pi.*y);
172     f2=y.*cos(25*pi.*x);
173     f3=max(0,(x.^2 + y.^2 -81));
174    p_min = -f1 - f2-20 + u* f3 ;
175 
176   if  (p_min < p_atual)
177       p_atual = p_min; x_atual = x_min; lambda = t2*lambda;
178   else
179       lambda = t1*lambda;
180   end
181 
182    cont= cont+1;
183    u=0.5*u;
184 end
185 plot(x_atual(1),x_atual(2),'*k',"markersize",8); % Plota novo
        valor de x encontrado, indicando a trajetoria para
       encontar o minimo local (*)
186 
187 %M todo de otimiza  o e caos - 2a. onda
188 
189 l=0;
190 Gama1=0.02; Gama2=0.01;
191 Alfa1 = 0.05; Alfa2=0.06;
192 x_min = x_atual;
193 
194 p0=p_atual;
195 S2=8; u=1;
196 
197 while l <= S2
198 
```

```
 x = x_min(1) + Alfa1*(Gama1 - 0.5);
 y = x_min(2) + Alfa2*(Gama2 - 0.5);
 plot(x,y,'.r',"markersize",10); % Plota novo valor de x
    encontrado, indicando a trajetoria para encontar o minimo
     local (*)

x_atual = [x,y];
f1=x.*sin(9*pi.*y);
f2=y.*cos(25*pi.*x);
f3=max(0,(x.^2 + y.^2 -81));
   p_atual = -f1 - f2-20 + u*f3;
u=0.5*u;
 if p_atual < p0
  p0= p_atual; x_min=x_atual;
  l = S2+1;       %stop o m todo de caos
 else
 Gama1= 4*Gama1*(1-Gama1);
 Gama2= 4*Gama2*(1-Gama2);
 Alfa1 = t3*Alfa1;
 Alfa2 = t3*Alfa2;
 l = l+1; %continua atualizando a solu   o pelo caos
 end
end
%plot(x,y,'.r',"markersize",10); % Plota novo valor de x
   encontrado, indicando a trajetoria para encontar o minimo
   local (*)
%Calcula o valor da fun   o objetivo na solu   o atual
x1=x_min(1);
x2=x_min(2);

xf(g,:) = [x1 x2];
i=1;
f_atual= -(xf(g,i)*sin(9*pi*xf(g,i+1))) - (xf(g,i+1)*cos(25*
   pi*xf(g,i)))-20 ;
F(g) = f_atual

[bestobj, ind] = min(F)
bestsol = xf(ind,:)
Soma = (sum(xf(ind,:).^2))
plot(bestsol(1),bestsol(2),'*b',"markersize",8);
% if (x_min(1)<b_max(1)) & (x_min(2)<b_max(2))& (x_min(1)>
   a_min(1))& (x_min(2)> a_min(2))
a_min(1)= x_min(1);
a_min(2)=x_min(2);

%end
end
plot(bestsol(1),bestsol(2),'or',"markersize",13);
```

Listing A.3: Implementação do AOC para o exemplo 2 (MATLAB)

A.4 Implementação do Problema 2 pelo Python

```
import numpy as np
from scipy import optimize
import matplotlib.pyplot as plt

# Fun  es necess rias
def M(gamma_l):
    a = 4
    return a * gamma_l * (1 - gamma_l)

def f(x, y):
    return -(x * np.sin(9 * np.pi * y) + y * np.cos(25 * np.pi * x)
        + 20)

def g1(x, y):
    return x**2 + y**2 - 81

def P(x, sigma=3):
    return f(x[0], x[1]) + sigma * max(0, g1(x[0], x[1]))

# Ajuste de sigma baseado na converg ncia
def adjust_sigma(gradient_norm, sigma):
    if gradient_norm < 1e-2:
        return sigma * 0.5
    elif gradient_norm < 1e-1:
        return sigma * 0.75
    else:
        return sigma

def gradient(x, sigma=10):
    df_dx = np.cos(9*np.pi*x[1]) - 25*np.pi*x[1]*np.sin(25*np.pi*x
       [0])
    df_dy = 9*np.pi*x[0]*np.cos(9*np.pi*x[1]) - np.sin(25*np.pi*x
       [0])
    dg_dx = 2 * x[0]
    dg_dy = 2 * x[1]
    return np.array([df_dx + sigma * max(0, dg_dx), df_dy + sigma *
        max(0, dg_dy)])

def hessian(x, sigma=10):
    ddf_dxdx = -625*(np.pi**2)*x[1]*np.cos(25*np.pi*x[0])
    ddf_dxdy = np.pi*(9*np.pi*x[0]*np.sin(9*np.pi*x[1]) + 25*np.cos
       (25*np.pi*x[0]))
    ddf_dydy = -81*(np.pi**2)*x[0]*np.sin(9*np.pi*x[1])
    ddg_dxdx = 2
    ddg_dydy = 2
    return np.array([[ddf_dxdx + sigma * max(0, ddg_dxdx), ddf_dxdy
       ], [ddf_dxdy, ddf_dydy + sigma * max(0, ddg_dydy)]])

def newton_method(x0, sigma=10, max_iter=1000, epsilon=1e-6):
    k = 0
    newton_trajectory = []
```

```
46      while np.linalg.norm(gradient(x0, sigma)) > epsilon and k <
           max_iter:
47          d = -np.linalg.inv(hessian(x0, sigma)).dot(gradient(x0,
               sigma))
48          alpha = optimize.minimize_scalar(lambda alpha: P(x0 + alpha
                * d, sigma)).x
49          x0 = x0 + alpha * d
50          newton_trajectory.append(x0)
51
52          # Atualiza  o de sigma
53          sigma = adjust_sigma(np.linalg.norm(gradient(x0, sigma)),
               sigma)
54
55          k += 1
56      return x0, np.linalg.norm(gradient(x0, sigma)) <= epsilon,
           newton_trajectory
57
58  # Inicializa  o das listas de trajet ria
59  chaotic_points1 = []
60  newton_points = []
61  chaotic_points2 = []
62
63  x = np.linspace(-10, 10, 400)
64  y = np.linspace(-10, 10, 400)
65  X, Y = np.meshgrid(x, y)
66  Z1 = f(X, Y)
67  plt.figure(figsize=(10,6))
68  plt.contour(X, Y, Z1, levels=14)
69  plt.title('Curvas de n vel da fun  o f(x, y)')
70  plt.xlabel('x')
71  plt.ylabel('y')
72  plt.colorbar()
73  plt.show()
74
75  Z2 = np.zeros_like(Z1)
76  for i in range(Z2.shape[0]):
77      for j in range(Z2.shape[1]):
78          Z2[i, j] = P([X[i, j], Y[i, j]])
79  plt.figure(figsize=(10,6))
80  plt.contour(X, Y, Z2, levels=14)
81  plt.title('Curvas de n vel da fun  o P(x, lambda)')
82  plt.xlabel('x')
83  plt.ylabel('y')
84  plt.colorbar()
85  plt.show()
86
87  S_1 = 1000000
88  S_2 = 20000
89  max_iter_chaos = 200000
90  t_3 = 0.05
91  gamma_0 = [0.64, 0.91]
92  x_star = gamma_0
93  alpha_i = [0.05 * (b - a) for a, b in [(-10, 10), (-10, 10)]]
```

```
94 P_star = float('inf')
95
96 for l in range(S_1):
97     x = [a + gamma_0[i]*(b - a) for i, (a, b) in enumerate([(-10,
           10), (-10, 10)])]
98     P_x = P(x)
99     chaotic_points1.append(x)  # Adiciona o ponto    lista
100    if P_x <= P_star:
101        x_star = x
102        P_star = P_x
103    gamma_0 = [M(gamma_0[i]) for i in range(len(gamma_0))]
104
105 continue_optimization = True
106 iteration = 0
107 while continue_optimization and iteration < max_iter_chaos:
108     x_star, converged, trajectory = newton_method(x_star)
109     newton_points.extend(trajectory)  # Adiciona pontos    lista
110     if not converged:
111         l_prime = 0
112         while alpha_i[0] < 0.1 * (10 - (-10)) and alpha_i[1] < 0.1
              * (10 - (-10)):
113             for i in range(len(gamma_0)):
114                 gamma_0[i] = M(gamma_0[i])
115                 x_star[i] = x_star[i] + alpha_i[i] * (gamma_0[i] -
                      0.5)
116             chaotic_points2.append(x_star)  # Adiciona o ponto
                  lista
117             P_x = P(x_star)
118             if P_x <= P_star:
119                 P_star = P_x
120             else:
121                 alpha_i = [t_3 * alpha_i[i] for i in range(len(
                      alpha_i))]
122             iteration += 1
123             if iteration >= max_iter_chaos:
124                 break
125         continue_optimization = False
126     else:
127         continue_optimization = False
128
129 z = x_star[0]**2 + x_star[1]**2
130 print(f'A solu   o  tima  ap s a aplica  o do m todo de Newton
      e da segunda onda de caos    {x_star}.')
131 print(f'O valor m nimo da fun  o objetivo nesta solu  o    {
    P_star}.')
132 print(f'O valor da restri  o no ponto  timo     {z}.')
133
134 # Plotando as trajet rias
135 plt.figure(figsize=(10,6))
136 plt.contour(X, Y, Z2, levels=14)
137 chaotic_points1_np = np.array(chaotic_points1)
138 chaotic_points2_np = np.array(chaotic_points2)
139 newton_points_np = np.array(newton_points)
```

```
plt.scatter(chaotic_points1_np[:,0], chaotic_points1_np[:,1], color='yellow', label='Primeira busca do CAOS')
plt.scatter(newton_points_np[:,0], newton_points_np[:,1], color='red', label='Metodo de Newton')
plt.scatter(chaotic_points2_np[:,0], chaotic_points2_np[:,1], color='green', label='Segunda busca de CAOS')
plt.title('Curvas de n vel da fun  o P(x, lambda) com pontos das buscas')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.colorbar()
plt.show()
```

Listing A.4: Implementação do Problema 2 (Python)

A.5 Determinar os zeros de uma equação ($x^3 - 3x$) (MATLAB)

```
% Fun  o a ser resolvida
f = @(x) x^3 - 3*x;

% Derivada da fun  o
df = @(x) 3*x^2 - 3;

% Fun  o a ser resolvida
df = @(x) 3*x^2 - 3;

% Derivada da fun  o
ddf = @(x) 6*x;

% Ponto cr tico inicial
x_d = 0.5;

% Toler ncia
epsilon = 1e-3;

% Vetor para armazenar as solu  es encontradas
solutions = [];

% Fun  o de itera  o de Newton-Raphson
Nf = @(x) x - (df(x)/ddf(x));

% Itera  o para encontrar as solu  es
x = x_d;
while true
    x = Nf(x);
    if abs(df(x)) < epsilon
        % Verificando se a solu  o    nova
        isNew = true;
```

```
32          for k = 1:length(solutions)
33              if abs(x - solutions(k)) < epsilon
34                  isNew = false;
35                  break;
36              end
37          end
38          % Adicionando a nova solu  o ao vetor
39          if isNew
40              solutions = [solutions x];
41          end
42          break;
43      end
44  end
45
46  % Imprimindo as solu  es encontradas
47  if isempty(solutions)
48      disp('N o foram encontradas solu  es.');
49  else
50      disp(['Pontos pertencentes ao conjunto de Julia: '
          num2str(solutions) '.']);
51  end
52
53
54  % Ponto cr tico
55  x_c = solutions; % Ponto cr tico em x = solutions
56
57  % Toler ncia
58  Tol = 1e-1;
59
60  % Vetor para armazenar as solu  es encontradas
61  solutions = [];
62
63  % Fun  o de itera  o de Newton-Raphson
64  Nf = @(x) x - (f(x)/df(x));
65
66  % Itera  o para encontrar as solu  es
67  for x0 = x_c - 1:epsilon:x_c + Tol
68      x = x0;
69      for n = 1:500
70          x = Nf(x);
71          if abs(f(x)) < epsilon
72              % Verificando se a solu  o    nova
73              isNew = true;
74              for k = 1:length(solutions)
75                  if abs(x - solutions(k)) < epsilon
76                      isNew = false;
77                      break;
78                  end
79              end
80              % Adicionando a nova solu  o ao vetor
```

```
81              if isNew
82                  solutions = [solutions x];
83              end
84              break;
85          end
86      end
87  end
88
89  % Imprimindo as solu  es encontradas
90  if isempty(solutions)
91      disp('N o foram encontradas solu  es.');
92  else
93      disp(['Solu  es encontradas: ' num2str(solutions) '.'])
           ;
94  end
95
96
97
98  index = 1;
99  menor = inf;
100
101 while index <= length(solutions)
102     element = solutions(index);
103     %disp(element);  % Fa a algo com o elemento
104     output = funcao_principal(element);
105     %disp(output)
106     if output < menor
107         menor = output;
108     end
109
110
111
112     index = index + 1;
113 end
114 disp(menor)
115
116
117
118 %% funcao
119 function [output] = funcao_principal(x0)
120 output = x0^3 - 3*x0;
121
122
123 end
```

Listing A.5: Determinar os zeros de uma equação ($x^3 - 3x$) (MATLAB)

A.6 Iplementação do Problema 1- PDE (MATLAB)

```
%Implementa  o do AOC para resolu  o do PDE


close all
clear all
clc

%% Parametros do problema

% 3 geradores
G = 3;                                  % Numero de geradores
a_min = [100,50,100];                   % Minimo
b_max = [600,200,400];                  % Maximo
Pd = 850;                               % Demanda
a = [0.001562,0.00482,0.00194];
b = [7.92,7.97,7.85];
c = [561,78,310];
e = [300,150,200];
ft = [0.0315,0.063,0.042];
t1=2; t2 = 0.75; t3=2;
Lambda = 0.25;
n = 0.5;%  atualiza  o do par metro de suaviza  o
u = 10; % par metro de penalidade de g(x) para a fun  o p(x,u)
k1 = 5;
syms x y% Define inc gnitas

% f = f + ((a1(i)*(x(i)^2.0))+(b1(i)* x(i))+c1(i)+ abs(e1(i)*(sin(f1(i)*(V_min(i)-x(i))))))
f1 = ((0.001562*(x^2))+(7.92*x)+561+(sqrt(((300*(sin(0.0315*(100-x))))^2)+n)));
f2 = ((0.00482*(y^2))+(7.97*y)+78+(sqrt(((150*(sin(0.063*(50-y))))^2)+n)));
f3 = ((0.00194*((Pd-x-y)^2))+(7.85*(Pd-x-y))+310+(sqrt(((200*(sin(0.042*(100-(Pd-x-y)))))^2)+n)));

% f4 = u*((exp(k1*(100-x)))+exp((k1*(x-600))));
% f5 = u*((exp(k1*(50-y)))+exp((k1*(y-200))));
% f6 = u*((exp(k1*(100-Pd+x+y)))+exp((k1*(Pd-x-y-400))));

%f = f1+f2+f3+f4+f5+f6;
f = f1+f2+f3;

%% Plota minimos e maximos
X = a_min(1):2.5:b_max(1);
Y = a_min(2):2.5:b_max(2);
[x,y] = meshgrid(X,Y);
% com efeito do carregamento de v lvula
%Z = a(1).*X.^2 + b(1).*X + c(1) + abs(e(1).*sin(ft(1).*(V_min(1)-X))) + a(2).*Y.^2 + b(2).*Y + c(2) +abs(e(2).*sin
```

```
   (ft(2).*(V_min(2)-Y)))+ a(3).*X.^2 + b(3).*X + c(3) + abs(
   e(3).*sin(ft(3).*(V_min(3)-X)));
45 f1 = ((0.001562.*(x.^2))+(7.92.*x)+561+(sqrt(((300.*(sin
   (0.0315.*(100-x)))).^2)+n)));
46 f2 = ((0.00482.*(y.^2))+(7.97.*y)+78+(sqrt(((150.*(sin
   (0.063.*(50-y)))).^2)+n)));
47 f3 = ((0.00194.*((Pd-x-y).^2))+(7.85.*(Pd-x-y))+310+(sqrt
   (((200.*(sin(0.042.*(100-(Pd-x-y))))).^2)+n)));
48 f = f1+f2+f3;
49
50 mesh(x,y,f);
51 figure;
52 contour(x,y,f,300);
53 %quiver(X,Y,Z);
54 clear x y f
55 % [x y]=meshgrid(-2:4/200:2,-1:2/200:1);
56 % grid;
57 % contour(x,y,f,100);
58 % hold;
59 % zoom
60 % clear x y
61
62 %% Inicio do algoritimo
63
64 %
   ----------------------------------------------------------------------

65 % Etapa 1
66 % Inicialmente, buscar um ponto de m nimo inicial utilizando
     o m todo de
67 % caos 1a. onda
68
69 % Em seguida Encontrar um minimo atual usando m todo de
   Newton ou de
70 % Newton- Levenberg
71
72 %
   ----------------------------------------------------------------------

73
74 %Dados iniciais
75
76 for g=1:4 %n mero de execu  es que o m todo realiza
77
78 syms x y%Define inc gnitas
79 % Gradiente da fun   o f
80 f1 = ((0.001562.*(x.^2))+(7.92.*x)+561+(sqrt(((300.*(sin
   (0.0315.*(100-x)))).^2)+n)));
81 f2 = ((0.00482.*(y.^2))+(7.97.*y)+78+(sqrt(((150.*(sin
   (0.063.*(50-y)))).^2)+n)));
```

```
f3 = ((0.00194.*((Pd-x-y).^2))+(7.85.*(Pd-x-y))+310+(sqrt
   (((200.*(sin(0.042.*(100-(Pd-x-y))))).^2)+n)));
f = f1+f2+f3;

gradf = gradient(f);

h= 0.75; % Escolhendo um valor para h para limitar o tamanho
   de cada passo do m todo de Newton (N o usa passo 1)

hold on

%par metros de controle - m todo do caos - 1a. onda

l=0; f10=10000;
Gama1=0.25; Gama2=0.25;  S1=6;

while l <= S1

 x = a_min(1) + Gama1*(b_max(1) - a_min(1));
 y = a_min(2) + Gama2*(b_max(2) - a_min(2));

 if S1==1
 plot(x,y,'*k'); % Plota novo valor de x encontrado,
    indicando a trajetoria para encontar o minimo local (*)
 end

f1 = ((0.001562.*(x.^2))+(7.92.*x)+561+(sqrt(((300.*(sin
   (0.0315.*(100-x)))).^2)+n)));
f2 = ((0.00482.*(y.^2))+(7.97.*y)+78+(sqrt(((150.*(sin
   (0.063.*(50-y)))).^2)+n)));
f3 = ((0.00194.*((Pd-x-y).^2))+(7.85.*(Pd-x-y))+310+(sqrt
   (((200.*(sin(0.042.*(100-(Pd-x-y))))).^2)+n)));
f_atual = f1+f2+f3;
%f_atual= f2;
n=n/2;
 x_inicial= [x;y];
% f_atual = f (x_inicial(1),x_inicial(2))

 if (f_atual < f10)
  f10= f_atual; x_atual=x_inicial;
 end

 Gama1= 4*Gama1*(1-Gama1);
 Gama2= 4*Gama2*(1-Gama2);
 l=l+1;
end

% Busca ao longo do m todo de Newton

erro=1; cont=1; x_min=x_inicial;
```

```
I=[1 0;0 1]; Beta = 0.01;%h=0.5;

while ((erro>10^-5) & (abs(x_min)<600) & (cont<100))


    % Calcula os valores de x na fun   o gradiente
    x = x_min(1);
    y = x_min(2);
    gradf0 = inline(gradf);
    x_grad = gradf0 (x,y); % Resolve a equa   o

    erro= max(abs(x_grad)); % Encontrar o maior valor
       absoluto do gradiente f(x_i)

    H = inline (hessian(f)); % Hessiana
    H_x= H (x,y);
    autovalores = eig(H_x); % Encontra autovalores e
       autovetores para novo encontrado
     n_autovalores = sum(autovalores>0);
  % if g >=3
   while n_autovalores~=2
      H_x= H_x + Beta*I;
      autovalores = eig(H_x);
      n_autovalores = sum(autovalores>0);
     Beta = 2*Beta;
   end

  H_x = inv(H (x,y)); %se quiser utilizar Newton puro
    %H_x = inv(H_x); %se quiser usar Newton-Levenberg

    x_min= x_min - h*H_x*x_grad;

    cont= cont+1;
    h=1.1*h;
end
plot(x_min(1),x_min(2),'*b',"markersize",13); % Plota novo
   valor de x encontrado, indicando a trajetoria para
   encontar o minimo local (*)

%par metros de controle - m todo do caos - 2a. onda
l=0;
Gama1=0.15; Gama2=0.15;
Alfa1 = 0.25; Alfa2=0.25;
f10=f_atual;
S2=6;
while l <= S2

 x = x_min(1) + Alfa1*(Gama1 - 6);
 y = x_min(2) + Alfa2*(Gama2 - 0.0125);
```

```
 plot(x,y,'.r'); % Plota novo valor de x encontrado,
    indicando a trajetoria para encontar o minimo local (*)

x_atual = [x,y];
 f1 = ((0.001562.*(x.^2))+(7.92.*x)+561+(sqrt(((300.*(sin
    (0.0315.*(100-x)))).^2)+n)));
f2 = ((0.00482.*(y.^2))+(7.97.*y)+78+(sqrt(((150.*(sin
   (0.063.*(50-y)))).^2)+n)));
f3 = ((0.00194.*((Pd-x-y).^2))+(7.85.*(Pd-x-y))+310+(sqrt
   (((200.*(sin(0.042.*(100-(Pd-x-y))))).^2)+n)));
f_atual = f1+f2+f3;
n=n/2;
%f_atual= f2;

 if f_atual <= f10
  f10= f_atual; x_min=x_atual;
  plot(x_min(1),x_min(2),'.r',"markersize",13); l = S2+1;
 else
 Gama1= 8*Gama1*(1-Gama1);
 Gama2= 8*Gama2*(1-Gama2);
 Alfa1 = t3*Alfa1;
 Alfa2 = t3*Alfa2;
 l = l+1;
 end
end

%Calcula do custo da fun   o do PDE
x1=x_min(1);
x2=x_min(2);
x3 = Pd - x1 - x2;


xf(g,:) = [x1 x2 x3];
a_min = [100,50,100];
C = 0;

for i=1:G       % Calcula a somatoria das fun   es custo dos
    geradores
C = C + ((a(i)*(xf(g,i)^2.0))+(b(i)* xf(g,i))+c(i)+ abs(e(i)
   *(sin(ft(i)*(a_min(i)-xf(g,i))))));
end
F(g) = C

[bestobj, ind] = min(F)
bestsol = xf(ind,:)
Soma = sum(xf(ind,:))

a_min(1)= x_min(1);
a_min(2)= x_min(2);
n=0.5;
```

```
h=0.75;
end
plot(bestsol(1),bestsol(2),'ob',"markersize",13);
```

Listing A.6: Iplementação do PDE (MATLAB)

A.7 Bacia de Atração (Python)

```python
import numpy as np
import matplotlib.pyplot as plt

def bacia_atracao(n, limite, max_iter, tamanho_janela):
    # Gerar a grade de pontos no plano complexo
    x = np.linspace(-limite, limite, n)
    y = np.linspace(-limite, limite, n)
    xx, yy = np.meshgrid(x, y)
    z0 = xx + 1j * yy

    # Inicializar a matriz de cores
    cores = np.zeros((n, n))

    # Gerar as  rbitas  dos pontos na grade
    for i in range(max_iter):
        z0 = z0**2
        mascara = np.abs(z0) > 2
        cores[mascara & (cores == 0)] = i+1
        z0[mascara] = 2

    # Plotar a bacia de atra  o
    plt.figure(figsize=(tamanho_janela, tamanho_janela))
    plt.imshow(cores.T, origin='lower', cmap='Blues', extent=(-
       limite, limite, -limite, limite))
    plt.xlabel('Parte real')
    plt.ylabel('Parte imagin ria')
    plt.show()
```

Listing A.7: Bacia de Atração (Python)

A.8 Função de Suavização Hiperbólica (Python)

```python
import numpy as np
import matplotlib.pyplot as plt

# Vetor de valores x
x = np.arange(-10, 10, 0.1)

# Fun  o de suaviza  o hiperb lica com diferentes valores de
    eta
phi1 = np.sqrt(x**2)
phi2 = np.sqrt(x**2 + 4)
phi3 = np.sqrt(x**2 + 1)
```

```
phi4 = np.sqrt(x**2 + 0.25)
phi5 = np.sqrt(x**2 + 0.04)
phi6 = np.sqrt(x**2 + 0.0001)

# Plot das fun  es
plt.plot(x, phi1, linewidth=2)
plt.plot(x, phi2, linewidth=2)
plt.plot(x, phi3, linewidth=2)
plt.plot(x, phi4, linewidth=2)
plt.plot(x, phi5, linewidth=2)
plt.plot(x, phi6, linewidth=2)

# Legenda e t tulo do gr fico
plt.legend(['$\phi(x) = \sqrt{x^2}$', '$\phi(x) = \sqrt{x^2 + 4}$',
    '$\phi(x) = \sqrt{x^2 + 1}$', '$\phi(x) = \sqrt{x^2 + 0.25}$',
   '$\phi(x) = \sqrt{x^2 + 0.04}$', '$\phi(x) = \sqrt{x^2 + 0.0001}
   $'], loc='upper left')
plt.title('Fun  es de Suaviza  o Hiperb lica')
plt.xlabel('x')
plt.ylabel('$\phi(x)$')
plt.grid(True)
plt.show()
```

Listing A.8: Função de Suavização Hiperbólica (Python)

A.9 Gráfico do Problema 2 (Python)

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Defina a fun  o
def tavazoei_funcao(x, y):
    return -(x * np.sin(9 * np.pi * y) + y * np.cos(25 * np.pi * x)
        + 20)

# Crie os valores de x e y para a grade
x = np.linspace(-1, 1, 100)
y = np.linspace(-1, 1, 100)

# Crie uma grade de pontos
X, Y = np.meshgrid(x, y)

# Calcule os valores correspondentes de Z
Z = tavazoei_funcao(X, Y)

# Crie o gr fico 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(X, Y, Z, cmap='viridis')

# Adicione a barra de cores
```

```
fig.colorbar(surf)

# Defina os r tulos dos eixos
ax.set_xlabel('Eixo X')
ax.set_ylabel('Eixo Y')
ax.set_zlabel('Eixo Z')
plt.title('Gr fico da Fun   o')

# Exiba o gr fico
plt.show()
```

Listing A.9: Gráfico da Função Objetivo do Problema 2

A.10 Gráfico da Função Objetivo - Superfície Planificada do Problema 2 (Python)

```
import numpy as np
import matplotlib.pyplot as plt

# Defina a fun   o
def tavazoei_funcao(x, y):
    return -(x * np.sin(9 * np.pi * y) + y * np.cos(25 * np.pi * x)
        + 20)

# Crie os valores de x e y para a grade
x = np.linspace(-1, 1, 100)
y = np.linspace(-1, 1, 100)

# Crie uma grade de pontos
X, Y = np.meshgrid(x, y)

# Calcule os valores correspondentes de Z
Z = tavazoei_funcao(X, Y)

# Crie o gr fico de contorno
fig, ax = plt.subplots()
contour = ax.contour(X, Y, Z, cmap='viridis')

# Defina os r tulos dos eixos
ax.set_xlabel('Eixo X')
ax.set_ylabel('Eixo Y')
plt.title('Gr fico da Fun   o - Superf cie Planificada')

# Adicione uma barra de cores
cbar = plt.colorbar(contour)

# Exiba o gr fico
plt.show()
```

Listing A.10: Gráfico da Função Objetivo - Superfície Planificada do Problema 2

A.11 Curvas de Nível da Função Objetivo do Problema 2 (Python)

```

import numpy as np
import matplotlib.pyplot as plt

# Defina a fun   o
def tavazoei_curvas_de_nivel_funcao(x, y):
    return -(x * np.sin(9 * np.pi * y) + y * np.cos(25 * np.pi * x)
        + 20)

# Crie os valores de x e y para a grade
x = np.linspace(-1, 1, 100)
y = np.linspace(-1, 1, 100)

# Crie uma grade de pontos
X, Y = np.meshgrid(x, y)

# Calcule os valores correspondentes de Z
Z = tavazoei_curvas_de_nivel_funcao(X, Y)

# Crie o gr fico de contorno
plt.contour(X, Y, Z, levels=20, cmap='viridis')
plt.colorbar()

# Defina os r tulos dos eixos
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Curvas de N vel da Fun   o')

# Exiba o gr fico
plt.show()
```

Listing A.11: Curvas de Nível da Função Objetivo do Problema 2

A.12 Gráfico da Função Penalizada do Problema 2 (Python)

```

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Defina a fun   o
def tavazoei_funcao_penalizada(x, y):
    return -(x * np.sin(9 * np.pi * y) + y * np.cos(25 * np.pi * x)
        + 20) + 10 * (x**2 + y**2 - 81)

# Crie os valores de x e y para a grade
x = np.linspace(-10, 10, 100)
y = np.linspace(-10, 10, 100)
```

```
13
14 # Crie uma grade de pontos
15 X, Y = np.meshgrid(x, y)
16
17 # Calcule os valores correspondentes de Z
18 Z = tavazoei_funcao_penalizada(X, Y)
19
20 # Crie o gr fico 3D
21 fig = plt.figure()
22 ax = fig.add_subplot(111, projection='3d')
23 ax.plot_surface(X, Y, Z, cmap='viridis')
24
25 # Crie uma inst ncia do objeto ScalarMappable para mapear os
     valores da cor
26 mappable = plt.cm.ScalarMappable(cmap='viridis')
27 mappable.set_array(Z)
28
29 # Adicione a barra de cores ao gr fico
30 colorbar = plt.colorbar(mappable)
31 colorbar.set_label('Valores de f(x,y)')
32
33 # Defina os r tulos dos eixos
34 ax.set_xlabel('Eixo X')
35 ax.set_ylabel('Eixo Y')
36 ax.set_zlabel('Eixo Z')
37
38 # Exiba o gr fico
39 plt.show()
```

Listing A.12: Gráfico da Função Penalizada do Problema 2

A.13 Curvas de Nível da Função Penalizada do Problema 2 (Python)

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Defina a fun   o
5 def tavazoei_funcao_penalizada_curvas_de_nivel(x, y):
6     return -(x * np.sin(9 * np.pi * y) + y * np.cos(25 * np.pi * x)
          + 20) + 10 * (x**2 + y**2 - 81)
7
8 # Crie os valores de x e y para a grade
9 x = np.linspace(-20, 20, 100)
10 y = np.linspace(-20, 20, 100)
11
12 # Crie uma grade de pontos
13 X, Y = np.meshgrid(x, y)
14
15 # Calcule os valores correspondentes de Z
16 Z = tavazoei_funcao_penalizada_curvas_de_nivel(X, Y)
17
```

```
# Crie o gr fico de contorno
plt.contour(X, Y, Z, levels=20, cmap='viridis')
plt.colorbar()

# Defina os r tulos dos eixos
plt.xlabel('Eixo X')
plt.ylabel('Eixo Y')
plt.title('Curvas de N vel da Fun   o')

# Exiba o gr fico
plt.show()
```

Listing A.13: Curvas de Nível da Função Penalizada do Problema 2

A.14 Gráfico da Função Objetivo do Problema 1 (PDE Suavizado, $\eta = 10$)- Python

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Definir a fun   o
def funcao(x, y):
    return (
        0.001562 * x**2 + 7.92 * x + 561 + np.sqrt((300 * np.sin
            (0.0315 * (100 - x)))**2+10)
        + 0.00482 * y**2 + 7.97 * y + 78 + np.sqrt((150 * np.sin
            (0.063 * (50 - y)))**2+10)
        + 0.00194 * (850 - x - y)**2 + 7.85 * (850 - x - y) + 310
        + np.sqrt((200 * np.sin(0.042 * (100 - (850 - x - y))))
            **2+10)
    )

# Criar os valores de x e y para a grade
x = np.linspace(0, 500, 100)
y = np.linspace(0, 300, 100)

# Criar uma grade de pontos
X, Y = np.meshgrid(x, y)

# Calcular os valores correspondentes de Z
Z = funcao(X, Y)

# Criar o gr fico 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(X, Y, Z, cmap='viridis')

# Configurar a rota   o interativa
ax.view_init(elev=30, azim=45)  # Definir  ngulos de eleva   o e
    azimute
ax.dist = 10  # Definir a dist ncia da c mera
```

```
fig.colorbar(surf)  # Adicionar barra de cores

# Definir os r tulos dos eixos
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('f(P_G)')

# Exibir o gr fico
plt.show()
```

Listing A.14: Gráfico da Função Objetivo do Problema 1 (PDE Suavizado, $\eta = 10$)- Python

A.15 Curvas de Nivel do Problema 1 (PDE)- Python

```
import numpy as np
import matplotlib.pyplot as plt

# Defina a fun   o
def funcao(x, y):
    return (
        0.001562 * x**2 + 7.92 * x + 561 + np.sqrt((300 * np.sin
            (0.0315 * (100 - x)))**2+10)
        + 0.00482 * y**2 + 7.97 * y + 78 + np.sqrt((150 * np.sin
            (0.063 * (50 - y)))**2+10)
        + 0.00194 * (850 - x - y)**2 + 7.85 * (850 - x - y) + 310
        + np.sqrt((200 * np.sin(0.042 * (100 - (850 - x - y))))
            **2+10)
    )

# Crie os valores de x e y para a grade
x = np.linspace(0, 600, 100)
y = np.linspace(0, 200, 100)

# Crie uma grade de pontos
X, Y = np.meshgrid(x, y)

# Calcule os valores correspondentes de Z
Z = +funcao(X, Y)

# Aumente o n mero de curvas de n vel para o dobro
num_curvas = 40

# Crie o gr fico de contorno com mais curvas
plt.contour(X, Y, Z, levels=num_curvas, cmap='viridis')
plt.colorbar()

# Defina os r tulos dos eixos
plt.xlabel('x')
plt.ylabel('y')
plt.title('Curvas de N vel da Fun   o')

```

```
35 # Exiba o gr fico
36 plt.show()
```

Listing A.15: Curvas de Nivel do Problema 1 (PDE)

A.16 Função Quase Convexa

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 x = np.linspace(-10, 10, 100)
5 y = x**2
6
7 plt.plot(x, y)
8 plt.xlabel('x')
9 plt.ylabel('f(x)')
10 plt.title('Fun   o quase convexa: f(x) = x^2')
11 plt.show()
```

Listing A.16: Função Quase Convexa (Python)

A.17 Gráfico da Função- Superfície Planificada (Python)

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Definir a fun   o
5 def funcao(x, y):
6     return (
7         0.001562 * x**2 + 7.92 * x + 561 + np.sqrt((300 * np.sin
              (0.0315 * (100 - x)))**2+10)
8         + 0.00482 * y**2 + 7.97 * y + 78 + np.sqrt((150 * np.sin
              (0.063 * (50 - y)))**2+10)
9         + 0.00194 * (850 - x - y)**2 + 7.85 * (850 - x - y) + 310
10         + np.sqrt((200 * np.sin(0.042 * (100 - (850 - x - y))))
              **2+10)
11     )
12
13 # Criar os valores de x e y para a grade
14 x = np.linspace(0, 600, 100)
15 y = np.linspace(0, 300, 100)
16
17 # Criar uma grade de pontos
18 X, Y = np.meshgrid(x, y)
19
20 # Calcular os valores correspondentes de Z
21 Z = funcao(X, Y)
22
23 # Crie o gr fico de contorno
24 fig, ax = plt.subplots()
25 contour = ax.contour(X, Y, Z, cmap='viridis')
```

```
26
27 # Defina os r tulos dos eixos
28 ax.set_xlabel('x')
29 ax.set_ylabel('y')
30 plt.title('Gr fico da Fun   o - Superf cie Planificada')
31
32 # Adicione uma barra de cores
33 cbar = plt.colorbar(contour)
34
35 # Exiba o gr fico
36 plt.show()
```

Listing A.17: Gráfico da Função- Superfície Planificada (Python)