Universidade Federal Fluminense

ALEX TAVARES SILVA

Análise dos Métodos Luus-Jaakola e Algoritmo de Colisão de Partículas na Estimativa de Parâmetros de um Problema de Transferência de Calor

VOLTA REDONDA

2019

ALEX TAVARES SILVA

Análise dos Métodos Luus-Jaakola e Algoritmo de Colisão de Partículas na Estimativa de Parâmetros de um Problema de Transferência de Calor

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional.

Orientador:

Professor D.Sc. Wagner Rambaldi Telles

Coorientador:

Professor D.Sc. Gustavo Silva Semaan

Universidade Federal Fluminense

VOLTA REDONDA

2019

Ficha catalográfica automática - SDC/BEM
Gerada com informações fornecidas pelo autor

S586a Silva, Alex Tavares
Análise dos Métodos Luus-Jaakola e Algoritmo de Colisão de Partículas na Estimativa de Parâmetros de um Problema de Transferência de Calor / Alex Tavares Silva ; Wagner Rambaldi Telles, orientador ; Gustavo Silva Semaan, coorientador. Volta Redonda, 2019.
123 p. : il.

Dissertação (mestrado)-Universidade Federal Fluminense, Volta Redonda, 2019.

DOI: http://dx.doi.org/10.22409/PPG-MCCT.2019.m.08928885710

1. Transferência de Calor. 2. Método das Diferenças Finitas. 3. Luus-Jaakola. 4. Algoritmo de Colisão de Partículas. 5. Produção intelectual. I. Telles, Wagner Rambaldi, orientador. II. Semaan, Gustavo Silva, coorientador. III. Universidade Federal Fluminense. Escola de Engenharia Industrial e Metalúrgica de Volta Redonda. IV. Título.

CDD -

Bibliotecária responsável: Andreia de Oliveira Paim - CRB7/5183

Análise dos Métodos Luus-Jaakola e Algoritmo de Colisão de Partículas na Estimativa de Parâmetros de um Problema de Transferência de Calor

Alex Tavares Silva

Dissertação apresentada ao Programa de Pós-graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Federal Fluminense, como requisito parcial para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia. Área de Concentração: Modelagem Computacional.

Aprovada por:

Prof. Wagner Rambaldi Telles, D.Sc. / INFES-UFF(Presidente)

Prof. Thiago Jordem Pereira, D.Sc. / INFES-UFF

Prof. Gustavo Benitez Alvarez, D.Sc. / MCCT-UFF

Prof. Nelson Machado Barbosa, D. Sc. / LCMAT-UENF

Volta Redonda, 27 de Março de 2019.

Dedicatória.

Ethel Riscado e Arthur Riscado

Agradecimentos

Gostaria de agradecer primeiramente a Deus, por me dar forças para superar as dificuldades da vida.

Agradecer meus pais e irmãos por acreditarem em mim.

Minha esposa e filho por me apoiar e incentivar.

Meu amigo Marcos Felipe Medeiros de Souza pelas conversas enriquecedoras sobre o curso e pelas dicas de paternidade.

Meus colegas de mestrado, por me propiciarem momentos inesquecíveis.

Agradecer aos meus orientadores Wagner Telles e Gustavo Semaan pelo apoio psicológico e intelectual durante o programa.

Aos demais professores que me ajudaram com seus conhecimentos científicos.

Aos organizadores desse programa de Pós-Graduação, por me aceitarem, me permitirem ampliar meus conhecimentos e por acreditarem em mim.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Métodos de otimização são importantes quando deseja-se buscar boas soluções, em grandes espaços de busca, respeitando as restrições do problema abordado. Neste trabalho um problema de transferência de calor é resolvido numericamente utilizando as abordagens direta e inversa. Para a abordagem direta, é adotado o Método das Diferenças Finitas empregando as formulações explícita e implícita. A formulação explícita é condicionalmente estável, logo a convergência fica limitada pelo critério de estabilidade de Neumann. Na formulação implícita, o sistema de equações obtido é resolvido utilizando o método Gauss-Seidel. Apesar do custo computacional ser maior na formulação implícita, os resultados convergem com mais frequência por ser incondicionalmente estável. Já a abordagem inversa busca estimar um conjunto de parâmetros considerados ótimos locais, baseado na comparação entre os resultados numéricos obtidos pelo modelo matemático e os dados experimentais da literatura. Ainda no que se refere à abordagem inversa, são propostas modificações nos métodos Luus-Jaakola e Algoritmo de Colisão de Partículas, assim como hibridizações entre os mesmos. No total são foram propostas 19 versões, entre modificações e hibridizações, e comparados os resultados obtidos com os respectivos métodos em suas versões clássicas. Conclui-se, com base nos experimentos realizados, que a abordagem empregada apresenta-se como um caminho alternativo para a resolução do problema proposto.

Abstract

Optimization methods are important when it is wanted to search for good solutions, in huge search space, according to the problem already mentioned and its restrictions. In this work a problem of heat transfer is numerically solved using the direct and inverse approaches. To the direct approach, it is adopted the Finite Differences Method using the implicit and explicit formulations. The explicit formulation is conditionally stable; this way the convergence gets limited by the Neumann's stability criterion. In the implicit formulation, the equations system obtained is solved using the Gauss-Seidel method. Although the computing cost is bigger in the implicit formulation, the results converge with more frequency due to fact of being unconditionally stable. The inverse approach aims to estimate a set of parameters considered local optima, based on the comparison between the numeric results obtained by the mathematic method and the experimental data obtained from literature. Also with regard to the inverse approach, are proposed modifications in the Luus-Jaakola and Particle Collision methods, as well as hybridizations between them. In total, were proposed 19 versions, between modifications and hybridizations, and compared the obtained results with the respective methods and their classic versions. It is concluded from the experiments carried out, that the approach employed presents itself an alternative way to solve the problem proposed.

Palavras-chave

1. Problemas Direto e Inverso
2. Equação do Calor
3. Método das Diferenças Finitas
4. Luus-Jaakola
5. Algoritmo de Colisão de Partículas

Glossário

EDP	:	Equação Diferencial Parcial
MDF	:	Método das Diferenças Finitas
LJ	:	Luus-Jaakola
PCA	:	Algoritmo de Colisão de Partículas - *Particle Colision Algorithm*
SA	:	Algoritmo de Recozimento Simulado - *Simulated Annealing*
NAF	:	Número de Avaliações da Função de Objetivo

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo é apresentada uma visão geral para melhor compreensão do trabalho realizado, descrevendo a estrutura da dissertação, as etapas desenvolvidas em cada capítulo e a análise dos resultados obtidos.

1.1 Considerações Gerais

Desde o princípio da humanidade, o homem encontra desafios no seu dia a dia que necessitam de solução para o seu próprio bem. Com o passar do tempo, a Matemática se tornou mais uma aliada, fornecendo mecanismos de apoio para a resolução de problemas. Nesse sentido, um modelo matemático é uma forma de se representar, matematicamente, a natureza com o intuito de resolver um dado problema.

Um problema real pode ser difícil ou impraticável de se modelar devido a grande quantidade de variáveis envolvidas. Dessa forma, uma prática comumente utilizada é propor um problema simplificado com uma quantidade menor de variáveis e resolvê-lo. Depois que se obtém a solução do problema simplificado, deve-se analisar se tal solução serve como solução aproximada do problema real. Assim, a matemática e a realidade são conjuntos disjuntos que podem ser levadas à interação através da Modelagem [28].

No que se refere à modelagem de problemas, o tema equações diferenciais é de grande importância, pois o conceito de derivada, assim como os problemas reais, envolve a ideia de taxa de variação de uma determinada grandeza. Apesar da dificuldade de propor um modelo matemático, muitos desses já existem nas mais diversas áreas, tais como, modelos que representam crescimento populacional, sendo possível estudar como se desenvolvem certos grupos de seres (pessoas, mosquitos transmissores de doenças, entre outros). Tam-

bém existem modelos aplicados em meios porosos, ondas e transferência de calor, sendo, esse último, abordado nesse trabalho.

Como exemplo, pode-se citar um modelo matemático utilizado por Núñes et al. [18] para estudar o escoamento de fluidos miscíveis incompresíveis em meios porosos homogêneos com razão de mobilidade adversa, cujo objetivo foi entender a influência dos parâmetros em processos de recuperação de reservatórios de petróleo. A abordagem proposta pelos autores mostrou influência da permeabilidade e da mobilidade na recuperação desses reservatórios.

Por outro lado, Lindemberg et al. [8] apresentaram resultados da modelagem sísmica em meios com fortes descontinuidades de propriedades físicas, tendo a Bacia do Amazonas como referência. Em meio de geologia complexa, como resultado dos experimentos realizados, foi observada a influência significativa das reflexões múltiplas devido à camada de alta velocidade. Esse fato provocou maior perda de energia e dificultou a interpretação dos alvos. Por esta razão os autores recomendaram, então, a integração de dados de superfície com os de poço, com o objetivo de obter melhor imagem dos alvos abaixo das soleiras de diabásio.

Já Carvalho e Pereira [5] aplicaram a modelagem matemática no estudo de crescimento populacional em Seropédica, Estado do Rio de Janeiro. Os autores adotaram, por meio de teoria de equações diferenciais, um modelo de crescimento clássico, cuja equação diferencial é linear e de primeira ordem. Posteriormente, um segundo modelo, com maior precisão, foi analisado com a utilização de uma equação diferencial não linear. A principal diferença entre os dois métodos, além da precisão, está em como calcular as constantes indeterminadas da solução de cada modelo. Com esta solução em mãos e, com a utilização de dados fornecidos pelo Instituto Brasileiro de Geografia e Estatística (IBGE), foi possível estimar a taxa de crescimento ou decrescimento, assim como o comportamento da população na região de interesse.

Modelos matemáticos podem ser combinados com recursos computacionais, como em Oliveira, Barreto e Pasqua [19], onde alguns métodos de emulação computacional de efeitos de distorção de guitarras elétricas e amplificadores valvulados foram revisados.

Nesse contexto envolvendo a Matemática e a Computação, segundo Cuminato e Júnior [7], a modelagem matemática cresceu muito nos últimos anos porque, com o salto tecnológico dos computadores, é possível entender melhor o evento modelado e analisar o comportamento, possibilitando a simulação de possíveis mudanças, assim como a análise das soluções geradas. Em suma, com o advento dos computadores é possível realizar

estudos com uma agilidade que não seria possível manualmente.

Em particular, no que se refere à modelagem de problemas de transferência de calor, as equações diferenciais variam muito de acordo com o problema devido à restrições conhecidas como problemas de valor inicial e de contorno [7]. Uma equação do calor sofre algumas variações de acordo com a situação estudada. Pode-se ter, por exemplo, um problema que envolva uma barra de um determinado tipo de material e a mesma mantenha a temperatura nas extremidades sempre nula ou, outro problema que envolva uma chapa de material qualquer e que, em seu contorno, a temperatura também seja nula. Às vezes, é necessário analisar ainda quão rapidamente uma chapa, ou barra, esquenta ou esfria com o decorrer do tempo. Para cada problema, uma abordagem diferente deve ser utilizada.

Em Costa e Dias [6], uma análise dos aspectos teóricos e computacionais do problema da condução do calor foi abordada utilizando as soluções analíticas, cuja solução é dada por uma série infinita, e aproximada numericamente pelo Método das Diferenças Finitas. Embora seja muito comum trabalhar com o Método das Diferenças Finitas (MDF) em problemas de transferência de calor, é possível utilizar outros métodos, como Método dos Volumes Finitos (MVF) ou Método dos Elementos Finitos (MEF).

Problemas que envolvem transferência de calor são de grande relevância para as engenharias e, por isso, esse trabalho apresenta a modelagem computacional de um problema de transferência de calor, resolvendo-o por meio das abordagens direta e inversa. O assunto é de interesse em cenários como: microprocessadores de computadores que não podem esquentar demais para que seu funcionamento não seja afetado; geração de energia através do aquecimento de placas solares; manutenção de determinadas condições climáticas com o isolamento de tetos e paredes de casas e edifícios. Além disso, o fenômeno da transferência de calor está presente em diversos ramos do conhecimento humano, o que pode ser percebido em vários equipamentos que contribuem para o nosso bem-estar e qualidade de vida, tais como aparelhos de ar-condicionado de residências e escritórios, radiadores dos motores dos carros e caldeiras industriais, dentre outros [12].

Outro ramo do conhecimento em que a transferência de calor é importante é o alimentício. Segundo Resende e Vivaldo Jr. [22], para o desenvolvimento de cálculos de transferência de calor que estão envolvidos nos projetos dos equipamentos de refrigeração e armazenamento de alimentos, é necessário o conhecimento das propriedades termofísicas dos mesmos, uma vez que a estrutura do alimento afeta o tipo de formulação matemática para a avaliação da condutividade térmica efetiva. No trabalho realizado pelos referi-

dos autores, o objetivo foi determinar experimentalmente a condutividade e difusividade térmica em função da temperatura do sistema. A análise ocorreu durante o processo de congelamento, ao comparar dados com propriedades estimadas, usando modelos encontrados na literatura, como ferramenta na avaliação do fenômeno de transferência de calor durante o congelamento de polpas de frutas embaladas em sacos de polietileno e acondicionados em caixas.

Ainda no que se refere à transmissão de calor, um estudo realizado por Yoalbys et. al. [23], em Moa (Cuba), avalia os processos fundamentais de transferência de calor que ocorrem durante a secagem natural de minérios lateríticos com a finalidade de determinar o modo predominante de transferência de calor. Os minérios foram submetidos à secagem solar a céu aberto para reduzir o teor de umidade antes de incorporá-los ao processo convencional de secagem térmica. Os autores registraram os valores da umidade, velocidade e temperatura do ar, além da radiação solar para avaliar a transferência de calor. Na secagem solar de minas lateríticas ferroniqueliferas a céu aberto, nas condições analisadas, predominou dois modos de transmissão de calor: convecção seguida de radiação.

No entanto, a modelagem de um problema real, seja ele um problema de transferência de calor ou não, requer o conhecimento e informações de alguns parâmetros que nem sempre são de fácil obtenção através de experimentação. Alternativamente, é utilizada uma abordagem inversa para o problema em questão, baseada em técnicas de otimização.

Aplicar métodos de otimização para solucionar problemas é uma abordagem utilizada por muitos autores com o objetivo de encontrar um conjunto de parâmetros considerados ótimos que minimizem (ou maximizem) uma dada função, denominada função objetivo. Para isso são utilizados métodos determinísticos e/ou estocásticos. Métodos determinísticos geralmente convergem mais rápido para a solução, porém podem ficar presos em ótimos locais, o que pode ser evitado utilizando fatores aleatórios (randômicos) [29]. Isso significa que, às vezes, o método determinístico não consegue sair de um ótimo local e o método estocástico realiza uma pequena perturbação na região fazendo com que outra região do intervalo de busca seja consultada, encontrando uma solução melhor.

Silva Neto, Becceneri e Campos Velho [29] aplicaram vários métodos de otimização para resolver problemas inversos em transferência radiativa, incluindo os métodos Luus-Jaakola e Algoritmo de Colisão de Partículas. Os referidos métodos também foram utilizados por Telles [30] para analisar o comportamento hidráulico de um rio com base na estimativa dos coeficientes que controlam os processos que ocorrem em uma bacia hidrográfica e rede de drenagem. Em Telles [30], os parâmetros do problema também

foram estimados utilizando as hibridizações Luus-Jaakola com Levenberg-Marquardt e Algoritmo de Colisão de Partículas com Levenberg-Marquardt, onde este último é um método determinístico.

Após a utilização dos métodos em questão para estimar parâmetros em corpos hídricos, tendo o método Luus-Jaakola apresentado um melhor desempenho quando comparado ao Algoritmo de Colisão de Partículas, neste trabalho, pôde-se observar também que este último método é eficaz em escapar de mínimos locais, devido à sua probabilidade de aceitar novas soluções que nem sempre são melhores do que as já encontradas. Por outro lado, segundo Linga, Al-Saifi e Englezos [13], Luus e Jaakola mostraram, com sucesso, que seu método poderia ser usado para resolver problemas difíceis de otimização com vários mínimos locais. Este procedimento usa pontos de pesquisa aleatórios e contração sistemática da região de pesquisa.

Além disso, no que se refere ao método Luus-Jaakola, Telles et al. [31] realizaram uma modificação no mesmo – com posterior aplicação na estimativa dos coeficientes de dispersão da Equação de Transporte Bidimensional – e os resultados apresentaram melhorias em relação à versão clássica, sendo possível estimar os parâmetros de interesse com menos avaliações da função objetivo.

A hipótese considerada no presente trabalho é que propostas de modificações, bem como hibridizações entre os métodos Luus-Jaakola e Algoritmo de Colisão de Partículas, de maneira a extrair o que ambos têm de melhor, alcancem resultados satisfatórios com custos computacionais (número de avaliações da função objetivo) reduzidos para resolução do problema abordado.

1.2 Objetivos

O objetivo geral deste trabalho é resolver um problema de transferência de calor, onde os parâmetros necessários à solução do modelo matemático são obtidos por métodos de otimização baseado em uma abordagem inversa em que os resultados numéricos das temperaturas são comparados com os dados experimentais descritos por Carollo [4].

Com base no objetivo geral apresentado, os objetivos específicos são detalhados a seguir:

• Aplicar o Método das Diferenças Finitas para resolver um problema de transferência de calor proposto por Carollo [4];

• Aplicar as formulações explícita e implícita para a solução do problema direto e comparar os resultados obtidos com os resultados do experimento realizado por Carollo [4];

• Testar e analisar várias configurações da malha computacional, afim de encontrar uma melhor configuração que ajuste os resultados numéricos aos dados experimentais de forma satisfatória;

• Estimar os parâmetros de interesse através do problema inverso, utilizando os métodos Luus-Jaakola e Algoritmo de Colisão de Partículas com o objetivo de obter valores aproximados de temperatura em relação ao experimento realizado;

• Modificar os métodos de otimização Luus-Jaakola e Algoritmo de Colisão de Partículas com o objetivo de analisar se tais modificações obtém soluções melhores (menor número de avaliações da função objetivo) em relação às versões clássicas dos algoritmos;

• Hibridizar os métodos de otimização abordados no presente trabalho e aplicar o problema inverso com o objetivo de analisar se as soluções também melhoram em relação às versões clássicas dos algoritmos.

1.3 Estrutura da Dissertação

Este capítulo abordou uma introdução sobre o problema de transferência de calor e a utilização dos métodos empregados ao longo do texto, onde os objetivos gerais e específicos são explanados, bem como, relevância e justificativa para realizar a presente pesquisa.

Na sequência, no Capítulo 2, é abordado o problema direto de transferência de calor, onde é apresentada a descrição e solução do mesmo por meio do Método das Diferenças Finitas. Neste capítulo, os resultados do problema direto também são analisados e comparados com os dados experimentais de temperatura descritos em Carollo [4].

Já no Capítulo 3, o problema inverso é formulado e os métodos de otimização Luus-Jaakola e Algoritmo de Colisão de Partículas são apresentados em suas versões clássicas. Em seguida, modificações e hibridizações são realizadas nos mesmos de forma a se obter resultados melhores (menor número de avaliações da função objetivo) em relação às versões clássicas dos respectivos algoritmos.

No Capítulo 4 é feita a estimativa dos parâmetros do problema de transferência de calor utilizando os métodos de otimização nas versões clássicas, modificadas e hibridizadas, descritas no capítulo anterior, assim como a análise dos resultados obtidos e as melhorias

alcançadas.

Por fim, no Capítulo 5 são feitas as conclusões com base nos resultados obtidos nos capítulos anteriores e sugestões para futuros trabalhos envolvendo o tema.

Capítulo 2

Descrição e Solução do Problema Direto de Transferência de Calor

Neste capítulo é descrito e definido o problema direto de transferência de calor, bem como os conceitos matemáticos relacionados ao problema abordado neste trabalho e a equação que o modela. São apresentados os diferentes processos de transferência de calor: condução, convecção e radiação. O conceito de distância entre dois pontos, definição importante para conceituar bola aberta e conjunto fechado, que por sua vez se fazem necessárias para a definição de fronteira, também são apresentados, seguidos da definição de Equações Diferenciais Parciais, para enfim, abordar as condições de fronteira do problema. Para finalizar, o Método das Diferenças Finitas é apresentado e aplicado ao problema proposto utilizando as formulações explícita e implícita, sendo os resultados obtidos, analisados e investigados no que tange às malhas computacionais empregadas.

2.1 Conceitos Básicos em Transferência de Calor

Termodinâmica, de acordo com Borgnakke e Sonntag [2], é um campo que se relaciona com a ciência da energia, tendo como foco o armazenamento e processo de conversão. Para Incropera et al. [10], essa energia pode ser transferida pelas interações de um sistema com seu entorno.

Ao passo que a termodinâmica lida com os estados finais do processo, não fornecendo informações sobre a natureza da interação, bem como a taxa de tempo em que ocorre, a transferência de calor explica como o calor é transferido e a que taxas ele ocorre.

Transferência de calor, ou simplesmente calor, é a energia térmica em trânsito devido a uma diferença de temperatura espacial [10]. Ainda segundo Incropera et al. [10],

há diferentes processos de transferência de calor, divididos em condução, convecção e radiação. Apesar de possuírem naturezas diferentes, esses processos podem ocorrer de maneira simultânea. A seguir, é descrita de forma sucinta cada um desses processos.

2.1.1 Condução

O processo de condução se dá quando há contato direto entre as substâncias em modo estacionário, ou seja, após um tempo em equilíbrio, a distribuição espacial das temperaturas no objeto condutor não mais se altera, e a transferência ocorre átomo a átomo. Esse fenômeno ocorre devido a diferença de temperatura entre as substâncias. Na Figura 2.1 é mostrado o processo de transferência de calor por condução, onde T_1 (^oC) e T_2 (^oC) são temperaturas $(T_1 \geq T_2)$, L (m) o comprimento do material que está sob análise e q''_{cond} (W/m^2) a taxa de fluxo de calor na direção x, dada por:

$$q''_{cond} = -k\frac{dT}{dx}, \tag{2.1}$$

onde k é uma constante de proporcionalidade denominada condutividade térmica (W/mK) e $\frac{dT}{dx}$ o gradiente de temperatura na direção de x.

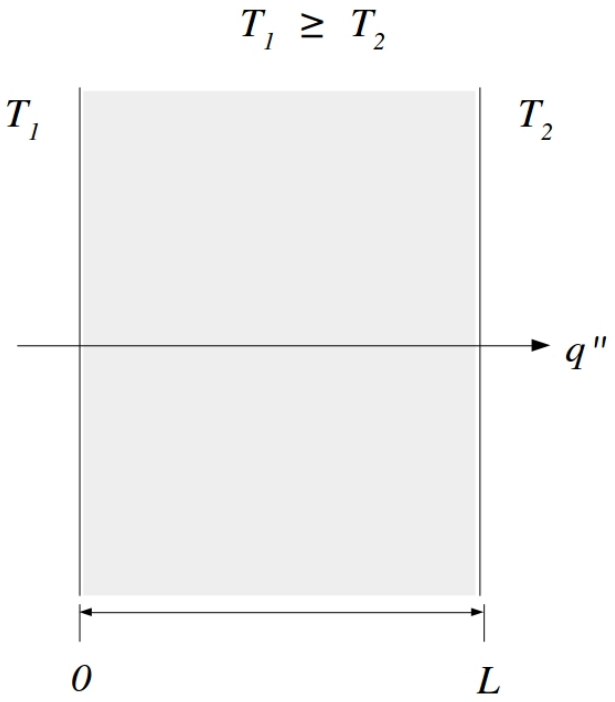


Figura 2.1: Condução de calor através de um sólido ou fluido estacionário.
Fonte: Adaptado de Incropera et al. [10].

Pela Lei de Fourier, na Equação (2.1), a constante de proporcionalidade k possui um sinal negativo porque o calor se move na direção contrária ao gradiente de temperatura.

Ao considerar L como o comprimento do material que está sob análise, caso o material seja homogêneo, o gradiente de temperaturas presente na Equação (2.1) é dado pela Equação (2.2).

$$\frac{dT}{dx} = \frac{T_2 - T_1}{L} \tag{2.2}$$

Assim, a taxa de fluxo de calor na direção x, q''_{cond} (W/m^2), presente na Figura 2.1 e descrita pela Equação (2.1), é dada pela Equação (2.3).

$$q''_{cond} = k\frac{T_1 - T_2}{L} = k\frac{\triangle T}{L} \tag{2.3}$$

Já a taxa de calor por condução, q_{cond} (W), é dada por $q_{cond} = q''_{cond}\ A$, onde A (m^2) é a área de transferência de calor no material que está sob análise.

2.1.2 Convecção

A convecção se dá através de um fluido em movimento em uma superfície, onde ambos estão em temperaturas diferentes e, segundo Çengel e Ghajar [32], é classificada em convecção forçada ou convecção natural (ou livre) dependendo de como o movimento do fluido é iniciado.

Qualquer movimento do fluido é causado por meios naturais quando se trata da convecção natural, como é o caso do efeito empuxo que se manifesta com fluidos quentes subindo e fluidos frios descendo [32].

Já na convecção forçada o fluido é forçado a escoar sobre a superfície ou dentro de um duto, o que a classifica, segundo Çengel e Ghajar [32], como convecção forçada externa ou interna. Os meios utilizados para o escoamento forçado podem ser a presença de uma bomba ou ventilador.

Na Figura 2.2 a seguir, tem-se uma representação da transferência de calor por convecção de uma superfície para um fluido em movimento.

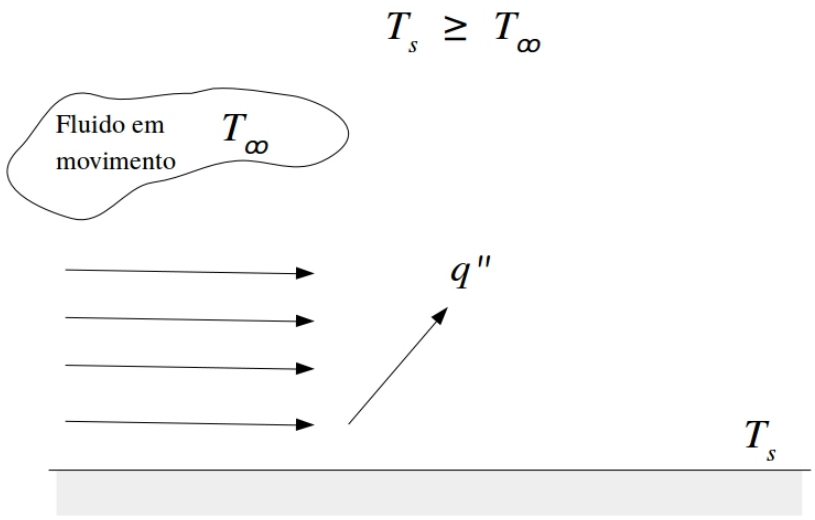


Figura 2.2: Convecção de calor entre uma superfície e um fluido em movimento. Fonte: Adaptado de Incropera et al. [10].

A troca de calor entre um sólido e um fluido pode ser descrita através da Lei de Newton do resfriamento, dada pela Equação (2.4),

$$q_{conv} = hA_s(T_s - T_\infty), \tag{2.4}$$

onde q_{conv} (W) é a taxa de calor, h (W/m^2K) é o coeficiente de transferência de calor por convecção, A_s (m^2) é a área de transferência de calor, T_s (^oC) é a temperatura da superfície e T_∞ (^oC) é a temperatura do fluido suficientemente longe da superfície.

A Equação (2.4) também pode ser representada como na Equação (2.5),

$$q''_{conv} = \frac{q_{conv}}{A_s} = h(T_s - T_\infty), \tag{2.5}$$

onde q''_{conv} (W/m^2) é o fluxo de calor.

Segundo Çengel e Ghajar [32], todas as observações experimentais indicam que um fluido em contato direto com um sólido "adere" à superfície por causa dos efeitos viscosos, logo, não há escorregamento e essa condição é conhecida como condição de não deslizamento. Sendo assim, tem-se a Equação (2.6),

$$q''_{conv} = q''_{cond} = -k_f \left.\frac{\partial T}{\partial x}\right|_{x=0}, \tag{2.6}$$

onde k_f (W/m^2K) é a constante de proporcionalidade do fluido.

A transferência de calor por convecção, a partir de uma superfície sólida para um fluido, é simplesmente a transferência de calor por condução a partir da superfície sólida para a camada de fluido adjacente à superfície [32]. Desta forma, pode-se igualar as Equações (2.5) e (2.6), obtendo a Equação (2.7),

$$h = \frac{-k_f \left.\frac{\partial T}{\partial x}\right|_{x=0}}{T_s - T_\infty}, \tag{2.7}$$

onde é possível determinar o coeficiente de transferência de calor por convecção quando a distribuição de temperatura no interior do fluido é conhecida.

2.1.3 Radiação

A radiação é um terceiro mecanismo de transferência de calor. Ela é emitida por cada ponto de uma superfície plana em todas as direções no hemisfério acima da superfície [32], sendo o calor transferido, por meio de ondas eletromagnéticas, de uma superfície para outra, levando em consideração o fato dessas superfícies se encontrarem separadas, não havendo a necessidade de um meio material. Na Figura 2.3 é mostrada a troca de calor de radiação líquida entre duas superfícies.

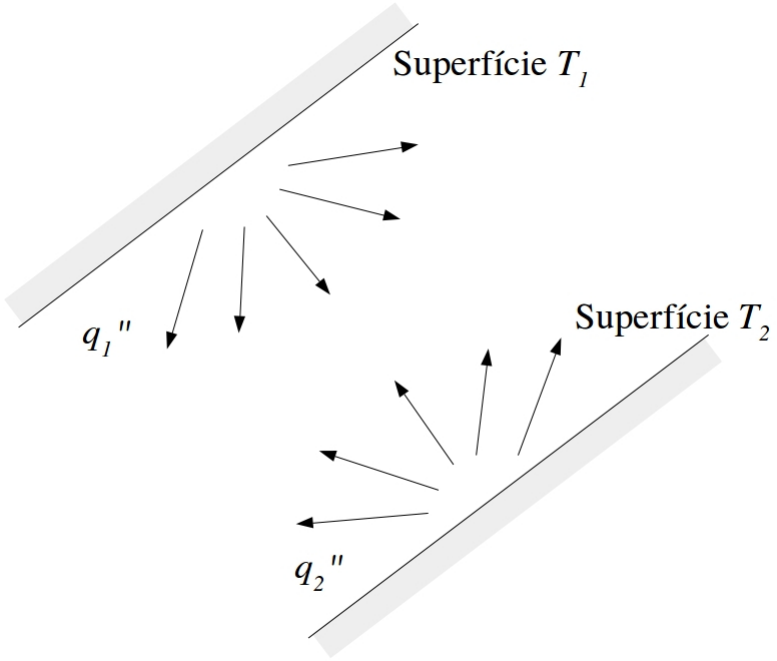


Figura 2.3: Troca de calor de radiação líquida entre duas superfícies.
Fonte: Adaptado de Incropera et al. [10].

Um corpo em uma temperatura absoluta acima de zero emite uma radiação em todas as direções ao longo da faixa de onda. A quantidade de radiação depende do material do corpo e da condição e temperatura da superfície, sendo assim, diferentes corpos emitem quantidades diferentes de radiação por unidade de área, mesmo que esteja sob a mesma temperatura.

Segundo Çengel e Ghajar [32], um corpo negro ou radiador ideal é um perfeito emissor e absorvedor de radiação, pois emite a maior potência, por unidade de área, para uma determinada temperatura, e absorve toda radiação que recebe, sem refletir qualquer parcela da mesma. Um corpo negro é um emissor difuso, pois emite energia de radiação uniformemente em todas as direções.

De forma experimental, Joseph Stefan, em 1879, determinou a energia de radiação, E_b, emitida por um corpo negro por unidade de tempo e por unidade de área, sendo expressa como na Equação (2.8),

$$E_b = \sigma(T_s^4), \tag{2.8}$$

onde σ é a constante de Stefan-Boltzmann determinada por $\sigma = 5,67 \times 10^{-8} \quad W/m^2K^4$.

Após as considerações físicas abordadas nesta seção, apresenta-se alguns conceitos matemáticos importantes para o desenvolvimento da modelagem matemática do problema de transferência de calor proposto neste trabalho.

2.2 Modelagem Matemática do Problema Proposto

No que se refere à modelagem de problemas reais que envolve a taxa de variação de uma determinada grandeza em relação a uma ou mais variáveis, pode-se dizer, de forma simplista que, quando se trabalha com problemas que envolvem apenas uma variável, tem-se que usualmente o mesmo é descrito por Equações Diferenciais Ordinárias (EDO), enquanto em problemas que envolvem duas ou mais variáveis, essa representação é dada por equações chamadas de Equações Diferenciais Parciais (EDP). Nesse trabalho o foco são as EDP's, pois os problemas envolvem as variáveis tempo e espaço, uma vez que o interesse é analisar o comportamento de uma estrutura ao longo do espaço no decorrer de determinado tempo.

Antes de definir formalmente uma Equação Diferencial Parcial e abordar a modelagem matemática do problema proposto, são apresentados alguns conceitos importantes para a

sua compreensão, os quais envolvem a definição de bola aberta, conjunto fechado, fecho e fronteira. A definição de distância entre dois pontos, ou vetores, também é abordada, uma vez que, para definir tanto uma bola aberta quanto uma bola fechada, se faz necessário conhecer tal definição. Todos esses conceitos levam a uma correta compreensão ao se abordar problemas de fronteira.

Definição 1: Sejam dois pontos, ou vetores, $\boldsymbol{x} = (x_1, ..., x_n)$ e $\boldsymbol{y} = (y_1, ..., y_n)$ em $\mathbb{R}^n$, a distância euclidiana entre $\boldsymbol{x}$ e $\boldsymbol{y}$ é dada pela Equação (2.9) [11].

$$|\boldsymbol{x} - \boldsymbol{y}| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.9}$$

Definição 2: Considere r um raio positivo qualquer, ou seja, $r > 0$. O conjunto de todos os pontos cuja distância a um ponto fixo $\boldsymbol{x_0} \in \mathbb{R}^n$ seja menor do que r é chamado de *bola aberta centrada em* $\boldsymbol{x_0}$ *de raio* r e denotada por $B(\boldsymbol{x_0}; r) = \{\boldsymbol{x} \in \mathbb{R}^n : |\boldsymbol{x} - \boldsymbol{x_0}| < r\}$ [11].

Definição 3: Um subconjunto $\Omega \subseteq \mathbb{R}^n$ é aberto se, para todo $\boldsymbol{x_0} \in \Omega$, existe uma bola aberta centrada em $\boldsymbol{x_0}$ que está completamente contida em Ω [11].

Definição 4: Um subconjunto F de $\mathbb{R}^n$ é fechado se seu complementar é aberto, ou seja, $\mathbb{R}^n - F = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x} \notin F\}$ é aberto [11].

Definição 5: Dado $A \subseteq \mathbb{R}^n$, o fecho de A é o menor conjunto fechado que o contém e é denotado por $\overline{A}$ [11].

Definição 6: O interior de A é o maior conjunto aberto contido em A e é denotado por $\mathring{A}$ [11].

Definição 7: O *bordo* ou *fronteira*, denotado por ∂A, é dado por ∂A=$\{\boldsymbol{x} \in \overline{A} : \boldsymbol{x} \notin \mathring{A}\}$ [11].

Definição 8: Equação Diferencial Parcial é uma equação que envolve duas ou mais variáveis independentes $x_1, x_2, x_3, ..., x_n$ e variáveis dependentes, ou seja, derivadas parciais de uma função $u(x_1, x_2, x_3, ..., x_n)$ [11]. Uma Equação Diferencial Parcial tem a forma da Equação (2.10),

$$H\left(x_1, ..., x_n, u, \frac{\partial u}{\partial x_1}, ..., \frac{\partial u}{\partial x_n}, \frac{\partial^2 u}{\partial x_1^2}, ..., \frac{\partial^2 u}{\partial x_1 \partial x_n}, ..., \frac{\partial^k u}{\partial x_n^k}\right) = 0, \tag{2.10}$$

onde $\boldsymbol{x} = (x_1, ...x_n) \in \Omega$, Ω é um subconjunto aberto de $\mathbb{R}^n$, H é uma função dada e

$u = u(\boldsymbol{x})$ é a função que se deseja determinar [11].

Existem diferentes notações para representar derivadas parciais de u. Caso se queira representar a derivada parcial de u, de primeira ordem, em relação à variável x_1, a derivada poderá ser denotada por $\frac{\partial u}{\partial x_1}, u_{x_1}, \partial_{x_1} u$ ou até mesmo como $D_1 u$. Da mesma forma, as derivadas de segunda ordem, também em relação a x_1, poderão ser denotadas por $\frac{\partial^2 u}{\partial x_1^2}, u_{x_1 x_1}, \partial_{x_1^2} u$ ou $D_1^2 u$.

Para o caso de variáveis diferentes, por exemplo, x_1 e x_2, derivando primeiro em relação a x_1 e depois em relação a x_2, pode-se denotar as derivadas como $\frac{\partial^2 u}{\partial x_2 \partial x_1}, u_{x_1 x_2}, \partial_{x_2} \partial_{x_1} u$ ou $D_2 D_1 u$.

Segundo Burden et al. [3], Equações Diferenciais Parciais se classificam de forma parecida com a classificação de seções cônicas. Considere a Equação (2.11), linear de segunda ordem, com coeficientes constantes ou que dependem apenas das variáveis x_1 e/ou x_2.

$$au_{x_1 x_1} + 2bu_{x_1 x_2} + cu_{x_2 x_2} + du_{x_1} + eu_{x_2} + fu = g \tag{2.11}$$

Diz-se que a Equação Diferencial Parcial é classificada como:

- elíptica se $b^2 - ac < 0$;
- parabólica se $b^2 - ac = 0$;
- hiperbólica se $b^2 - ac > 0$.

Como os coeficientes a, b e c, presentes na Equação (2.11), dependem de x_1 e x_2, Cuminato e Júnior [7] afirmam que a equação pode mudar de classificação em diferentes intervalos do domínio. Nos casos em que $a = b = c = 0$, a Equação (2.11) se transforma em $du_{x_1} + eu_{x_2} + fu = g$ e a equação diferencial passa a ser uma equação de primeira ordem.

Outro conceito importante no que se refere às Equações Diferenças Parciais são as condições de fronteira. Condições de fronteira são condições extras que envolvem um problema e que necessitam serem consideradas para solucionar a equação que modela o mesmo, pois indicam o comportamento da grandeza de interesse nas extremidades do domínio.

Segundo Cuminato e Júnior [7], quando se conhece o valor da função nas fronteiras, tem-se um problema de condição de fronteira de Dirichlet. No entanto, quando não se

conhece o valor da função, mas o valor das derivadas direcionais na fronteira é conhecido, o problema passa a ser um problema de condição de fronteira de Neumann. Dependendo do problema, pode-se ter as duas situações envolvidas, ou seja, uma extremidade com valor da função conhecido e outra em que se sabe apenas o valor da derivada direcional. Caso o problema possua condições de contorno de Dirichlet e de Neumann, a condição é chamada de condição mista ou condição de Robin.

2.2.1 Modelo Matemático

O problema investigado neste trabalho constitui-se de um experimento realizado por Carollo [4], o qual trata-se de uma amostra constituída de aço inox, modelo AISI 304, de comprimento L, que está compreendida entre um aquecedor e um isolante térmico de forma que se deseja obter a temperatura da mesma, ao longo do tempo, onde se encontra um termopar localizado na parte inferior de sua superfície, com o objetivo de se aferir tal temperatura, conforme representado na Figura 2.4.

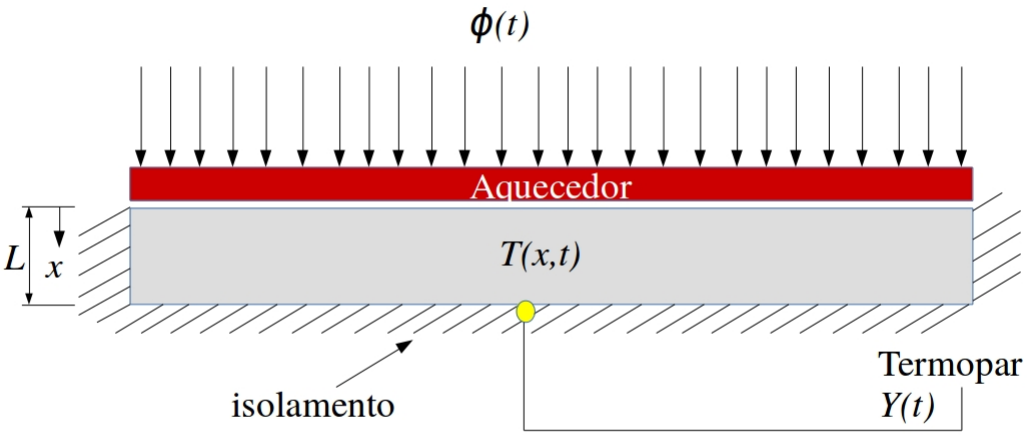


Figura 2.4: Represetação esquemática do problema proposto.
Fonte: Adaptado de Carollo [4].

Para a realização do experimento, foi aplicado um fluxo de calor $\phi(t)$, na parte superior da placa, ou seja, em $x = 0$, durante 160 s, o qual foi dividido em três etapas, da seguinte

maneira: na primeira etapa, aplicou-se um fluxo de 2640 W/m^2 no intervalo de 0 a 20 s; na segunda etapa, no intervalo de 20 a 140 s, a intensidade do fluxo aplicada foi de 660 W/m^2; nos últimos 20 s, a fonte térmica foi desligada, resultando num fluxo de calor nulo, conforme Figura 2.5.

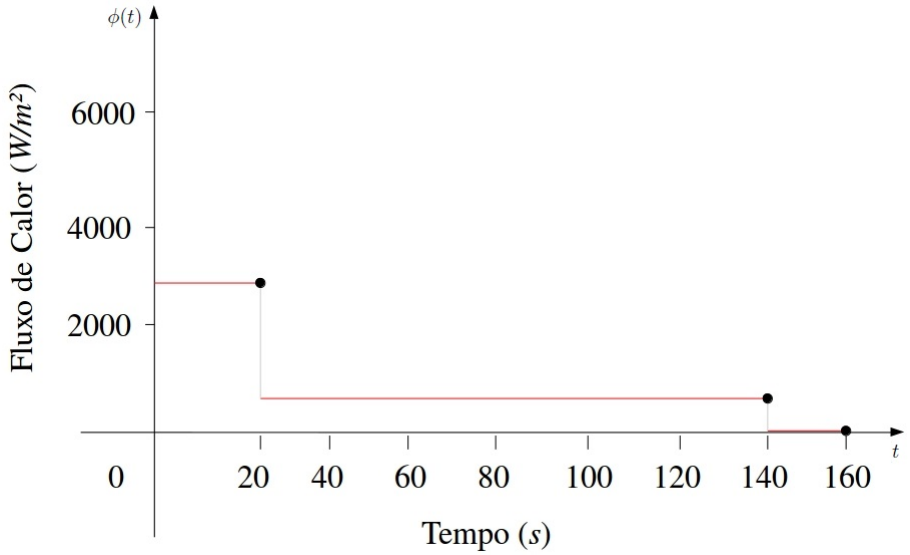


Figura 2.5: Perfil do fluxo de calor aplicado à placa.
Fonte: O Autor, 2019.

Considerando que a variação da temperatura ao longo da placa ocorre apenas no sentido vertical, de cima para baixo, representado pela variável x, o modelo matemático para esse problema é representado pela Equação (2.12).

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{\rho c_p}{\lambda}\frac{\partial T(x,t)}{\partial t} \tag{2.12}$$

Com condições de contorno dadas pelas Equações (2.13) e (2.14).

$$-\lambda\frac{\partial T(x,t)}{\partial x} = \phi(t) \text{ em } x = 0 \text{ e } t > 0 \tag{2.13}$$

$$\frac{\partial T(x,t)}{\partial x} = 0, \text{ em } x = L \text{ e } t > 0 \tag{2.14}$$

E condição inicial,

$$T(x,0) = \varphi(x), \text{ em } 0 \leq x \leq L, \tag{2.15}$$

onde ρc_p é a capacidade volumétrica de calor (s/m), λ é a condutividade térmica (W/mK),

x é a coordenada espacial, L é o comprimento da placa (m), t é o tempo (s), $\frac{\partial T(x,t)}{\partial t}$ é a variação da temperatura (que depende da posição x e do tempo t) em relação à variação do tempo. Já a temperatura inicial da placa é de 18,84 ^oC, ou seja, $T(x,0) = 18,84$, $0 \leq x \leq L$.

Na seção a seguir, apresenta-se o Método das Diferenças Finitas e aplica-se as formulações explícita e implícita para a solução das Equações (2.12) à (2.15) com a finalidade de resolver numericamente o problema direto de transferência de calor.

2.3 Método das Diferenças Finitas

Em estudos envolvendo equações diferenciais existe o que se chama de solução analítica e solução aproximada. Na solução analítica o domínio do problema é contínuo, enquanto para soluções aproximadas é preciso discretizar o domínio. Essas soluções, obtidas através da discretização do domínio, geram uma solução aproximada que deve ser analisada para saber se satisfaz o problema. Dependendo dos resultados gerados, é possível aproximar a solução real do problema por essa aproximação.

Discretizar é transformar um problema contínuo em um problema discreto finito, onde, existindo apenas uma variável independente, o domínio é um intervalo e o problema envolve uma Equação Diferencial Ordinária. Se existirem duas ou mais variáveis independentes, o domínio deixa de ser um intervalo e passa a ser uma região no plano ou no espaço e, nesse caso, o problema envolve uma Equação Diferencial Parcial [7].

Para discretizar um problema é preciso gerar uma malha com uma determinada quantidade de nós do domínio. Entretanto, é preciso ter cautela na escolha dessa quantidade de nós, pois quanto mais nós a malha possuir, espera-se estar mais próximo da solução analítica. Porém, o custo computacional pode ser muito alto, levando muito tempo ou, até mesmo, não resolvendo o problema.

Segundo Silva, Telles e Semaan [26] em métodos numéricos existem várias técnicas de soluções numéricas e, dentro desse grande universo, o Método de Diferenças Finitas é uma técnica para obtenção de soluções aproximadas de equações diferenciais, sendo que, para se obter tais soluções, deve-se discretizar o domínio em que se está trabalhando. A ideia fundamental que está por trás dos métodos numéricos é a de discretização do contínuo, ou seja, dado um domínio contínuo, deve-se discretizá-lo de forma a encontrar soluções aproximadas da solução analítica, quando essas existam.

O Método das Diferenças Finitas tem como base a fórmula de Taylor, útil em várias situações, como arredondamento de números, extração de raízes, etc. A fórmula de Taylor é um somatório infinito envolvendo derivadas onde, quanto mais parcelas forem desenvolvidas, mais próximo o resultado obtido será da solução exata. Uma vez que o somatório é infinito, não é possível concluí-lo, por isso, ao utilizar a fórmula de Taylor, existem parcelas não calculadas que podem ser chamadas de resto em relação à solução exata.

No Método das Diferenças Finitas, uma vez discretizado o domínio, deve-se substituir as derivadas presentes na equação diferencial por aproximações que envolvam apenas valores numéricos da função [7].

A seção a seguir aborda procedimentos para encontrar as aproximações para as derivadas utilizando as fórmulas avançada, atrasada e centrada, as quais tem como base as séries de Taylor.

2.3.1 Fórmulas Avançada, Atrasada e Centrada

Uma ferramenta matemática fundamental para se trabalhar com aproximações de uma função e suas derivadas é a fórmula de Taylor, pois é possível fazer a relação entre os valores de uma função e suas derivadas em um ponto x, utilizando valores dessa mesma função na vizinhança do referido ponto. Segundo Porto e Silva [21], a fórmula de Taylor tem grande importância na Análise Matemática, pois é através dela que se define o conceito de diferenciabilidade, planos e retas tangentes.

Considere que $f(x)$ possui derivadas de ordem $n+1$. Fazendo uma expansão de Taylor em $(x+h)$, $h>0$, tomando como base o ponto x, tem-se a Equação (2.16),

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + ... + \frac{h^n}{n!}f^{(n)}(x) + \frac{h^{(n+1)}}{(n+1)!}f^{(n+1)}(\xi), \qquad (2.16)$$

onde $x < \xi < x+h$.

O termo $\dfrac{h^{(n+1)}}{(n+1)!}f^{(n+1)}(\xi)$ presente na Equação (2.16), também chamado de resto, representa o erro de aproximação de $f(x+h)$ pelo polinômio de Taylor $P(h)$ de grau n na variável h, dado pela Equação (2.17). Logo, quando esse resto tende a zero, a solução aproximada do problema tende a ser igual a solução analítica, quando esta última existir.

$$P_n(h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + ... + \frac{f^{(n)}(x)}{n!}h^n \tag{2.17}$$

Fazendo $n = 1$ na Equação (2.16), obtém-se o resultado conforme a Equação (2.18).

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(\xi) \tag{2.18}$$

Na Equação (2.18), o resto que representa o erro da solução aproximada em relação à analítica é, evidentemente, $\frac{h^2}{2!}f''(\xi)$. Isolando $f'(x)$ na Equação (2.18), obtém-se a chamada *fórmula progressiva* ou *avançada*, que é uma das maneiras de se resolver um problema utilizando o Método de Diferenças Finitas, conforme explicitado na Equação (2.19).

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(\xi) \tag{2.19}$$

Ao fazer $h = -h$ na Equação (2.18), obtém-se:

$$f(x + (-h)) = f(x) + (-h)f'(x) + \frac{(-h)^2}{2!}f''(\xi),$$

que resulta na Equação (2.20),

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \frac{h}{2!}f''(\xi), \tag{2.20}$$

também conhecida como *fórmula regressiva* ou *atrasada.*

Pode-se ainda encontrar, na literatura sobre o tema, os termos *diferença ascendente* e *descendente*, respectivamente, para diferença progressiva e regressiva [7].

Há, ainda, a chamada *fórmula centrada.* Para obtê-la, é preciso fazer $n = 2$ na Equação (2.16), tomando h e $-h$, respectivamente, conforme as Equações (2.21) e (2.22).

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(\xi_1) \tag{2.21}$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(\xi_2) \tag{2.22}$$

Subtraindo a Equação (2.22) da Equação (2.21) chega-se ao resultado explicitado na Equação (2.23),

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{3!}f'''(\xi), \tag{2.23}$$

onde $\xi \in (x-h, x+h)$ para algum $\xi \in [min\{\xi_1, \xi_2\}, max\{\xi_1, \xi_2\}]$ e $f'''(\xi) = f'''(\xi_1) + f'''(\xi_2)$. A fórmula dada pela Equação (2.23) é chamada de *fórmula centrada* ou *fórmula de diferenças centrais.*

Já a aproximação para a segunda derivada é dada pela Equação (2.24), a qual foi obtida somando-se as Equações (2.21) e (2.22),

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{h^2}{3!}f^{(4)}(\xi), \tag{2.24}$$

onde $\xi \in (x-h, x+h)$ para algum $\xi \in [min\{\xi_1, \xi_2\}, max\{\xi_1, \xi_2\}]$ e $f^{(4)}(\xi) = f^{(4)}(\xi_1) - f^{(4)}(\xi_2)$.

Para o problema de transferência de calor, presente neste trabalho, é necessário e suficiente que a fórmula seja desenvolvida até a derivada de segunda ordem. Como no processo de discretização do domínio, toma-se h próximo a 0 (zero), $h \in (0, 1)$, a diferença centrada tende à zero muito mais rápido do que o erro das diferenças avançada e atrasada, uma vez que seu erro é proporcional a h^2.

Com relação ao tempo, no Método de Diferenças Finitas existem duas abordagens que podem ser exploradas para solucionar o problema. Uma delas é a abordagem explícita e a outra implícita, sendo essa segunda mais trabalhosa de se desenvolver. A abordagem explícita é aquela em que apenas um valor da grandeza de interesse é desconhecido, enquanto na abordagem implícita há a necessidade de determinar três valores da grandeza de interesse de forma simultânea, pois trata-se de um problema unidimensional.

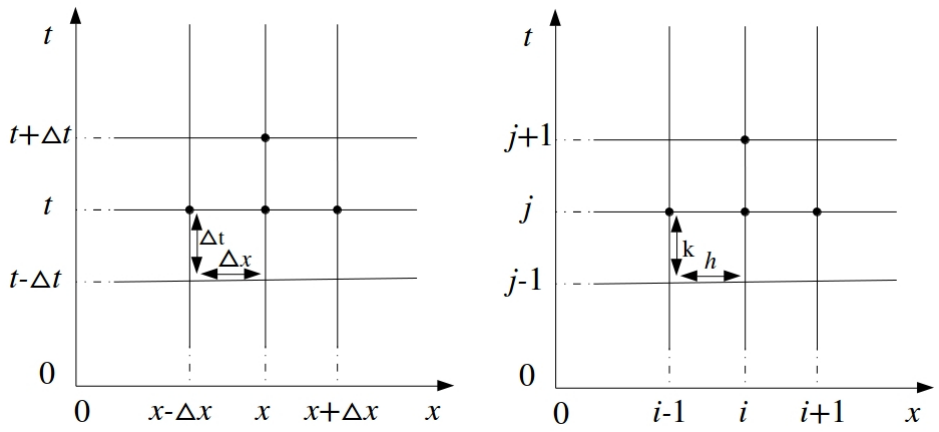
Nas seções a seguir são descritas as abordagens explícita e implícita, bem como sua aplicação no problema proposto.

2.3.2 Aplicação da Formulação Explícita ao Problema Proposto

Para aplicação do Método das Diferenças Finitas com formulação explícita considere $\triangle t = k$ e $\triangle x = h$, sendo $\triangle t$ a variação em relação ao tempo e $\triangle x$ a variação em relação ao espaço, além da seguinte notação indicial: $t - k = j - 1$, $t = j$, $t + k = j + 1$ e $x - h = i - 1$, $x = i$, $x + h = i + 1$.

Na Figura 2.6 tem-se a malha computacional conforme a formulação explícita. Nesse caso, deseja-se obter a temperatura no ponto $T_{i,j+1}$, considerando valores de $T_{i-1,j}$, $T_{i,j}$ e $T_{i+1,j}$ conhecidos. Note que a equação do problema é parabólica e isso indica uma

continuidade na função em todo o intervalo.



(a) Notação original (b) Notação indicial

Figura 2.6: Malha computacional utilizada no Método das Diferenças Finitas com formulação explícita, com a equivalência entre as notações.
Fonte: O Autor, 2019.

Como a derivada em relação ao tempo é de primeira ordem, é utilizada a fórmula avançada, cujo erro é dado por $-\frac{k}{2!}f''(\xi)$, assim obtém-se a Equação (2.25).

$$\frac{\partial T(x,t)}{\partial t} \simeq \frac{T(x,t+k)-T(x,t)}{k} = \frac{T_{i,j+1}-T_{i,j}}{k} \tag{2.25}$$

Ao aplicar a fórmula centrada para o espaço, cujo erro é dado por $-\frac{h^2}{3!}f^{(4)}(\xi)$, obtém-se a Equação (2.26).

$$\frac{\partial^2 T(x,t)}{\partial x^2} \simeq \frac{T(x+h,t)-2T(x,t)+T(x-h,t)}{h^2} = \frac{T_{i+1,j}-2T_{i,j}+T_{i-1,j}}{h^2} \tag{2.26}$$

Ao considerar $\beta = \frac{\rho c_p}{\lambda}$ e aplicar os resultados obtidos nas Equações (2.25) e (2.26) na Equação (2.12), obtém-se como resultado a Equação (2.27).

$$\frac{T_{i+1,j}-2T_{i,j}+T_{i-1,j}}{h^2} - \beta\frac{T_{i,j+1}-T_{i,j}}{k} = 0 \tag{2.27}$$

Seja $u_{i,j}$ uma aproximação de $T_{i,j}$. A Equação (2.27) pode ser reescrita conforme indica a Equação (2.28).

$$\frac{k}{\beta h^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) - u_{i,j+1} + u_{i,j} = 0 \tag{2.28}$$

Isolando $u_{i,j+1}$, pois trata-se da única variável desconhecida, obtém-se:

$$u_{i,j+1} = u_{i,j} + \frac{k}{\beta h^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})$$

$$u_{i,j+1} = (1 - \frac{2k}{\beta h^2})u_{i,j} + \frac{k}{\beta h^2}(u_{i+1,j} + u_{i-1,j})$$

$$u_{i,j+1} = \frac{k}{\beta h^2}u_{i-1,j} + (1 - \frac{2k}{\beta h^2})u_{i,j} + \frac{k}{\beta h^2}u_{i+1,j}, \tag{2.29}$$

$\forall i = 1, ..., n-1$ e $j = 0, ..., m-1$, ou seja, a Equação (2.29) é válida para os nós internos do domínio espacial, onde n é o número de divisões da malha em x e m é o número de divisões da malha em t.

Ainda falta analisar as condições de contorno dadas pelas Equações (2.13) e (2.14). Em ambos os casos, é aplicada a fórmula centrada, juntamente com a estratégia do nó fictício.

Em $x = 0$, ou seja, $i = 0$, obtém-se:

$$-\lambda\frac{\partial T(x,t)}{\partial x} = \phi(t) \implies \frac{\partial T(x,t)}{\partial x} = -\frac{\phi(t)}{\lambda} \implies \frac{u_{1,j} - u_{-1,j}}{2h} = -\frac{\phi(t)}{\lambda}.$$

Logo:

$$u_{-1,j} = u_{1,j} + \frac{2h}{\lambda}\phi(t). \tag{2.30}$$

Ao substituir a Equação (2.30) na Equação (2.29) para $i = 0$, obtém-se:

$$u_{0,j+1} = \frac{k}{\beta h^2}u_{-1,j} + (1 - \frac{2k}{\beta h^2})u_{0,j} + \frac{k}{\beta h^2}u_{1,j}$$

$$u_{0,j+1} = \frac{k}{\beta h^2}(u_{1,j} + \frac{2h}{\lambda}\phi_1(t)) + (1 - \frac{2k}{\beta h^2})u_{0,j} + \frac{k}{\beta h^2}u_{1,j}$$

$$u_{0,j+1} = (1 - \frac{2k}{\beta h^2})u_{0,j} + \frac{2k}{\beta h^2}u_{1,j} + \frac{2k}{\beta h\lambda}\phi(t). \tag{2.31}$$

Em $x = L$, ou seja, $i = n$, obtém-se:

$$\frac{\partial T(x,t)}{\partial x} = 0 \implies \frac{u_{n+1,j} - u_{n-1,j}}{2h} = 0.$$

Assim:

$$u_{n+1,j} = u_{n-1,j}. \tag{2.32}$$

Ao substituir a Equação (2.32) na Equação (2.29) para $i = n$, obtém-se:

$$u_{n,j+1} = \frac{k}{\beta h^2}u_{n-1,j} + (1 - \frac{2k}{\beta h^2})u_{n,j} + \frac{k}{\beta h^2}u_{n+1,j}$$

$$u_{n,j+1} = \frac{k}{\beta h^2}u_{n-1,j} + (1 - \frac{2k}{\beta h^2})u_{n,j} + \frac{k}{\beta h^2}u_{n-1,j}$$

$$u_{n,j+1} = \frac{2k}{\beta h^2}u_{n-1,j} + (1 - \frac{2k}{\beta h^2})u_{n,j}. \tag{2.33}$$

A solução do problema, ao utilizar notação vetorial, é dada pela Equação (2.34),

$$\mathbf{u}_{j+1} = A\mathbf{u}_j + \mathbf{b}, \tag{2.34}$$

onde $\mathbf{u}_{j+1}$ é o vetor de temperaturas no instante de tempo $j+1$, $\mathbf{u}_j$ é o vetor de temperaturas no instante de tempo j, $\mathbf{b}$ é um vetor cujas componentes contém informações das condições de fronteira e A é a matriz dos coeficientes, conforme descrito a seguir.

$$A = \begin{pmatrix} 1 - \frac{2k}{\beta h^2} & \frac{2k}{\beta h^2} & 0 & \dots & 0 \\ 1 - \frac{2k}{\beta h^2} & \frac{k}{\beta h^2} & 0 & \dots & 0 \\ \frac{k}{\beta h^2} & 1 - \frac{2k}{\beta h^2} & \frac{k}{\beta h^2} & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & \dots & \frac{k}{\beta h^2} & 1 - \frac{2k}{h^2} & \frac{k}{\beta h^2} \\ 0 & \dots & 0 & \frac{k}{\beta h^2} & 1 - \frac{2k}{\beta h^2} \\ 0 & \dots & 0 & \frac{2k}{\beta h^2} & 1 - \frac{2k}{\beta h^2} \end{pmatrix} \tag{2.35}$$

$$\mathbf{u}_j = \begin{pmatrix} u_{0,j} \\ u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n-2,j} \\ u_{n-1,j} \\ u_{n,j} \end{pmatrix} \tag{2.36}$$

$$\mathbf{u}_{j+1} = \begin{pmatrix} u_{0,j+1} \\ u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{n-2,j+1} \\ u_{n-1,j+1} \\ u_{n,j+1} \end{pmatrix} \tag{2.37}$$

$$\mathbf{b} = \begin{pmatrix} \frac{2k}{\beta h \lambda}\phi(t) \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \tag{2.38}$$

Como no sistema descrito na Equação (2.34), o único vetor de incógnitas é $\mathbf{u}_j$+1 (temperaturas no instante j+1), para resolvê-lo, basta realizar o produto entre a matriz A e o vetor $\mathbf{u}_j$ e, o resultado, somar ao vetor $\mathbf{b}$.

Conforme já mencionado, as soluções obtidas pelo Método das Diferenças Finitas são aproximações para as soluções analíticas. Quando essas últimas existirem, haverá uma diferença nos valores obtidos ao se comparar a solução analítica e a solução numérica. Segundo Cuminato e Júnior [7], assumindo que são utilizadas as soluções exatas nos cálculos dos valores anteriores, essa diferença, que é ponderada por k, ou seja, $\triangle t$, é chamada de *erro de truncamento local*, denotado por $\tau_{i,j}$, o qual é dado pela Equação (2.39).

$$\tau_{i,j} = \frac{T_{i,j+1} - u_{i,j+1}}{k} \tag{2.39}$$

No entanto, se for considerado que nos cálculos dos valores anteriores não são utili-

zadas as soluções exatas, o erro é chamado de *erro de truncamento global*. Em erros de truncamento globais, como o próprio nome diz, podem incluir até erros de arredondamento do computador.

Assim, a Equação (2.29) é expressa conforme a Equação (2.40).

$$u_{i,j+1} = u_{i,j} + \frac{k}{\beta h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + \tau_{i,j} \tag{2.40}$$

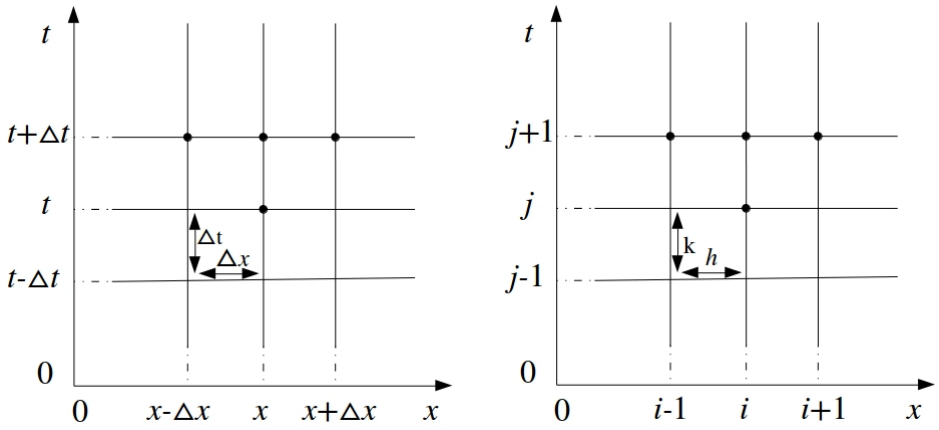
Cuja representação, na forma vetorial, incluindo o erro de truncamento, é dada da seguinte maneira:

$$\mathbf{u}_{j+1} = A\mathbf{u}_j + \mathbf{b} + \boldsymbol{\tau}. \tag{2.41}$$

2.3.3 Aplicação da Formulação Implícita ao Problema Proposto

Segundo Cuminato e Júnior [7], a abordagem implícita é aquela em que dois ou mais valores desconhecidos da grandeza de interesse são especificados, simultaneamente, em termos de valores já conhecidos em instâncias anteriores.

Ao considerar as Equações (2.12) à (2.15) do problema de transferência de calor, proposto por Carollo [4], na Figura 2.7 é apresentada a malha computacional para o domínio espaço-tempo utilizada na formulação implícita.



(a) Notação original. (b) Notação indicial.

Figura 2.7: Malha computacional utilizada no Método das Diferenças Finitas com formulação implícita, com a equivalência entre as notações.
Fonte: O Autor, 2019.

Considerando ainda $u_{i,j}$ uma aproximação de $T_{i,j}$ e, utilizando diferenças centradas do lado esquerdo da Equação (2.12) e diferenças atrasadas do lado direito obtém-se a Equação (2.42).

$$\frac{u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1}}{h^2} = \beta \frac{u_{i,j+1} - u_{i,j}}{k}$$

$$\frac{k}{\beta h^2}(u_{i-1,j+1} - 2u_{i,j+1} + u_{i+1,j+1}) = u_{i,j+1} - u_{i,j}$$

$$\frac{k}{\beta h^2}u_{i-1,j+1} - \frac{2k}{\beta h^2}u_{i,j+1} + u_{i,j} + \frac{k}{\beta h^2}u_{i+1,j+1} = u_{i,j+1} \tag{2.42}$$

Ou ainda:

$$\frac{k}{\beta h^2}u_{i-1,j+1} + u_{i,j} + \frac{k}{\beta h^2}u_{i+1,j+1} = (1 + \frac{2k}{\beta h^2})u_{i,j+1}$$

$$\frac{k}{\beta h^2}u_{i-1,j+1} - (1 + \frac{2k}{\beta h^2})u_{i,j+1} + \frac{k}{\beta h^2}u_{i+1,j+1} = -u_{i,j}. \tag{2.43}$$

Cabe ressaltar que, na Equação (2.43), apenas $u_{i,j}$ é um valor conhecido. Como já foi dito, a abordagem implícita possui um grau maior de dificuldade para resolvê-la. A dificuldade está justamente no fato de resolver o sistema que é gerado com base na Equação (2.43), juntamente com as condições de fronteira, tomando $i = 1, 2, ..., n-1$ e $j = 0, 1, ..., m-1$.

Pelas condições de contorno em $x = 0$, na formulação implícita, fazendo $i = 0$ e utilizando o nó fictício, obtém-se:

$$-\lambda \frac{\partial T(x,t)}{\partial x} = \phi(t)$$

$$\frac{\partial T(x,t)}{\partial x} = -\frac{\phi(t)}{\lambda}$$

$$\frac{u_{1,j+1} - u_{-1,j+1}}{2h} = -\frac{\phi(t)}{\lambda}.$$

Logo:

$$u_{-1,j+1} = u_{1,j+1} + \frac{2h}{\lambda}\phi(t). \tag{2.44}$$

Ao substituir a Equação (2.44) na Equação (2.43) para $i = 0$, obtém-se:

$$\frac{k}{\beta h^2}u_{-1,j+1} - (1 + \frac{2k}{\beta h^2})u_{0,j+1} + (\frac{k}{\beta h^2})u_{1,j+1} = -u_{0,j}$$

$$\frac{k}{\beta h^2}u_{1,j+1} - (1 + \frac{2k}{\beta h^2})u_{0,j+1} + \frac{k}{\beta h^2}u_{1,j+1} = -u_{0,j} - \frac{2k}{\beta\lambda h}\phi(t)$$

$$-(1 + \frac{2k}{\beta h^2})u_{0,j+1} + \frac{2k}{\beta h^2}u_{1,j+1} = -u_{0,j} - \frac{2k}{\beta\lambda h}\phi(t). \tag{2.45}$$

Pelas condições de contorno em $x = L$, na formulação implícita, ao fazer $i = n$ e utilizando novamente o nó fictício, obtém-se:

$$\frac{\partial T(x,t)}{\partial x} = 0$$

$$\frac{u_{n+1,j+1} - u_{n-1,j+1}}{2h} = 0.$$

Assim:

$$u_{n+1,j+1} = u_{n-1,j+1}. \tag{2.46}$$

Ao substituir a Equação (2.46) na Equação (2.43), na formulação implícita, com $i = n$, obtém-se:

$$\frac{k}{\beta h^2}u_{n-1,j+1} - (1 + \frac{2k}{\beta h^2})u_{n,j+1} + \frac{k}{\beta h^2}u_{n+1,j+1} = -u_{n,j}$$

$$\frac{k}{\beta h^2}u_{n-1,j+1} - (1 + \frac{2k}{\beta h^2})u_{n,j+1} + \frac{k}{\beta h^2}u_{n-1,j+1} = -u_{n,j}$$

$$\frac{2k}{\beta h^2}u_{n-1,j+1} - (1 + \frac{2k}{\beta h^2})u_{n,j+1} = -u_{n,j}. \tag{2.47}$$

A solução do problema, ao utilizar notação vetorial é dada por:

$$A\mathbf{u}_{j+1} = \mathbf{u}_j + \mathbf{b}, \tag{2.48}$$

onde $\mathbf{u}_{j+1}$ é o vetor de temperaturas a serem determinadas no instante de tempo $j+1$, $\mathbf{u}_j$ é o vetor de temperaturas conhecidas no instante de tempo j, $\mathbf{b}$ é um vetor de termos independentes que levam em consideração as condições de contorno e A é a matriz dos coeficientes, dados por:

$$A = \begin{pmatrix} -(1+\frac{2k}{\beta h^2}) & \frac{2k}{\beta h^2} & 0 & \dots & 0 \\ -(1+\frac{2k}{\beta h^2}) & \frac{k}{\beta h^2} & 0 & \dots & 0 \\ \frac{k}{\beta h^2} & -(1+\frac{2k}{\beta h^2}) & \frac{k}{\beta h^2} & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & \dots & \frac{k}{\beta h^2} & -(1+\frac{2k}{\beta h^2}) & \frac{k}{\beta h^2} \\ 0 & \dots & 0 & \frac{k}{\beta h^2} & -(1+\frac{2k}{\beta h^2}) \\ 0 & \dots & 0 & \frac{2k}{\beta h^2} & -(1+\frac{2k}{\beta h^2}) \end{pmatrix} \tag{2.49}$$

$$\mathbf{u}_j = \begin{pmatrix} -u_{0,j} \\ -u_{1,j} \\ -u_{2,j} \\ \vdots \\ -u_{n-2,j} \\ -u_{n-1,j} \\ -u_{n,j} \end{pmatrix} \tag{2.50}$$

$$\mathbf{u}_{j+1} = \begin{pmatrix} u_{0,j+1} \\ u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{n-2,j+1} \\ u_{n-1,j+1} \\ u_{n,j+1} \end{pmatrix} \tag{2.51}$$

$$\mathbf{b} = \begin{pmatrix} -\frac{2k}{\beta\lambda h}\phi(t) \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \tag{2.52}$$

Na formulação implícita a matriz resultante do sistema de equações gerado, descrita na Equação (2.48), é resolvida pelo método Gauss-Seidel. Este método é iterativo para resolução de sistemas de equações lineares e, segundo Ruggiero e Lopes [24], sendo $\boldsymbol{x}^0$ uma aproximação inicial, o processo iterativo consiste em calcular $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, ..., $\boldsymbol{x}^{(k)}$, ..., através do sistema dado pela Equação (2.53), até que um critério de parada estabelecido a *priori* seja satisfeito.

$$\begin{cases} x_1^{(k+1)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^k - a_{13}x_3^k - \cdots - a_{1n}x_n^k) \\ x_2^{(k+1)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^k - \cdots - a_{2n}x_n^k) \\ \vdots \qquad\quad \vdots \qquad \vdots \qquad\quad \vdots \qquad\quad \vdots \\ x_n^{(k+1)} = \frac{1}{a_{nn}}(b_n - a_{n1}x_1^{(k+1)} - a_{n2}x_2^{(k+1)} - \cdots - a_{n,n-1}x_{n-1}^{(k+1)}) \end{cases} \tag{2.53}$$

No processo iterativo de Gauss-Seidel, no momento de se calcular $x_j^{(k+1)}$ todos os valores obtidos anteriormente, $x_1^{(k+1)}, x_2^{(k+1)}, \ldots, x_{j-1}^{(k+1)}$, são utilizados, juntamente com os valores restantes $x_{j+1}^{(k)}$, ... , $x_n^{(k)}$. Nesse trabalho é adotado como critério de parada para o método Gauss-Seidel, o erro entre duas iterações sucessivas menor que 10^{-10}.

Quanto ao erro de truncamento da formulação implícita, a ideia é semelhante à formulação explícita, desta forma, considerando $\tau_{i,j}$ o erro de truncamento que ocorre no cálculo de $u_{i,j+1}$, obtém-se a Equação (2.54),

$$\frac{k}{\beta h^2}u_{i-1,j+1} - (1 + \frac{2k}{\beta h^2})u_{i,j+1} + \frac{k}{\beta h^2}u_{i+1,j+1} = -u_{i,j} + \tau_{i,j}, \tag{2.54}$$

que é a solução exata, pois a Equação (2.43) está acrescida do erro de truncamento.

Dessa maneira, a solução na forma vetorial para a formulação implícita, incluindo o erro de truncamento, é descrita conforme a Equação (2.55).

$$A\mathbf{u}_{j+1} = \mathbf{u}_j + \mathbf{b} + \boldsymbol{\tau} \tag{2.55}$$

2.3.4 Consistência, Estabilidade e Convergência

Em determinados casos, é necessário que o problema com domínio discretizado atenda a alguns critérios para que ocorra a convergência.

A discretização numérica pode introduzir alguma perturbação no método numérico, o que Cuminato e Júnior [7], definem como "ruído". Dessa forma, a convergência está

associada aos conceitos de consistência e estabilidade. Segundo Cuminato e Júnior [7], o erro cometido ao se substituir a solução exata na equação de diferenças é chamado de *erro de truncamento local*. Se o erro de trucamento local é $O(h^p)$, com $p \geq 1$, o método tem *ordem de consistência* p, ou simplesmente é dito que o método é de ordem p. Ainda existe um outro conceito, chamado de *zero-estabilidade* e ocorre quando as oscilações se "dissipam", não prejudicando a convergência [7].

Cuminato e Júnior [7] afirmam que quanto menor o valor de h, maior será o número de cálculos que deverão ser efetuados para que a solução se aproxime de um ponto x fixado, o que leva a um acúmulo de erros, tornando o método *instável* ou *não estável*. A *estabilidade* está associada ao tamanho de h da malha, onde é possível estabelecer o controle de propagação do erro. Formalmente, pode-se definir esses conceitos conforme a seguir:

Definição 9: Um problema é dito bem posto se ele tem única solução que depende continuamente dos dados iniciais ou de fronteira ou de ambos [20].

Definição 10 (Consistência): Para que uma discretização seja consistente com a Equação Diferencial Parcial, seu erro de truncamento local deve tender à zero quando $\triangle x$ e $\triangle t$ tenderem à zero, sendo assim o erro de truncamento local deve ser pelo menos $O(h)$ [9, 20].

Definição 11 (Estabilidade): Um método numérico estável é aquele no qual quaisquer erros ou perturbações na solução não são amplificados sem limites [9].

Definição 12 (Convergência - Teorema de Lax): Uma condição necessária e suficiente para convergência de um método, quando aplicado a um problema de valor inicial bem posto, é que o esquema de discretização seja consistente e estável [20].

No que se refere ao Método das Diferenças Finitas, a formulação explícita é condicionalmente convergente e, antes de abordar a formulação implícita, é fundamental esclarecer o que isso significa para futura compreensão e análise dos resultados dos algoritmos. Para isso, considere a difusividade térmica dada por $\alpha = \frac{\lambda}{\rho c_p}$ cuja unidade de medida é m^2/s. Com isso, a Equação (2.29) é reescrita conforme Equação (2.56).

$$u_{i,j+1} = \alpha \frac{k}{h^2}(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}) + u_{i,j} \tag{2.56}$$

Para determinar a estabilidade do método, segundo Cuminato e Júnior [7], é utilizado o critério de Neumann. De acordo com Pereira et al. [20], havendo solução da equação

diferencial através da formulação explícita, pelo critério de estabilidade de Neumann e, considerando a unidade imaginária $I = \sqrt{-1}$, obtem-se:

$$u_{i,j} = e^{\xi j} e^{I\beta i} = (e^{\xi})^j e^{I\beta i} = (e^{\xi})^j (cos(\beta i) + Isen(\beta i))$$
$$u_{i,j+1} = e^{\xi(j+1)} e^{I\beta i} = (e^{\xi})^{j+1} e^{I\beta i} = (e^{\xi})^{j+1} (cos(\beta i) + Isen(\beta i))$$
$$u_{i+1,j} = e^{\xi j} e^{I\beta(i+1)} = (e^{\xi})^j e^{I\beta(i+1)} = (e^{\xi})^j [cos(\beta(i+1)) + Isen(\beta(i+1))]$$
$$u_{i-1,j} = e^{\xi j} e^{I\beta(i-1)} = (e^{\xi})^j e^{I\beta(i-1)} = (e^{\xi})^j [cos(\beta(i-1)) + Isen(\beta(i-1))].$$

Ao substituir esses resultados na Equação (2.56) tem-se:

$$e^{\xi(j+1)} e^{I\beta i} = e^{\xi j} e^{I\beta i} + \alpha \frac{k}{h^2} (e^{\xi j} e^{I\beta(i-1)} - 2e^{\xi j} e^{I\beta i} + e^{\xi j} e^{I\beta(i+1)}).$$

Se aplicada as propriedades da potenciação, chega-se ao resultado da Equação (2.57).

$$e^{\xi} = 1 + \frac{k}{h^2} (e^{-I\beta} - 2 + e^{I\beta}) \tag{2.57}$$

Como,

$$e^{-I\beta} + e^{I\beta} = cos(-\beta) + Isen(-\beta) + cos\beta + Isen\beta = 2cos\beta, \tag{2.58}$$

caso seja substituído esse resultado na Equação (2.57), chega-se à Equação (2.59).

$$e^{\xi} = 1 + \frac{k}{h^2} (-2 + 2cos\beta) \tag{2.59}$$

Ao tomar $|e^{\xi}| \leq 1$, por propriedade de módulo, tem-se $-1 \leq e^{\xi} \leq 1 \implies -1 \leq e^{\xi}$. Assim $1 + \frac{k}{h^2}(-2 + 2cos\beta) \geq -1 \implies \frac{k}{h^2}(-2 + 2cos\beta) \geq -2 \implies \frac{k}{h^2} \leq \frac{2}{(2-2cos\beta)} \implies \frac{k}{h^2} \leq \frac{1}{(1-cos\beta)}$, logo obtém-se a relação dada pela Equação (2.60).

$$r = \alpha \frac{k}{h^2} \leq \frac{1}{1 - cos\beta} \tag{2.60}$$

Como $-1 \leq cos\beta \leq 1$, o caso em que $cos\beta = 1$ não faz sentido analisar, pois o denominador de r será zero, porém, para um valor muito próximo 1, o denominador de r tende a zero, logo r tende ao infinito, e assim tem-se r menor do que infinito. Para $cos\beta = -1$, $r \leq \frac{1}{2}$, que também é menor do que o infinito, implicando em $\alpha \frac{k}{h^2} \leq \frac{1}{2}$, que depende do tamanho de k. Caso a Equação (2.60) não seja satisfeita, o método explícito

acumula os erros dos resultados anteriores, fazendo com que a solução numérica não se aproxime da solução analítica, ou seja, o método explícito não será estável.

2.4 Resultados do Problema Direto de Transferência de Calor

Nesta seção são apresentados os resultados obtidos pelo problema direto utilizando as formulações explícita e implícita, assim como diferentes configurações para a malha computacional.

2.4.1 Análise da Variação das Malhas Espacial e Temporal

Para resolver o problema direto de transferência de calor proposto neste trabalho, os valores da condutividade térmica e capacidade de calor volumétrica, são configurados como: λ=14,611 W/mK e $\rho c_p = 3,9073 \times 10^6\ m/s$. Já o comprimento da placa foi $L = 10,88 \times 10^{-3}\ m$, a temperatura inicial $T_0 = 18,84\ ^oC$ e o tempo final $t_f = 160\ s$. Os referidos valores foram retirados de Carollo [4].

O problema direto foi resolvido dividindo a malha espacial em 5, 10 e 15 nós, ou seja, $\triangle x = 0,00272$, $\triangle x = 0,001209$ e $\triangle x = 0,000777$, respectivamente. No trabalho de Carollo [4] foi utilizada uma malha temporal com 1600 nós. No entanto, com o intuito de verificar a possibilidade de adotar uma quantidade menor de nós, o que possibilita uma redução no tempo computacional, foram testadas, para cada malha espacial, divisões em 16, 32, 80, 160, 320, 800 e 1600 nós na malha temporal. Em cada um dos resultados também é mostrado o critério de estabilidade descrito por Neumann ($r \leq \frac{1}{2}$), de acordo com cada configuração de malha adotada.

Para a malha espacial com 5 nós, $\triangle x = 0,00272$, obteve-se os resultados apresentados na Tabela 2.1 referentes ao número de nós da malha temporal, bem como o critério de estabilidade r.

Na sequência, na malha espacial com 10 nós, $\triangle x = 0,001209$, obteve-se os resultados apresentados na Tabela 2.2, onde r é o critério de estabilidade para o modo explícito.

Tabela 2.1: Discretizações obtidas tomando como base $\triangle x = 0,00272$ utilizando a formulação explícita.

Tempo total do experimento (s)	$\triangle t$ (s)	Nós (-)	r (-)
160	0,1	1600	0,050544
160	0,2	800	0,101087
160	0,5	320	0,252718
160	1	160	0,505435
160	2	80	1,01087
160	5	32	2,527175
160	10	16	5,054351

Tabela 2.2: Discretizações obtidas tomando como base $\triangle x = 0,001209$ utilizando a formulação explícita.

Tempo total do experimento (s)	$\triangle t$ (s)	Nós (-)	r (-)
160	0,1	1600	0,255877
160	0,2	800	0,511753
160	0,5	320	1,279383
160	1	160	2,558765
160	2	80	5,11753
160	5	32	12,79383
160	10	16	25,58765

Por fim, para a malha espacial com 15 nós, $\triangle x = 0,000777$, obteve-se a seguinte configuração para a malha temporal e o respectivo critério de estabilidade, apresentados na Tabela 2.3.

Tabela 2.3: Discretizações obtidas tomando como base $\triangle x = 0,000777$ utilizando a formulação explícita.

Tempo total do experimento (s)	$\triangle t$ (s)	Nós (-)	r (-)
160	0,1	1600	0,619158
160	0,2	800	1,238316
160	0,5	320	3,09579
160	1	160	6,19158
160	2	80	12,38316
160	5	32	30,9579
160	10	16	61,9158

Os resultados numéricos obtidos pelos algoritmos, desenvolvidos na linguagem de programação C e gráficos contendo o perfil das temperaturas gerados usando a ferramenta

Gnuplot em um ambiente GNU/Linux, tendo como base as formulações explícita e implícita, foram satisfatórios para ambos os casos, ou seja, a temperatura no final da placa, obtida ao longo do tempo pelo método numérico, foi compatível com a temperatura experimental, quando respeitado o critério de estabilidade do método explícito, uma vez que o mesmo é condicionalmente estável $\left(\alpha\frac{k}{h^2} \leq \frac{1}{2}\right)$.

Na Figura 2.8 são mostrados os resultados para a formulação explícita com malha espacial contendo 5 nós, ou seja, $\triangle x = 0,00272$. Cabe ressaltar, de acordo com a Tabela 2.1, que a formulação explícita teve a estabilidade atendida apenas para o número de nós temporais iguais a 1600, 800 e 320, ou seja, $\triangle t$ igual a 0,1, 0,2 e 0,5, respectivamente. Para malha temporal com 160 nós, $\triangle t$ igual a 1, ainda é possível visualizar os resultados no gráfico por ser um valor muito próximo de $\frac{1}{2}$. Já para 80, 32 e 16 nós, que equivalem à $\triangle t$ igual a 2, 5 e 10, respectivamente, não é possível realizar a sua representação gráfica, devido ao alto grau de instabilidade.

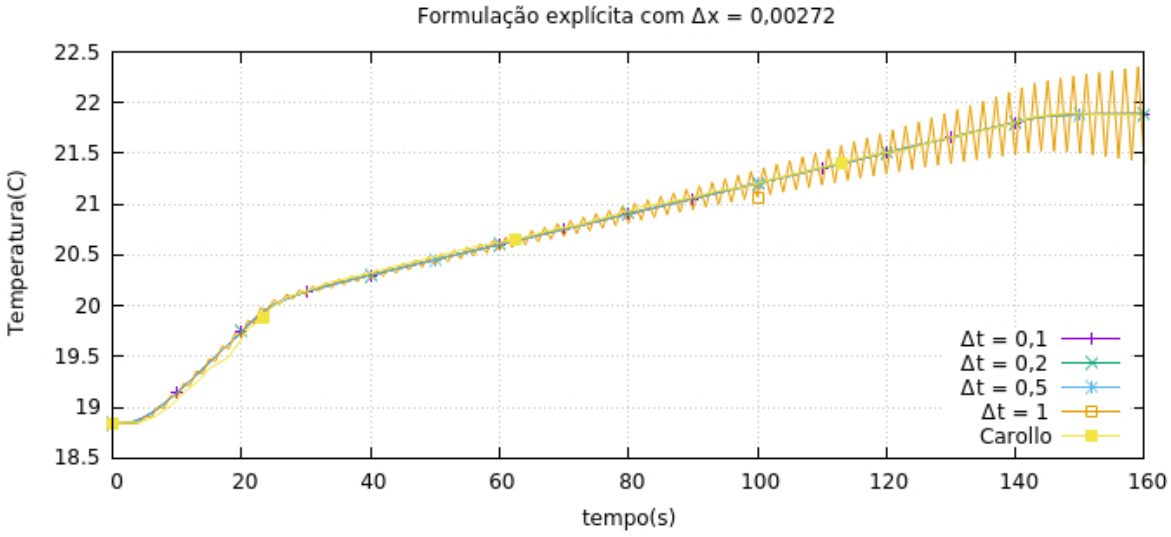


Figura 2.8: Resultados obtidos com a formulação explícita com $\triangle x = 0,00272$.
Fonte: O Autor, 2019.

Posteriormente, Figura 2.9, são apresentados os resultados obtidos com a formulação explícita utilizando a malha espacial com 10 nós, $\triangle x = 0,001209$. Como esperado, devido aos valores apresentados na Tabela 2.2, o método só é estável para malha temporal com 1600 nós, $\triangle t$ igual a 0,1. Nesse caso, o resultado é satisfatório em sua totalidade ao se comparar com os dados obtidos experimentalmente por Carollo [4]. Para os demais números de nós, devido à alta instabilidade, os resultados dos perfis das temperaturas foram omitidos.

Por fim, na formulação explícita, para $\triangle x = 0,000777$, os gráficos dos perfis das tem-

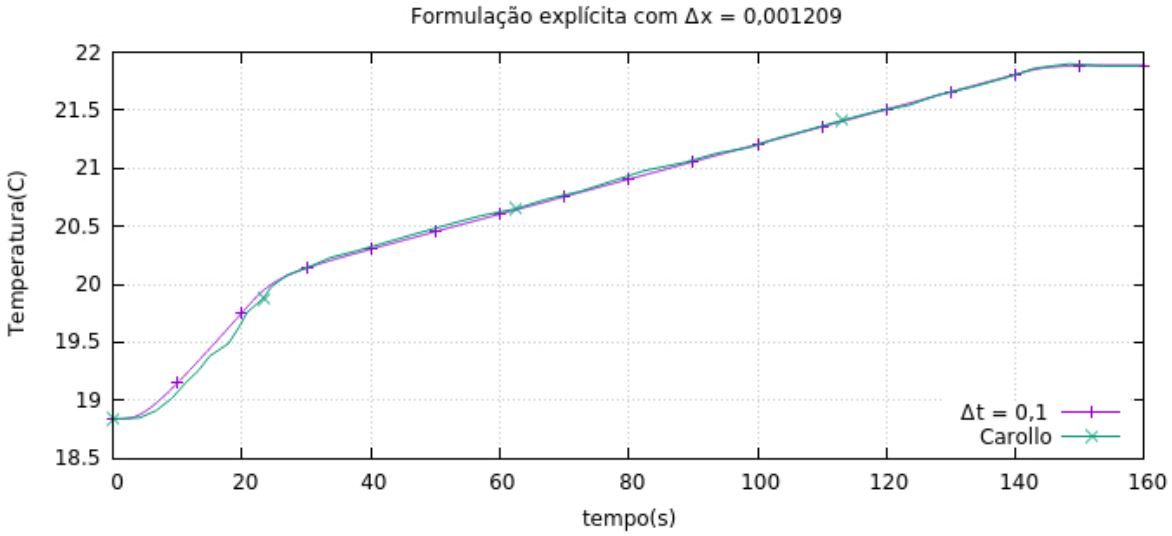


Figura 2.9: Resultados obtidos com a formulação explícita com $\triangle x = 0,001209$.
Fonte: O Autor, 2019.

peraturas mostradas na Figura 2.10 apresentam coerência com os resultados da Tabela 2.3, logo o método é instável para todas as malhas temporais escolhidas devido ao grande refinamento da malha. Por razão do alto grau de instabilidade, a escala da temperatura fica muito desproporcional para comportar os resultados de todas as malhas temporais, criando a falsa sensação do perfil das temperaturas comportar-se como uma reta para a maioria dos casos.

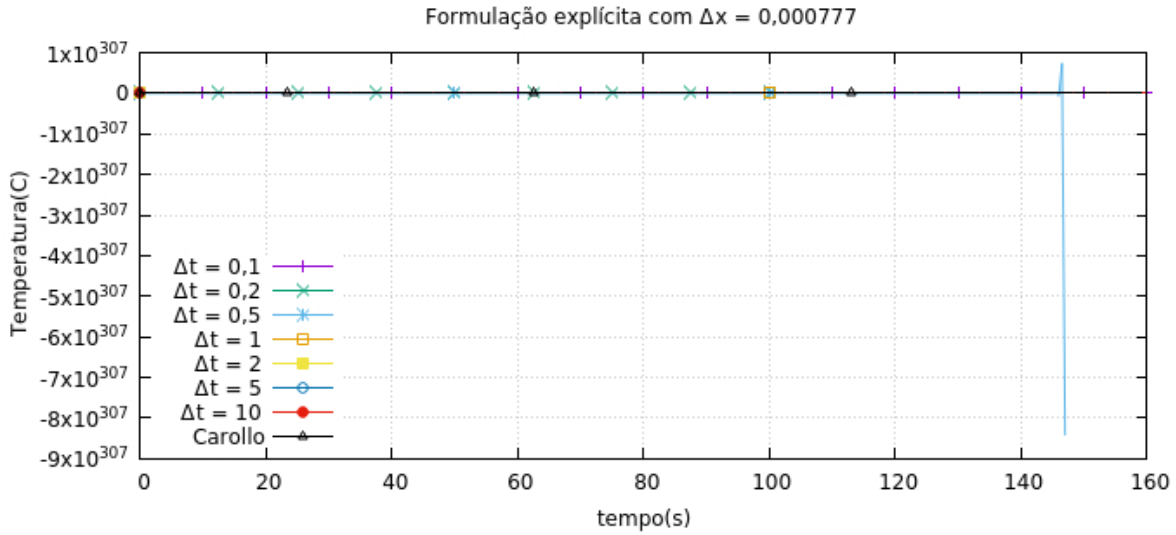


Figura 2.10: Resultados obtidos com a formulação explícita com $\triangle x = 0,000777$.
Fonte: O Autor, 2019.

Como a formulação implícita é incondicionalmente estável, a última coluna das Ta-

belas 2.1, 2.2 e 2.3, não interfere nos resultados obtidos pelo algoritmo no que se refere à estabilidade do método.

Para malha espacial contendo 5 nós, nos resultados apresentados na Figura 2.11, as malhas temporais com 16 e 32 nós, que equivalem a $\triangle t$ igual 10 e 5, apresentam erro maior, mas sem as oscilações espúrias típicas da instabilidade. Entretanto, as outras malhas possibilitam reproduzir um perfil de temperatura compatível com o experimento de Carollo [4].

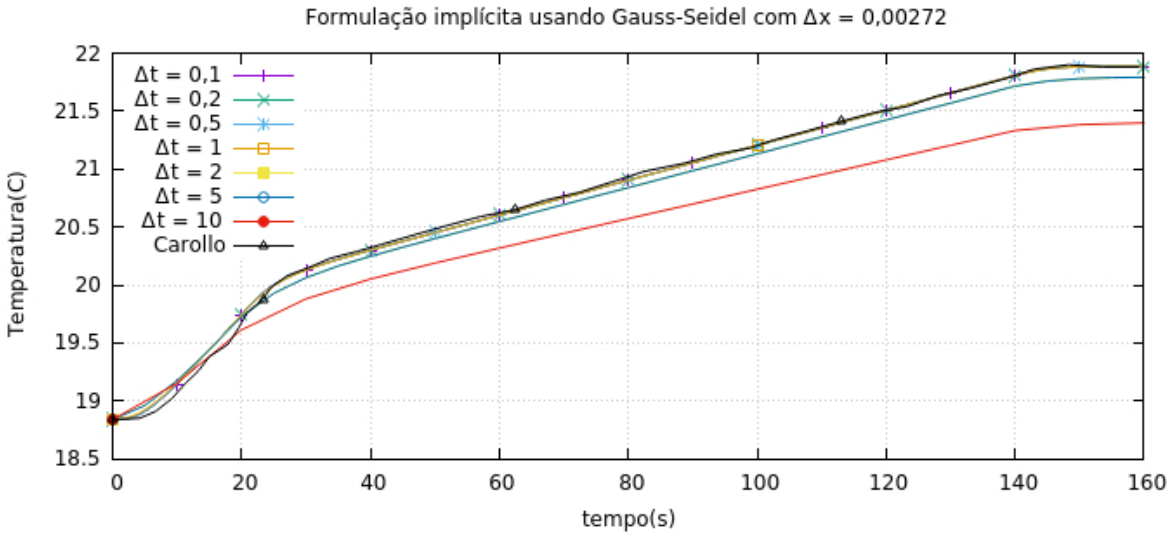


Figura 2.11: Resultados obtidos com a formulação implícita com $\triangle x = 0,00272$.
Fonte: O Autor, 2019.

Por outro lado, quando adotado 10 nós para a malha espacial, $\triangle x = 0,001209$, os resultados são satisfatórios para as malhas temporais com 320, 800 e 1600 nós relacionadas a $\triangle t$ igual a 0,5, 0,2 e 0,1, respectivamente, como o perfil mostrado na Figura 2.12. Assim como a malha temporal com 16 e 32 nós, $\triangle t = 10$ e $\triangle t = 5$, respectivamente (Figura 2.11), as malhas temporais com 80 e 160 nós, $\triangle t = 2$ e $\triangle t = 1$, esta última em menor escala, também apresentam valores distantes daqueles obtidos no experimento de Carollo [4]. Para $\triangle t = 5$ e $\triangle t = 10$ o método apresentou os piores resultados.

Finalizando, para $\triangle x = 0,000777$, apenas as malhas temporais contendo 800 e 1600 nós, $\triangle t$ igual a 0,2 e 0,1, obtiveram resultado numérico para as temperaturas próximos ao experimento de Carollo [4], conforme Figura 2.13. As demais malhas não reproduziram um perfil de temperatura próximo ao resultado experimental, não obstante, a malha temporal com 320 nós, $\triangle t = 0,5$, foi a que mais se aproximou, mas para efeitos de resultados não pode ser considerada como uma boa discretização para o problema em questão.

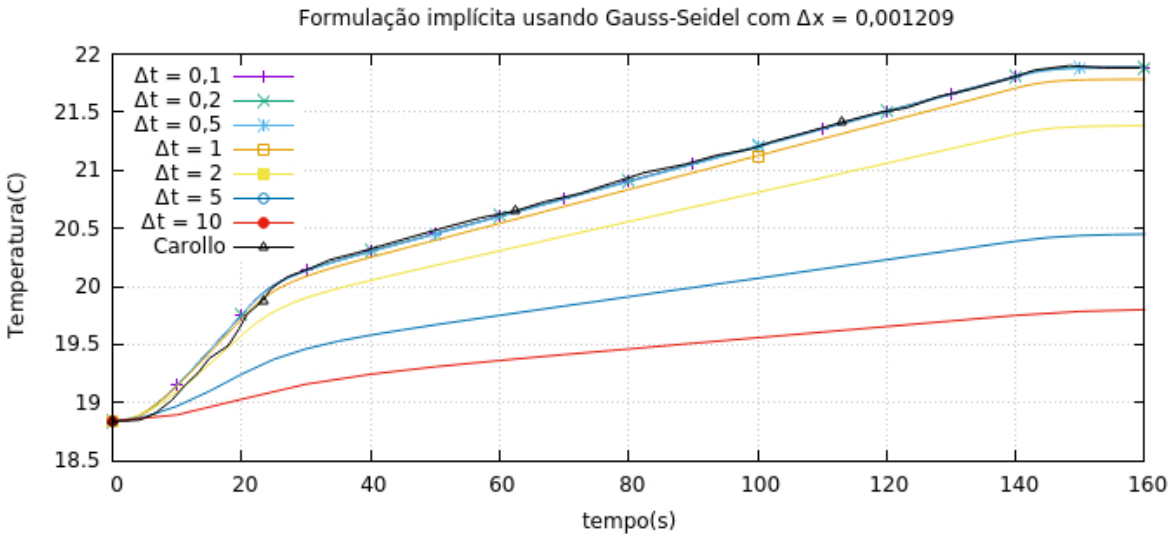


Figura 2.12: Resultados obtidos com a formulação implícita com $\triangle x = 0,001209$.
Fonte: O Autor, 2019.

Cabe ressaltar, porém, que apesar de algumas configurações adotadas para as malhas espacial e temporal na formulação implícita não apresentarem um perfil de temperatura próximo aos dados experimentais de Carollo [4], em nenhum dos casos, ocorreu instabilidade no método, como na formulação explícita.

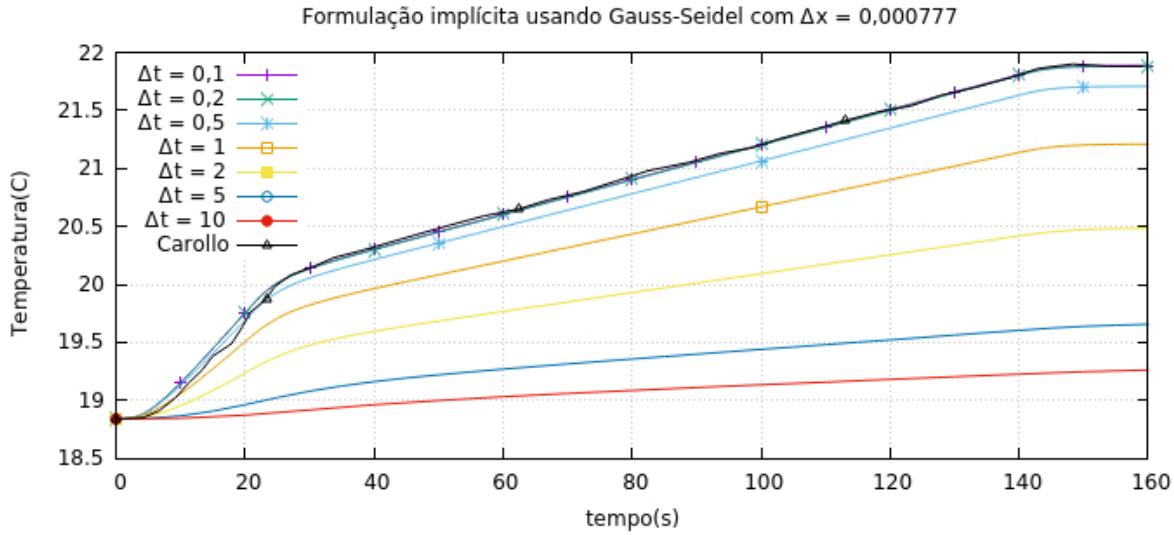


Figura 2.13: Resultados obtidos com a formulação implícita com $\triangle x = 0,000777$.
Fonte: O Autor, 2019.

Após a avaliação do perfil das temperaturas no ponto de localização do termopar (superfície inferior da placa), utilizando as formulações explícita e implícita e a diversas discretizações envolvendo os domínios espacial e temporal, também foi analisado o perfil

da temperatura ao longo de todo o domínio espacial $x \in [0, L]$, com ambas formulações.

No que tange ao referido perfil ao longo do domínio, os gráficos contendo os perfis das temperaturas presentes nas Figuras 2.14 e 2.15 mostram um resultado compatível e satisfatório com o experimento, onde o mesmo foi obtido considerando $\triangle x = 0,001209$ e $\triangle t = 0,1$, ou seja, malha espacial com 10 nós e malha temporal com 1600 nós, em intervalos de 20 s, tanto para a formulação explícita quanto para a formulação implícita.

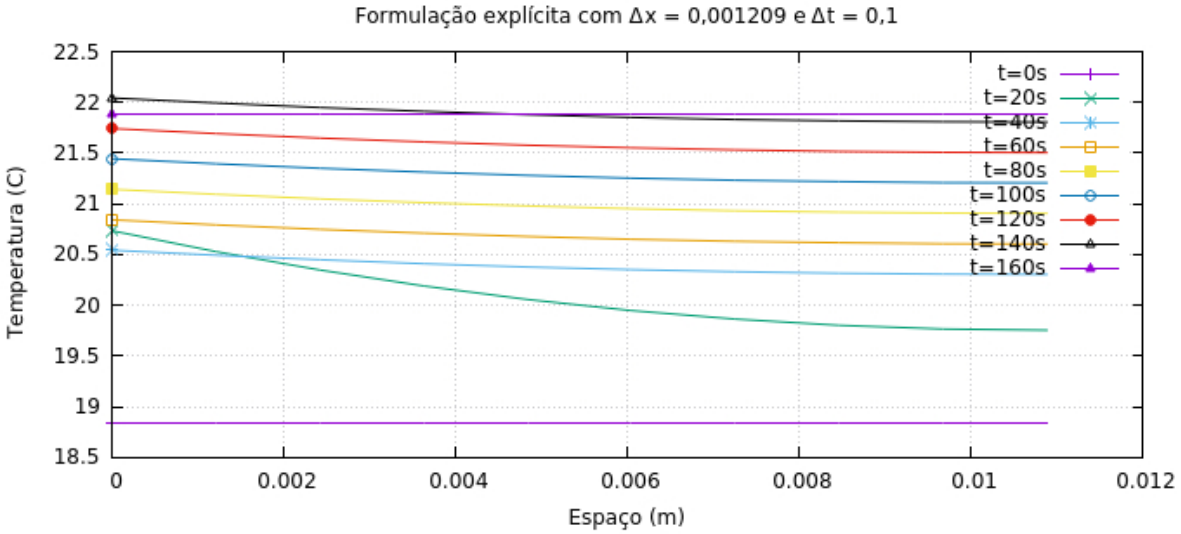


Figura 2.14: Resultado para o perfil da temperatura em todo o domínio espacial $[0, L]$ obtido pela formulação explícita, onde $\triangle x = 0,001209$ e $\triangle t = 0,1$.
Fonte: O Autor, 2019.

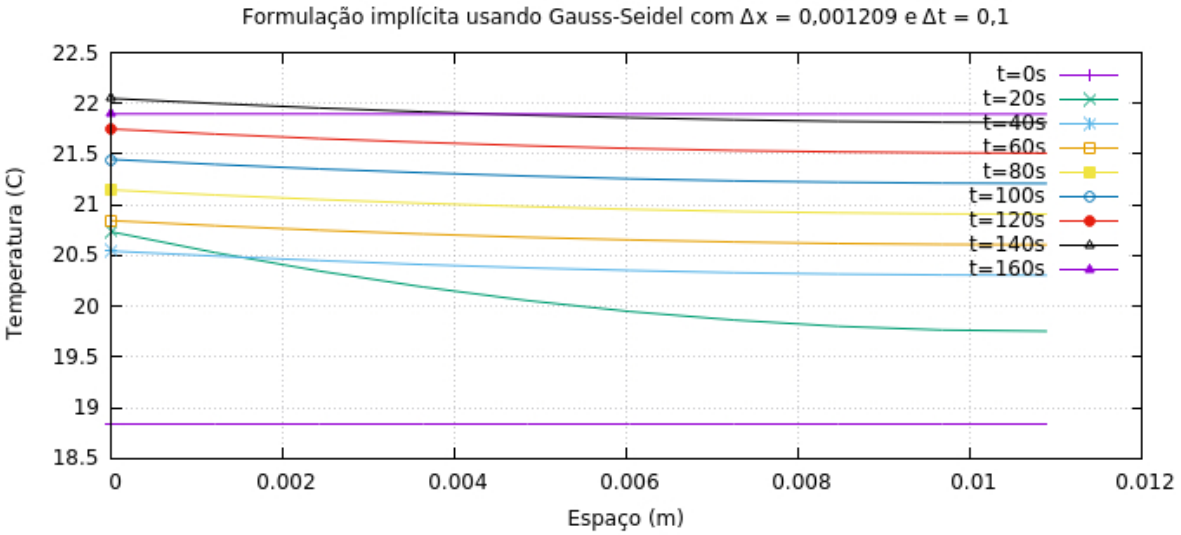


Figura 2.15: Resultado para o perfil da temperatura em todo o domínio espacial $[0, L]$ obtido pela formulação implícita, onde $\triangle x = 0,001209$ e $\triangle t = 0,1$.
Fonte: O Autor, 2019.

No tempo inicial, quando $t = 0\ s$, a temperatura é a mesma, ou seja, constante, podendo-se observar, em ambos os gráficos, uma reta. Nos outros instantes de tempo, a temperatura começa alta, devido ao termostato, fluxo de calor na superfície em $x = 0$, porém diminui com o passar do tempo, até que no tempo $t = 160\ s$, o gráfico mostra que a temperatura na placa obteve o estado de equilíbrio conforme os dados experimentais. É importante observar que no tempo $t = 160\ s$, o gráfico aparenta ser uma reta. Isso ocorre porque a temperatura final no último segundo do experimento se dá por diferença de valor a partir da quarta casa decimal, sendo assim, para se observar uma alteração, a escala da temperatura deveria ser maior, em outras palavras, o sistema entrou em equilíbrio.

Por outro lado, ao considerar $\triangle x = 0,001209$ e $\triangle t = 0,2$ para a formulação explícita, conforme representado na Figura 2.16, é possível verificar a oscilação nos resultados, uma vez que a quantidade de nós interfere nos resultados devido ao critério de estabilidade de Neumann. Para $\triangle x = 0,001209$ e $\triangle t = 0,5$ (incluindo os demais valores) o perfil das temperatura segue o mesmo comportamento (resultados não apresentados aqui).

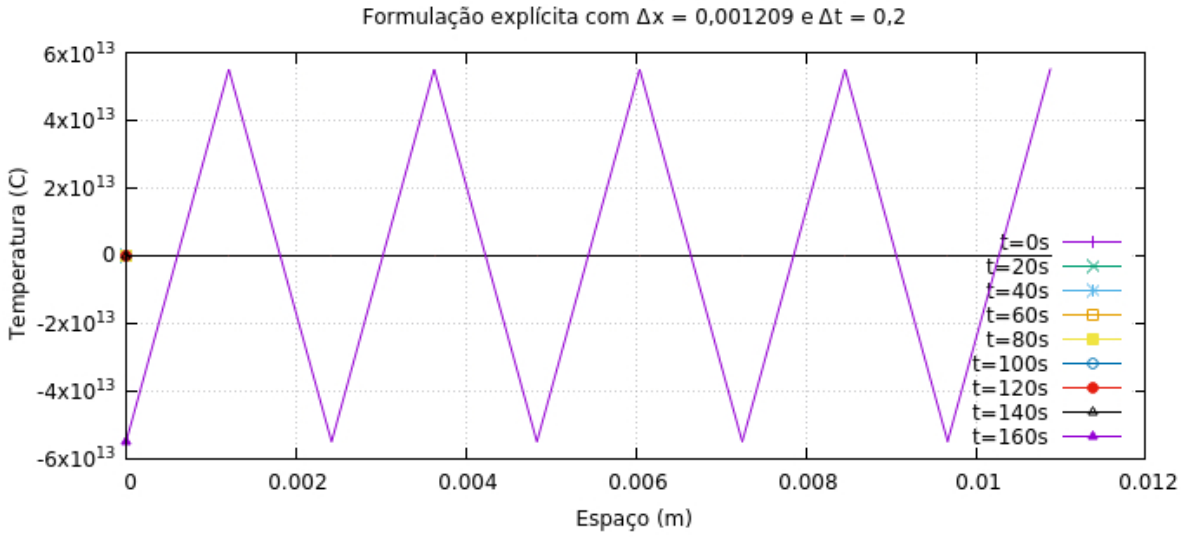


Figura 2.16: Resultado para o perfil da temperatura em todo o domínio espacial $[0, L]$ obtido pela formulação explícita, onde $\triangle x = 0,001209$ e $\triangle t = 0,2$.
Fonte: O Autor, 2019.

Por fim, nas Figuras 2.17 e 2.18 é possível visualizar a convergência do Método das Diferenças Finitas na formulação implícita em instantes de 20 em 20 s, adotando-se $\triangle x$ igual a $0,001209$ e $\triangle t$ igual a 0,2 e 0,5, respectivamente.

Em resumo, para a formulação explícita com malha espacial de 5 nós ($\triangle x = 0,00272$), o método se apresentou estável apenas para $\triangle t$ igual a 0,1, 0,2 e 0,5. Para malha espacial de 10 nós ($\triangle x = 0,001209$), o método se apresentou estável apenas para $\triangle t = 0,1$. Já

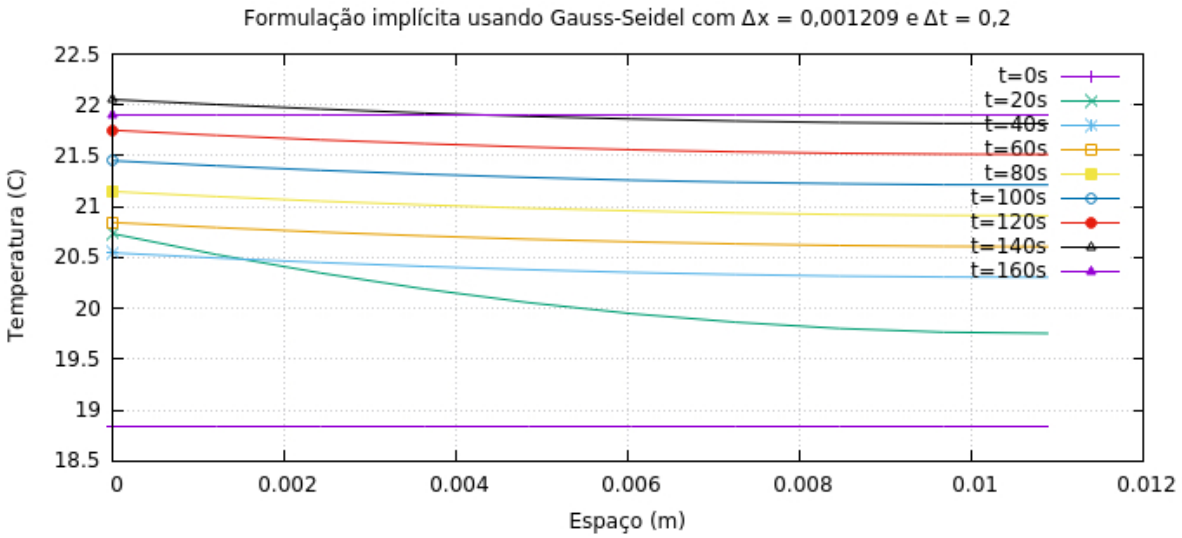


Figura 2.17: Resultado para o perfil da temperatura em todo o domínio espacial $[0, L]$ obtido pela formulação implícita, onde $\triangle x = 0,001209$ e $\triangle t = 0,2$.
Fonte: O Autor, 2019.

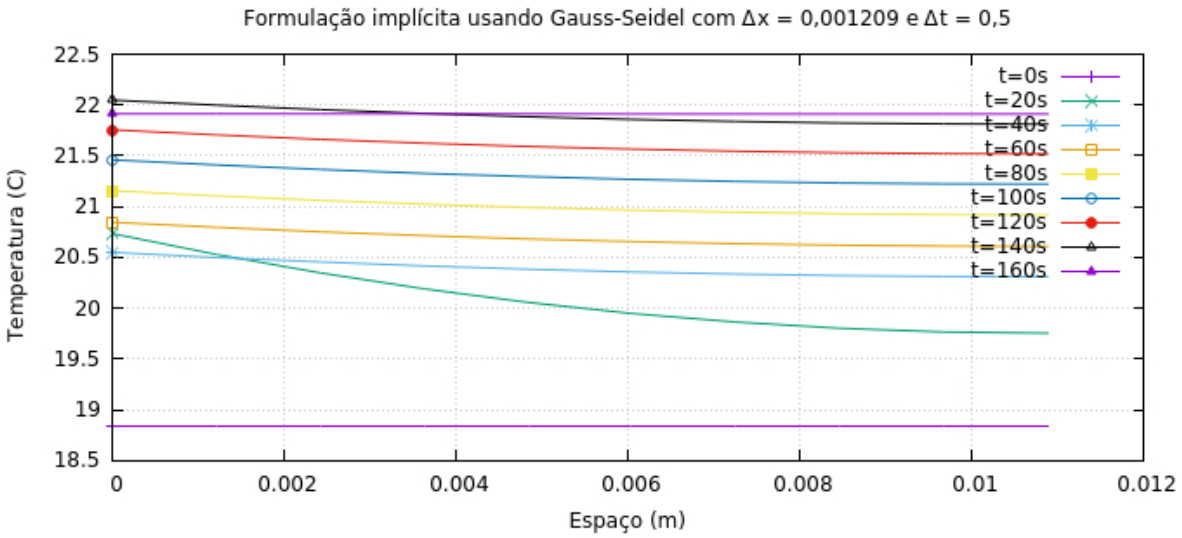


Figura 2.18: Resultado para o perfil da temperatura em todo o domínio espacial $[0, L]$ obtido pela formulação implícita, onde $\triangle x = 0,001209$ e $\triangle t = 0,5$.
Fonte: O Autor, 2019.

para a malha espacial com 15 nós ($\triangle x = 0,000777$), ainda se tratando da formulação explícita, o método se tornou instável para todas as configurações da malha temporal. No que se refere à formulação implícita, embora incondicionalmente estável, não foi possível encontrar solução compatível com o experimento realizado por Carollo [4], para $\triangle x = 0,00272$ com $\triangle t$ igual a 10, $\triangle x = 0,001209$ com $\triangle t$ iguais a 2, 5 e 10 e, para $\triangle x = 0,000777$ com $\triangle t$ iguais a 1, 2, 5 e 10.

Diante do exposto, verifica-se que, apesar da malha espacial com 10 nós e malha temporal com 1600 nós gerar resultados muito próximos àqueles obtidos experimentalmente por Carollo [4], tanto na formulação explícita como na formulação implícita, a malha com 800 nós na formulação implícita, ou seja, $\triangle x = 0,001209$ e $\triangle t = 0,2$, também obtém resultados satisfatórios, com custo computacional menor, conforme Figura 2.11. A Figura 2.19 apresenta o comportamento do erro absoluto obtido entre os dados experimentais (Carollo [4]) e os resultados numéricos das temperaturas obtidas com a formulação implícita, adotando $\triangle x = 0,01209$ e $\triangle t = 0,2$.

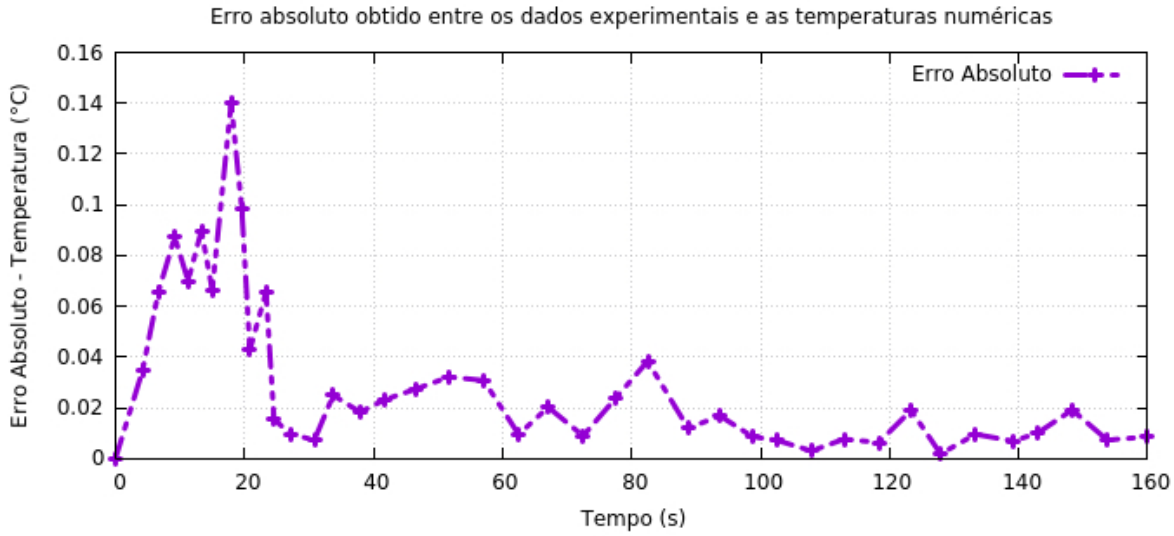


Figura 2.19: Erro absoluto obtido entre os dados experimentais e temperaturas obtidas com a formulação implícita, adotando $\triangle x = 0,01209$ e $\triangle t = 0,2$.
Fonte: O Autor, 2019.

Capítulo 3

Formulação do Problema Inverso

De maneira geral, na análise de um problema sob a ótica da abordagem direta, se conhece os parâmetros de entrada (causa), bem como o modelo matemático que representa o problema e, o que se deseja obter é o efeito, ou seja, a informação desconhecida, conforme ilustrado na Figura 3.1.

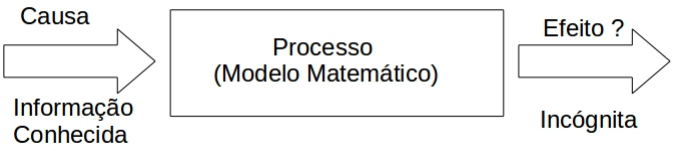


Figura 3.1: Formulação básica de um Problema Direto.
Fonte: Adaptado de Silva Neto, Becceneri e Campus Velho [29].

Agora imagine um problema em que se deseja analisar as causas (informação desconhecida), mas não se tem os respectivos parâmetros, o que se conhece são os efeitos. Esse tipo de problema é conhecido como problema inverso e é representado de maneira ilustrativa na Figura 3.2.

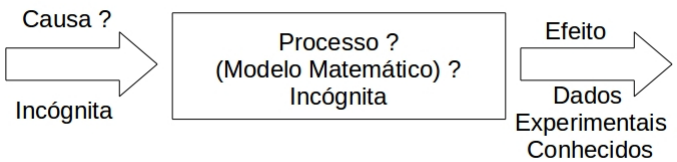


Figura 3.2: Formulação básica de um Problema Inverso.
Fonte: Adaptado de Silva Neto, Becceneri e Campus Velho [29].

No caso do problema inverso, se conhece o resultado final (dados experimentais conhecidos) que, no problema de transferência de calor abordado nesse trabalho, são as temperaturas, obtidas ao longo do tempo, através de um termopar localizado na superfície inferior ($x = L$) da placa de acordo com o experimento realizado por Carollo [4], o qual foi descrito na Seção 2.2.

Segundo Silva Neto, Becceneri e Campus Velho [29], existem diversas classificações para os problemas inversos e os mesmos podem ser divididos em dois grupos: estimativa de parâmetros e estimativa de funções. Quando a função objetivo é conhecida, o que se busca é estimar os parâmetros iniciais, enquanto estima-se a função quando os parâmetros iniciais já são conhecidos. No presente trabalho, a função objetivo já é conhecida, logo o que se deseja é estimar a condutividade térmica (λ) e a capacidade volumétrica (ρc_p) presentes no modelo matemático descrito pela Equação (2.12), utilizando dados experimentais de temperatura para essa finalidade.

Ao resolver um problema, nem sempre a solução encontrada é a melhor, sendo preciso, às vezes, lançar mão de técnicas para melhorá-la. A maioria dos problemas são representados por meio de uma função conhecida como função objetivo e possuem restrições na solução. Segundo Arenales et al. [1], resolver um problema de otimização consiste em determinar uma solução ótima, isto é, determinar uma solução que satisfaça todas as restrições de problema e atribua o melhor valor à função objetivo.

Uma função objetivo pode vir acompanhada de uma ou mais restrições e caso o problema possua solução, ou seja, ser for possível obter solução que satisfaça todas as restrições, então o problema possui solução factível ou viável, porém, se ao menos uma restrição imposta não puder ser satisfeita, denomina-se tal solução como infactível ou inviável. De acordo com Arenales et al. [1], uma solução é dita factível se satisfizer todas as restrições e condições de não negatividade do problema abordado. Muitos problemas buscam encontrar a solução ótima, que por sua vez é a melhor de todas as soluções.

Sob a ótica dos métodos desenvolvidos para formulação e solução de problemas inversos, pode-se dividi-los conforme duas abordagens: métodos inversos puros e métodos baseados em otimização [29]. Otimizar um problema, segundo Silva Neto, Becceneri e Campos Velho [29], é encontrar o valor máximo ou mínimo da função objetivo, quando for possível. Dentre os vários problemas em que se busca uma solução ótima, pode-se agrupá-los em duas grandes classes: problemas lineares e não lineares. Um problema é dito linear se todas as variáveis, incluindo as variáveis das restrições, possuem grau 1. Caso o grau seja maior do que 1 (2, 3, ..., n), então o problema é não-linear.

Um exemplo de problema linear seria encontrar a solução para $x + y = 0$ com a restrição $x < 0$ e $y > 0$. Esse problema possui infinitas soluções, uma delas seria $x = -1$ e $y = 1$. Essa solução é factível ou viável, no entanto, se as restrições fossem $x < 0$ e $y < 0$ ou até mesmo $x > 0$ e $y > 0$ a solução do mesmo problema seria infactível ou inviável, uma vez que não seria possível somar dois números positivos ou dois números negativos cujo resultado fosse zero. Um exemplo de problema não-linear seria $x^2 + y^2 \geq 1$ com restrição $x \geq 0$ e $y \geq 0$. Todos esses, são exemplos de problemas bidimensionais por envolver duas variáveis. Quanto mais variáveis e restrições, mais complexo o problema, daí a necessidade do uso de computadores para solucionar problemas.

Em se tratando de otimização, existem os métodos que podem ser classificados em determinístico, estocástico e híbrido. Nos métodos determinísticos, segundo Telles [30], busca-se parâmetros a partir de uma estimativa inicial, tendo como base informações do gradiente da função objetivo, o qual, determina sua direção até a solução do problema. Como característica positiva, essa classe de métodos leva a uma rápida convergência, porém não há garantia de que o valor encontrado é o mínimo ou máximo da função.

Já os métodos estocásticos possuem características aleatórias, fornecem uma garantia probabilística de que o mínimo ou máximo da função, dependendo do que se pretende alcançar, seja obtido, conforme afirmado por Telles [30]. Nos métodos estocásticos são aplicados procedimentos que envolvem a geração de números randômicos em um domínio de busca, enquanto que nos métodos determinísticos, sempre que se partir do mesmo ponto inicial, chega-se a mesma solução.

Por fim, os métodos híbridos são definidos quando se combina mais de um método no problema. Quando boas estimativas são geradas pelo método estocástico, o método determinístico pode sair de mínimos ou máximos locais e encontrar uma solução melhor. A hibridização entre métodos estocásticos também pode melhorar a solução de um problema.

Neste trabalho o intuito do problema inverso é minimizar a função objetivo dada pela diferença entre as temperaturas experimentais ($\boldsymbol{T_{exp}}$), obtidas ao longo do tempo por meio de um termopar localizado na superfície inferior da placa de espessura L [4], e numéricas ($\boldsymbol{T_{num}}(\lambda, \rho c_p)$), onde essa última é obtida através da equação do calor, Equação (2.12), juntamente com suas condições de contorno e iniciais, Equações (2.13) a (2.15), cuja solução é dada pelo Método das Diferenças Finitas com formulação implícita, gerando um sistema de equações descrito pela Equação (2.48), o qual é resolvido pelo métdodo Gauss-Seidel.

A função objetivo é dada pela Equação (3.1):

$$f(\lambda, \rho c_p) = |\boldsymbol{T_{exp}} - \boldsymbol{T_{num}}(\lambda, \rho c_p)|, \tag{3.1}$$

a qual também pode ser expressada como:

$$f(\lambda, \rho c_p) = \sqrt{\sum_{i}^{n} (T_{exp_i} - T_{num_i}(\lambda, \rho c_p))^2}, \tag{3.2}$$

onde T_{exp_i} e T_{num_i} são, respectivamente, as temperaturas experimentais e numéricas, obtidas em cada instante de tempo.

Para a estimativa das componentes do vetor de incógnitas do problema, $\mathbf{Z} = (\lambda, \rho c_p)$, as quais estão presentes na Equação (2.12), são utilizados métodos de otimização, mais especificamente, os métodos Luus-Jaakola e Algoritmo de Colisão de Partículas, bem como algumas modificações e hibridizações nos mesmos.

3.1 Método Luus-Jaakola (LJ)

O Metódo Luus-Jaakola (LJ) é um procedimento estocástico de busca aleatória, onde o domínio de busca inicial é o domínio de cada variável do problema e, a cada iteração, o tamanho do espaço de busca é reduzido [15].

Nesse método a principal ideia é considerar uma região de busca ampla, englobando possíveis valores das variáveis, gerando soluções aleatórias, enquanto a região de busca vai diminuindo cada vez mais ao longo das iterações [30].

Inicialmente deve-se determinar os intervalos de busca para as variáveis que se deseja estimar, os quais dependem do problema a ser tratado, bem como a amplitude dos referidos intervalos que, no algoritmo LJ, é denotado por $\boldsymbol{r}^{(k)} = (r_1^{(k)}, \ldots, r_n^{(k)})$ onde n é o número de variáveis, ou seja, o número da dimensão do problema.

A redução do intervalo de busca é dada por um fator de contração, ϵ, que está relacionado com a porcentagem em que o intervalo será contraído e, o *loop* externo do algoritmo, n_{out}, o qual determina quantas vezes essa contração irá acontecer. Em termos mais técnicos, na versão clássica do algoritmo, esse fator de contração pode ser considerado uma constante, uma vez que é definido no início da execução do algoritmo e depois não se altera. Por fim, o número de estimativas geradas a cada *loop* é dada pelo número de *loops* internos, n_{int}.

Após configurar todos os parâmetros iniciais, executa-se o algoritmo e o mesmo gera

uma solução inicial, $\boldsymbol{x}_0^{(0)}$, tida como melhor solução para o problema até o momento, ou seja, $\boldsymbol{x}^*= \boldsymbol{x}_0^{(0)}$, a qual é obtida de forma aleatória. A partir daí, são geradas novas estimativas candidatas a solução ótima do problema, sempre definidas pelo processo iterativo dos *loops* externos e internos, n_{out} e n_{int}, ($\boldsymbol{x}_j^{(k)}, j = 1, \ldots, n_{int}$ e $k = 1, \ldots, n_{out}$). Caso o vetor solução $\boldsymbol{x}_j^{(k)}$ seja melhor que a solução $\boldsymbol{x}^*$, a nova estimativa armazenada em $\boldsymbol{x}^*$ receberá o valor de $\boldsymbol{x}_j^{(k)}$, ou seja, $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$. Após a finalização dos *loops* externos, onde todas as iterações foram executadas, obtém-se a solução do problema inverso, que resulta no maior valor da função objetivo (caso o problema seja de maximização) ou no menor valor da função objetivo (caso o problema seja de minimização).

No Algoritmo 1 tem-se o pseudocódigo do método Luus-Jaakola. Nas primeiras versões do Luus-Jaakola, $R_j^{(k)}$ era uma matriz diagonal onde os elementos da diagonal variavam entre $-0,5$ e $0,5$.

Algoritmo 1 *Luus-Jaakola (LJ-Clássico)*

Defina o número de loops externos: n_{nout}
Defina o número de loops internos: n_{int}
Defina o fator de contração: ϵ
Defina o intervalo de busca inicial: $\boldsymbol{r}^{(0)}$
Gere uma solução inicial: $\boldsymbol{x}_0^{(0)}$

1: **para** $k = 1$ até n_{out} **faça**
2: **para** $j = 1$ até n_{int} **faça**
3: $\boldsymbol{x}_j^{(k)} = \boldsymbol{x}^* + R_j^{(k)}\boldsymbol{r}^{(k-1)}$
4: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
5: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
6: **fim se**
7: **fim para**
8: $\boldsymbol{r}^{(k)} = (1 - \epsilon)\boldsymbol{r}^{(k-1)}$
9: **fim para**

Nota: R é uma matriz diagonal de números aleatórios entre -0,5 e 0,5.

3.2 Algoritmo de Colisão de Partículas (PCA)

O Algoritmo de Colisão de Partículas (do inglês *Particle Collision Algorithm*, PCA) foi desenvolvido em 2005 por Sacco e Oliveira [25], e assemelha-se ao algoritmo de Recozimento Simulado (do inglês *Simulated Annealing*, SA). O PCA se associa com fenômenos físicos de absorção (*absorption*) e espalhamento (*scattering*) que ocorrem em reatores

nucleares [25].

Este algoritmo (PCA), é inspirado no processo de colisão de partículas nucleares, onde a partícula é espalhada por um núcleo-alvo e absorvida pelo núcleo de destino [30]. No Algoritmo de Colisão de Partículas primeiro é escolhida uma configuração inicial; então há uma modificação da configuração antiga (configuração inicial) em uma nova. As qualidades das duas configurações são comparadas. Em seguida, é tomada uma decisão se a nova configuração é "aceitável". Caso afirmativo, ela é considerada como a configuração antiga para a próxima etapa. Se não for aceitável, o algoritmo prossegue com uma nova mudança da configuração antiga [25].

De acordo com Lobato et al. [14] e Luz, Becceneri e Campus Velho [16, 17], o PCA é inicializado com a seleção (randômica ou não) de um projeto inicial denominado **OldConfig**, o qual é modificado de forma estocástica por uma função denominada *Perturbation()* que, por sua vez, permite a obtenção de uma nova solução candidata a ótimo denominada **NewConfig**.

Conforme Sacco e Oliveira [25], o PCA possui uma quantidade menor de parâmetros a serem configurados em comparação a algoritmos clássicos como Algoritmos Genéticos, por exemplo, pois a maioria dos parâmetros são definidos por rotinas aleatórias.

Assim como no Luus-Jaakola, no PCA se faz necessário definir os intervalos de busca para as variáveis que se deseja estimar, de acordo com o problema a ser tratado, além do número de vezes (n_{PCA}) em que ocorrerá o processo iterativo, onde novas estimativas são geradas e avaliadas de acordo com a função objetivo [30]. Caso uma nova solução seja a melhor, uma vizinhança é explorada. O número de vezes em que novas soluções serão geradas em torno da melhor solução é determinado pelo parâmetro n_{EXP}, o qual está relacionado às rotinas *Perturbation()* e *SmallPerturbation()*. Cabe ressaltar que os parâmetros n_{PCA} e n_{EXP} não se alteram durante a execução do PCA.

No que se refere ao valor da função objetivo, a nova solução é comparada com a solução anterior, onde pode ser aceita ou não. Caso a nova solução não seja aceita em um primeiro momento, verifica-se a probabilidade da mesma ser aceita levando em consideração sua aptidão, processo esse realizado pela rotina denominada *Scattering()*.

No Algoritmo 2, tem-se o pseudocódigo da rotina principal do PCA, onde f é a função objetivo, n é o número de variáveis do problema (dimensão) e **BestConfig** é o vetor contendo a melhor solução. O algoritmo verifica se $f(\textbf{NewConfig}) < f(\textbf{OldConfig})$ e, caso seja verdade, então o vetor **OldConfig** recebe os valores do vetor **NewConfig**; além

disso, se $f(\mathbf{NewConfig}) < f(\mathbf{BestConfig})$ então o vetor **BestConfig** também recebe os valores do vetor **NewConfig**. A rotina principal do PCA (Algoritmo 2) chama outras três funções, a saber: *Perturbation()*, *Exploitation()* e *Scattering()*, descrita nos Algoritmos 3, 4 e 5, respectivamente.

Algoritmo 2 *PCA: principal() (PCA-Clássico)*

Defina o número de vezes que o PCA irá executar o processo: n_{PCA}
Defina o número de explorações do PCA em torno de uma solução: n_{EXP}
Defina o número de variáveis (dimensão): n
Defina o vetor que contém o limite inferior: **min**
Defina o vetor que contém o limite superior: **max**
Gere um vetor com estimativas iniciais: **NewConfig**

faça:
OldConfig = **NewConfig**
BestConfig = **OldConfig**

```
1:  para k = 1 até n_PCA faça
2:     Perturbation()
3:     se (f(NewConfig) < f(OldConfig)) então
4:        OldConfig = NewConfig
5:        se (f(NewConfig) < f(BestConfig)) então
6:           BestConfig = NewConfig
7:        fim se
8:        Exploitation()
9:     senão
10:       Scattering()
11:    fim se
12: fim para
```

No Algoritmo 3 uma variável aleatória, R, é configurada usando uma randomização que gera um número aleatório entre 0 e 1 (linha 1), **NewConfig** é o vetor de novas soluções e **max** e **min** são os vetores que contém os limites superior e inferior, respectivamente. Caso um elemento de **NewConfig** seja menor do que o seu limite inferior, então esse elemento de **NewConfig** recebe o respectivo valor de **min** e, caso seja maior do que o limite superior, esse elemento de **NewConfig** recebe o respectivo valor de **max**, isso mantém a busca sempre dentro do intervalo de busca inicial.

Algoritmo 3 *PCA: Perturbation() (PCA-Clássico)*

1: **NewConfig** = **OldConfig** + {R(**max** - **OldConfig**) + (I-R)(**OldConfig** - **min**) }
2: **para** $i = 1$ até n **faça**
3: **se** (NewConfig$_i$ < min$_i$) **então**
4: NewConfig$_i$ = min$_i$
5: **fim se**
6: **fim para**
7: **para** $i = 1$ até n **faça**
8: **se** (NewConfig$_i$ > max$_i$) **então**
9: NewConfig$_i$ = max$_i$
10: **fim se**
11: **fim para**

Nota: R é uma matriz diagonal de números randômicos entre 0 e 1 e I é a matriz Identidade.

Já no Algoritmo 4, a função *SmallPerturbation()* é chamada e, na linha 4, o vetor **OldConfig** recebe os valores armazenados em **NewConfig**, caso f(**NewConfig**) < f(**OldConfig**). Já na linha 6, o vetor **BestConfig** recebe os valores armazenados no vetor **NewConfig** se f(**NewConfig**) < f(**BestConfig**).

Algoritmo 4 *PCA: Exploitation() (PCA-Clássico)*

1: **para** $j = 1$ até n_{EXP} **faça**
2: *SmallPerturbation()*
3: **se** (f(**NewConfig**) < f(**OldConfig**)) **então**
4: **OldConfig** = **NewConfig**
5: **se** (f(**NewConfig**) < f(**BestConfig**)) **então**
6: **BestConfig** = **NewConfig**
7: **fim se**
8: **fim se**
9: **fim para**

Na função *SmallPerturbation()*, executada dentro do Algoritmo 4 e descrita no Algoritmo 6, cada elemento do vetor **Upper** recebe um valor aleatório e se esse valor ultrapassar o correspondente valor de **max**, então o elemento de **Upper** recebe o valor de **max** para evitar que ultrapasse o limite superior do intervalo. De maneira análoga, **Lower** recebe um valor aleatório e se esse valor ultrapassar o valor de **min**, então **Lower** recebe o valor de **min** para evitar que ultrapasse o limite inferior do intervalo. Na linha 13, **NewConfig** recebe um valor aleatório e a mesma verificação que é feita em **Upper** e **Lower**, é feita em **NewConfig**.

Por fim, a função *Scattering()* representada no Algoritmo 5, verifica se a variável

p_scattering, definida por $1 - \frac{f(\mathbf{Bestconfig})}{f(\mathbf{Newconfig})}$, é maior que um número aleatório gerado entre 0 e 1 e, caso seja, a função *FOldConfig()* descrita no Algoritmo 7, é executada, na qual verifica também se $f(\mathbf{OldConfig}) < f(\mathbf{BestConfig})$ e, caso a desigualdade seja verdadeira, o vetor **BestConfig** recebe os valores do vetor **OldConfig**. Porém, se *p_scattering* não for maior, a função *Exploitation()* é executada. A função *FOldConfig()* que gera um número aleatório, que permanece ao intervalo entre 0 e 1, e cria um vetor **OldConfig**.

Algoritmo 5 *PCA: Scattering() (PCA-Clássico)*

1: **faça:** *p_scattering* = 1-f(**BestConfig**)/f(**NewConfig**)
2: **se** (*p_scattering* > *rand*) **então**
3: *FOldConfig()*
4: **se** (f(**OldConfig**) < f(**BestConfig**)) **então**
5: **BestConfig** = **OldConfig**
6: **fim se**
7: **senão**
8: *Exploitation()*
9: **fim se**

Nota: *rand* é um número randômico entre 0 e 1.

Algoritmo 6 *PCA: SmallPerturbation() (PCA-Clássico)*

1: **Upper** = **OldConfig** + $|(0.2\ R\ \textbf{OldConfig})|$
2: **para** $i = 1$ até n **faça**
3: **se** $(\text{Upper}_i > \max_i)$ **então**
4: $\text{Upper}_i = \max_i$
5: **fim se**
6: **fim para**
7: **Lower** = **OldConfig** - $|(0.2\ R\ \textbf{OldConfig})|$
8: **para** $i = 1$ até n **faça**
9: **se** $(\text{Lower}_i < \min_i)$ **então**
10: $\text{Lower}_i = \min_i$
11: **fim se**
12: **fim para**
13: **NewConfig** = $\frac{1}{2}\{$**OldConfig** + [**Lower** + R (**Upper - Lower**)$\}$;
14: **para** $i = 1$ até n **faça**
15: **se** $(\text{NewConfig}_i < \min_i)$ **então**
16: $\text{NewConfig}_i = \min_i$
17: **fim se**
18: **fim para**
19: **para** $i = 1$ até n **faça**
20: **se** $(\text{NewConfig}_i > \max_i)$ **então**
21: $\text{NewConfig}_i = \max_i$
22: **fim se**
23: **fim para**

Nota: R é uma matriz diagonal de números randômicos entre 0 e 1.

Algoritmo 7 *PCA: FOldConfig() (PCA-Clássico)*

1: **OldConfig** = **min** + $\{R($**max - min**$)\}$

Nota: R é uma matriz diagonal de números randômicos entre 0 e 1.

3.3 Métodos Luus-Jaakola e Algoritmo de Colisão de Partículas Modificados

Nessa seção são apresentadas algumas propostas de modificações para os algoritmos LJ e PCA. Como na versão clássica do método Luus-Jaakola, descrita na Seção 3.1, um novo vetor $\boldsymbol{x}_j^{(k)}$ é gerado e compara-se $f(\boldsymbol{x}_j^{(k)})$ com $f(\boldsymbol{x}^*)$, muitas vezes descarta-se todos os elementos de $\boldsymbol{x}_j^{(k)}$ quando $f(\boldsymbol{x}_j^{(k)}) > f(\boldsymbol{x}^*)$. No entanto, pode ocorrer que o vetor $\boldsymbol{x}_j^{(k)}$ contenha algum componente melhor do que seu correspondente em $\boldsymbol{x}^*$. Sendo assim, nessa primeira modificação é proposta a combinação entre os elementos $\boldsymbol{x}_j^{(k)}$ e $\boldsymbol{x}^*$. Como no problema de transferência de calor abordado nesse trabalho busca-se estimar dois parâmetros (λ e ρc_p), no desenvolvimento a seguir, são considerados os vetores contendo apenas duas componentes.

A alteração foi realizada após a linha 3 do Algoritmo 1, onde são realizadas 3 avaliações da função objetivo, combinando os elementos dos vetores $\boldsymbol{x}^*$ e $\boldsymbol{x}_j^{(k)}$, de maneira a aproveitar os elementos que estejam próximos de seus "valores alvo" em algum dos dois vetores. Em outras palavras, a cada nova estimativa dos parâmetros, é feita uma combinação com a melhor estimativa, ou seja, se a melhor for $\boldsymbol{x}^*=(x_1^*, x_2^*)$ e a nova for $\boldsymbol{x}_j^{(k)}=(x_1, x_2)$, então, com a combinação tem-se 3 estimativas a serem avaliadas, incluindo a que acabou de ser gerada: (x_1, x_2), (x_1^*, x_2) e (x_1, x_2^*). Consequentemente, o número de *loops* internos tem que ser um terço da versão original do algoritmo Luus-Jaakola. No Algoritmo 8 é mostrada a referida alteração.

Essa mesma idéia de combinação implementada no LJ, foi implementada no PCA. No entanto, as três avaliações da função objetivo são realizadas dentro da função *Exploitation()*, conforme Algoritmo 9. Com essa alteração na função, o método PCA passa a aproveitar, assim como ocorreu no método LJ, os melhores resultados em cada vetor, ao contrário da versão clássica que descarta o vetor inteiro, caso alguma solução seja pior do que a solução atual.

Algoritmo 8 *Luus-Jaakola (com combinação de estimativas) (LJ-M1)*

Defina o número de loops externos: n_{nout}
Defina o número de loops internos: n_{int}
Defina o fator de contração: ϵ
Defina o intervalo de busca inicial: $\boldsymbol{r}^{(0)}$
Gere uma solução inicial: $\boldsymbol{x}_0^{(0)}$

para $k = 1$ até n_{out} **faça**
  para $j = 1$ até $\frac{n_{int}}{3}$ **faça**
    $\boldsymbol{x}_j^{(k)} = R_j^{(k)}\ \boldsymbol{r}_j^{(k-1)}$
    $\boldsymbol{xaux}_{1j}^{(k)} = (x_1^*, x_2)$
    $\boldsymbol{xaux}_{2j}^{(k)} = (x_1, x_2^*)$
    se $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
      $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
    fim se
    se $f(\boldsymbol{xaux}_{1j}^{(k)}) < f(\boldsymbol{x}^*)$ **então**
      $\boldsymbol{x}^* = \boldsymbol{xaux}_{1j}^{(k)}$
    fim se
    se $f(\boldsymbol{xaux}_{2j}^{(k)}) < f(\boldsymbol{x}^*)$ **então**
      $\boldsymbol{x}^* = \boldsymbol{xaux}_{2j}^{(k)}$
    fim se
  fim para
  $\boldsymbol{r}^{(k)} = (1 - \epsilon)\boldsymbol{r}^{(k-1)}$
fim para

Nota: R é uma matriz diagonal de números aleatórios entre -0,5 e 0,5.

Algoritmo 9 *PCA: Exploitation() (com combinação de estimativas) (PCA-M1)*

1: **para** $j = 1$ até n_{EXP} **faça**
2: *SmallPerturbation()*
3: **Bestaux**$_1$ = (BestConfig$_1$, NewConfig$_2$)
4: **Bestaux**$_2$ = (NewConfig$_1$, BestConfig$_2$)
5: **se** (f(**NewConfig**) $<$ f(**OldConfig**)) **então**
6: **OldConfig** = **NewConfig**
7: **se** (f(**NewConfig**) $<$ f(**BestConfig**)) **então**
8: **BestConfig** = **NewConfig**
9: **fim se**
10: **fim se**
11: **se** (f(**Bestaux**$_1$) $<$ f(**OldConfig**)) **então**
12: **OldConfig** = **Bestaux**$_1$
13: **se** (f(**Bestaux**$_1$) $<$ f(**BestConfig**)) **então**
14: **BestConfig** = **Bestaux**$_1$
15: **fim se**
16: **fim se**
17: **se** (f(**Bestaux**$_2$) $<$ f(**OldConfig**)) **então**
18: **OldConfig** = **Bestaux**$_2$
19: **se** (f(**Bestaux**$_2$) $<$ f(**BestConfig**)) **então**
20: **BestConfig** = **Bestaux**$_2$
21: **fim se**
22: **fim se**
23: **fim para**

No Algoritmo 1, a contração do intervalo de busca para cada uma das variáveis é feita sempre a uma taxa constante, ou seja, $\boldsymbol{r}^{(k)} = (1-\epsilon)\boldsymbol{r}^{(k-1)}$, independentemente se os intervalos possuem ou não a mesma amplitude. O Algoritmo 10 (linhas 8 a 11), apresenta uma alteração realizada no Algoritmo 1 (linha 8), cuja redução do intervalo de busca é dada por $\epsilon = \frac{10^0}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, onde $i = 1, \ldots, n$.

Além disso, foram realizadas outras alterações no Algoritmo 1, ou seja, na linha 9 do Algoritmo 10: $\epsilon = \frac{10^{-1}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$ (LJ-M3), $\epsilon = \frac{10^{-3}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$ (LJ-M4) e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$ (LJ-M5), onde $i = 1, \ldots, n$.

Algoritmo 10 *Luus-Jaakola (com alteração na contração do intervalo de busca) (LJ-M2)*

Defina o número de loops externos: n_{nout}
Defina o número de loops internos: n_{int}
Defina o intervalo de busca inicial: $\boldsymbol{r}^{(0)}$
Gere uma solução inicial: $\boldsymbol{x}_0^{(0)}$

1: **para** $k = 1$ até n_{out} **faça**
2: **para** $j = 1$ até n_{int} **faça**
3: $\boldsymbol{x}_j^{(k)} = \boldsymbol{x}^* + R_j^{(k)} \boldsymbol{r}^{(k-1)}$
4: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
5: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
6: **fim se**
7: **fim para**
8: **para** $i = 1$ até n **faça**
9: $\epsilon = \frac{10^0}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$
10: $r_i^{(k)} = \epsilon \, r_i^{(k-1)}$
11: **fim para**
12: **fim para**

Nota: R é uma matriz diagonal de números randômicos entre -0,5 e 0,5.

Também foi realizada uma alteração no fator de contração levando em consideração o número de *loops* externos do Algoritmo 1. O Algoritmo 11 (linhas 8 a 11), apresenta a alteração feita na linha 8 do Algoritmo 1, onde $\epsilon = \frac{10^0}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $i = 1, \ldots, n$.

Assim como ocorrido na modificação anterior, foram realizadas outras alterações no Algoritmo 1, ou seja, na linha 9 do Algoritmo 11: $\epsilon = \frac{10^{-1}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$ (LJ-M7), $\epsilon = \frac{10^{-3}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$ (LJ-M8) e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$ (LJ-M9), onde $i = 1, \ldots, n$.

Algoritmo 11 *Luus-Jaakola (com alteração na contração do intervalo de busca) (LJ-M6)*

Defina o número de loops externos: n_{nout}
Defina o número de loops internos: n_{int}
Defina o intervalo de busca inicial: $\boldsymbol{r}^{(0)}$
Gere uma solução inicial: $\boldsymbol{x}_0^{(0)}$

1: **para** $k = 1$ até n_{out} **faça**
2: **para** $j = 1$ até n_{int} **faça**
3: $\boldsymbol{x}_j^{(k)} = \boldsymbol{x}^* + R_j^{(k)} \boldsymbol{r}^{(k-1)}$
4: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
5: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
6: **fim se**
7: **fim para**
8: **para** $i = 1$ até n **faça**
9: $\epsilon = \frac{10^0}{(max_i - min_i)}^{\frac{k}{n_{out}}}$
10: $r_i^{(k)} = \epsilon \, r_i^{(0)}$
11: **fim para**
12: **fim para**

Nota: R é uma matriz diagonal de números randômicos entre -0,5 e 0,5.

Posteriormente, uma terceira alteração foi realizada na estrutura do método Luus-Jaakola, fazendo com que novas estimativas sejam obtidas utilizando a mesma técnica de combinação de estimativas apresentada no Algoritmo 8, porém, nessa nova versão, a redução dos intervalos é obtida por $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}, i = 1, ..., n$. No Algoritmo 12, linha 16, é mostrada a alteração no fator de contração do intervalo.

Com o intuito de tornar o método Luus-Jaakola mais robusto e eficiente na solução do problema de transferência de calor, de forma similiar à alteração descrita no Algoritmo 12, outra alteração na forma como o algoritmo vai trabalhar o intervalo também foi realizada, tomando $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, onde a variável k varia de 1 até n_{out}. Nesse caso ainda se está trabalhando com as combinações da nova estimativa com a melhor estimativa armazenada em $\boldsymbol{x}^*$. No Algoritmo 13, na linha 16, é apresentada a referida alteração para obter ϵ.

Algoritmo 12 *Luus-Jaakola (combinação de estimativas com alteração na contração do intervalo de busca) (LJ-M10)*

Defina o número de loops externos: n_{nout}

Defina o número de loops internos: n_{int}

Defina o intervalo de busca inicial: $\boldsymbol{r}^{(0)}$

Gere uma solução inicial: $\boldsymbol{x}^{(0)}$

1: **para** $k = 1$ até n_{out} **faça**
2: **para** $j = 1$ até $\frac{n_{int}}{3}$ **faça**
3: $\boldsymbol{x}_j^{(k)} = R_j^{(k)}\ \boldsymbol{r}_j^{(k-1)}$
4: $\boldsymbol{xaux}_{1j}^{(k)} = (x_1^*, x_2)$
5: $\boldsymbol{xaux}_{2j}^{(k)} = (x_1, x_2^*)$
6: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
7: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
8: **fim se**
9: **se** $f(\boldsymbol{xaux}_{1j}^{(k)}) < f(\boldsymbol{x}^*)$ **então**
10: $\boldsymbol{x}^* = \boldsymbol{xaux}_{1j}^{(k)}$
11: **fim se**
12: **se** $f(\boldsymbol{xaux}_{2j}^{(k)}) < f(\boldsymbol{x}^*)$ **então**
13: $\boldsymbol{x}^* = \boldsymbol{xaux}_{2j}^{(k)}$
14: **fim se**
15: **para** $i = 1$ até n **faça**
16: $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$
17: $r_i^{(k)} = \epsilon\ r_i^{(k-1)}$
18: **fim para**
19: **fim para**
20: **fim para**

Algoritmo 13 *Luus-Jaakola (combinação de estimativas com alteração na contração do intervalo de busca) (LJ-M11)*

Defina o número de loops externos: n_{nout}
Defina o número de loops internos: n_{int}
Defina o intervalo de busca inicial: $\boldsymbol{r}^{(0)}$
Gere uma solução inicial: $\boldsymbol{x}^{(0)}$

1: **para** $k = 1$ até n_{out} **faça**
2: **para** $j = 1$ até $\frac{n_{int}}{3}$ **faça**
3: $\boldsymbol{x}_j^{(k)} = R_j^{(k)}\ \boldsymbol{r}_j^{(k-1)}$
4: $\boldsymbol{xaux}_{1j}^{(k)} = (x_1^*, x_2)$
5: $\boldsymbol{xaux}_{2j}^{(k)} = (x_1, x_2^*)$
6: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
7: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
8: **fim se**
9: **se** $f(\boldsymbol{xaux}_{1j}^{(k)}) < f(\boldsymbol{x}^*)$ **então**
10: $\boldsymbol{x}^* = \boldsymbol{xaux}_{1j}^{(k)}$
11: **fim se**
12: **se** $f(\boldsymbol{xaux}_{2j}^{(k)}) < f(\boldsymbol{x}^*)$ **então**
13: $\boldsymbol{x}^* = \boldsymbol{xaux}_{2j}^{(k)}$
14: **fim se**
15: **para** $i = 1$ até n **faça**
16: $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$
17: $r_i^{(k)} = \epsilon\ r_i^{(0)}$
18: **fim para**
19: **fim para**
20: **fim para**

Finalizando essa seção contendo as modificações nos métodos LJ e PCA, no Algoritmo 14 são apresentadas modificações na função *Perturbation()* nas linhas 1, 2 e 3, referente ao Algoritmo de Colisão de Partículas. Nas linhas 1 e 2 são definidos os valores do intervalo inferior e superior, respectivamente. Na linha 3, os valores do vetor solução são definidos.

Algoritmo 14 *PCA: Perturbation (modificado) (PCA-M2)*

1: $\boldsymbol{L}$ = **BestConfig** - $\frac{1}{2}$ (**max** − **min**)
2: $\boldsymbol{U}$ = **BestConfig** + $\frac{1}{2}$ (**max** − **min**)
3: **NewConfig** = $\frac{1}{4}((3\textbf{BestConfig}) + (\boldsymbol{L} + R(\boldsymbol{U} - \boldsymbol{L})))$
4: **para** $i = 1$ até n **faça**
5: **se** (NewConfig$_i$ < min$_i$) **então**
6: NewConfig$_i$ = min$_i$
7: **fim se**
8: **fim para**
9: **para** $i = 1$ até n **faça**
10: **se** (NewConfig$_i$ > max$_i$) **então**
11: NewConfig$_i$ = max$_i$
12: **fim se**
13: **fim para**

Nota: R é uma matriz diagonal de números aleatórios entre 0 e 1.

3.4 Hibridizações entre os Métodos Luus-Jaakola e Algoritmo de Colisão de Partículas Modificados

Além das modificações descritas na Seção 3.3, também foram realizadas várias hibridizações entre os algoritmos LJ e PCA com o objetivo de se obter uma redução do número de avaliações da função objetivo. Tais hibridizações foram realizadas com base nas versões clássicas (canônicas) dos referidos algoritmos, ou seja, Algoritmo 1 (LJ) e Algoritmos 2 a 7 (PCA).

No método LJ, foram realizados dois tipos de hibridização, a primeira chama a função *Scattering* do método PCA, que realiza um espalhamento da partícula, ou seja, busca uma solução em outro local do domínio, caso $f(\boldsymbol{x}_j^{(k)}) > f(\boldsymbol{x}^*)$. Outra hibridização se refere à perturbação realizada pelo método PCA, com o objetivo de escapar de mínimos locais. Já no PCA, um intervalo de busca r, do método LJ, foi acrescentado, assim como um fator de contração. Com essas hibridizações, objetiva-se a obtenção de melhores resultados, por fazerem com que os métodos verifiquem outras alternativas para melhorarem a solução antes de executarem a próxima iteração.

No método Luus-Jaakola, a primeira hibridização foi realizada após a linha 6 no Algoritmo 1, onde, agora, o algoritmo faz a chamada da função *Scattering()*, Algoritmo 5, utilizada no PCA, quando $f(\boldsymbol{x}_j^{(k)}) > f(\boldsymbol{x}^*)$. Tal alteração se justifica pela fato de, na versão clássica do Luus-Jaakola, quando $f(\boldsymbol{x}_j^{(k)}) > f(\boldsymbol{x}^*)$, o algoritmo não faz nada e não entra no *loop* "**se**"(linhas 4 a 6). Nessa primeira hibridização é adicionada uma condição "**senão**", incluindo a função *Scattering()*, conforme linhas 6 a 8 do Algoritmo 15, e com uma alteração na função *Scattering()* conforme, Algoritmo 16.

Algoritmo 15 *Luus-Jaakola (modificado e hibridizado) (LJ-H1)*

Defina o número de loops externos: n_{nout}
Defina o número de loops internos: n_{int}
Defina o número de variáveis: n
Defina o número de explorações: n_{EXP}
Defina o fator de redução de intervalo de busca: ϵ_{scat}
Defina o fator de contração: ϵ
Defina o intervalo de busca inicial: $\boldsymbol{r}^{(0)}$
Gere uma solução inicial: $\boldsymbol{x}^{(0)}$

1: **para** $k = 1$ até n_{out} **faça**
2: **para** $j = 1$ até n_{int} **faça**
3: $\boldsymbol{x}_j^{(k)} = \boldsymbol{x}^* + R_j^{(k)}\boldsymbol{r}^{(k-1)}$
4: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
5: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
6: **senão**
7: *Scattering()*
8: **fim se**
9: **fim para**
10: $\boldsymbol{r}^{(k)} = (1 - \epsilon)\boldsymbol{r}^{(k-1)}$
11: **fim para**

Nota: R é uma matriz diagonal de números randômicos entre -0,5 e 0,5.

Algoritmo 16 *Luus-Jaakola: Scattering (modificado e hibridizado) (LJ-H1)*

1: **faça:** $p\_scattering$ = 1-f(**BestConfig**)/f(**NewConfig**)
2: **se** ($p\_scattering < rand$) **então**
3: $\boldsymbol{xaux} = \boldsymbol{x}_j^{(k)}$
4: **para** $i = 1$ até n_{EXP} **faça**
5: $\boldsymbol{x}_j^{(k)} = \boldsymbol{xaux} + \epsilon_{scat}\ R\boldsymbol{r}^{(k-1)}$
6: **para** $w = 1$ até n **faça**
7: **se** ($x_w < min_w$) **então**
8: $x_w = min_w$
9: **fim se**
10: **se** ($x_w > max_w$) **então**
11: $x_w = max_w$
12: **fim se**
13: **fim para**
14: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{xaux})$ **então**
15: $\boldsymbol{xaux} = \boldsymbol{x}_j^{(k)}$
16: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
17: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
18: **fim se**
19: **fim se**
20: **fim para**
21: **fim se**

Nota: *rand* é um número aleatório entre 0 e 1, R é uma matriz diagonal de números randômicos entre -0,5 e 0,5 e $\boldsymbol{x}_j^{(k)} = (x_1, \ldots, x_w, \ldots, x_n)$.

Já o Algoritmo 17 explora uma abordagem diferente no método LJ, realizando uma alteração na função *Scattering()* apresentada no Algoritmo 16, a qual é descrita no Algoritmo 18, que por sua vez, faz uma chamada à função *SmallPerturbation()*, representada pelo Algoritmo 19.

Algoritmo 17 *Luus-Jaakola (modificado e hibridizado) (LJ-H2)*

Defina o número de variáveis: n
Defina o número de explorações: n_{EXP}
Defina o número de loops externos: n_{nout}
Defina o número de loops internos: n_{int}
Defina o fator de contração: ϵ
Defina o intervalo de busca inicial: $\boldsymbol{r}^{(0)}$
Gere uma solução inicial: $\boldsymbol{x}^{(0)}$

1: **para** $k = 1$ até n_{out} **faça**
2: **para** $j = 1$ até n_{int} **faça**
3: $\boldsymbol{x}_j^{(k)} = \boldsymbol{x}^* + R_j^{(k)}\boldsymbol{r}^{(k-1)}$
4: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
5: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
6: **senão**
7: *Scattering()*
8: **fim se**
9: **fim para**
10: $\boldsymbol{r}^{(k)} = (1 - \epsilon)\boldsymbol{r}^{(k-1)}$
11: **fim para**

Nota: R é uma matriz diagonal de números randômicos entre -0,5 e 0,5.

Algoritmo 18 *Luus-Jaakola: Scattering (modificado e hibridizado)(LJ-H2)*

1: **faça:** $p\_scattering$ = 1-f(**BestConfig**)/f(**NewConfig**)
2: **se** ($p\_scattering < rand$) **então**
3: $\boldsymbol{xaux} = \boldsymbol{x}_j^{(k)}$
4: **para** $i = 1$ até n_{EXP} **faça**
5: *SmallPerturbation()*
6: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{xaux})$ **então**
7: $\boldsymbol{xaux} = \boldsymbol{x}_j^{(k)}$
8: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
9: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
10: **fim se**
11: **fim se**
12: **fim para**
13: **fim se**

Nota: *rand* é um número randômico entre 0 e 1.

Algoritmo 19 *Luus-Jaakola: SmallPerturbation (hibridizado) (LJ-H2)*

1: **Upper** = $\boldsymbol{x}^*$ + 0.2 $R\boldsymbol{x}^*$
2: **para** $w = 1$ até n **faça**
3: **se** ($\text{Upper}_w > \max_w$) **então**
4: $\text{Upper}_w = \max_w$
5: **fim se**
6: **fim para**
7: **Lower** = $\boldsymbol{x}^*$ - 0.2 $R\boldsymbol{x}^*$
8: **para** $w = 1$ até n **faça**
9: **se** ($\text{Lower}_w < \min_w$) **então**
10: $\text{Lower}_w = \min_w$
11: **fim se**
12: **fim para**
13: $\boldsymbol{x}_j^{(k)} = \frac{1}{2}(\boldsymbol{x}^* + (\textbf{Lower} + R(\textbf{Upper - Lower})))$
14: **para** $w = 1$ até n **faça**
15: **se** ($x_w < \min_w$) **então**
16: $x_w = \min_w$
17: **se** ($x_w > \max_w$) **então**
18: $x_w = \max_w$
19: **fim se**
20: **fim se**
21: **fim para**

Nota: R é uma matriz diagonal de números randômicos entre 0 e 1 e $\boldsymbol{x}_j^{(k)} = (x_1, \ldots, x_w, \ldots, x_n)$.

Uma nova hibridização também foi realizada envolvendo a função *Perturbation()*. No Algoritmo 20 é mostrada a função *Perturbation()* sendo chamada na função principal do algoritmo Luus-Jaakola. Para a utilização da referida função, foi necessário realizar algumas adaptações no algoritmo original (Algoritmo 3). Nos Algoritmos 21 e 22 são apresentadas duas versões para as referidas alterações. No entanto, embora apresentem implementações aparentemente diferentes, os Algoritmos 21 e 14 se equivalem, levando aos mesmos resultados.

Algoritmo 20 *Luus-Jaakola (modificado e hibridizado) (LJ-H3, LJ-H4)*

Defina o número de loops externos: n_{nout}
Defina o número de loops internos: n_{int}
Defina o fator de contração: ϵ
Defina o número de variáveis: n
Defina o intervalo de busca inicial: $\boldsymbol{r}^{(0)}$
Gere uma solução inicial: $\boldsymbol{x}^0$

1: **para** $k = 1$ até n_{out} **faça**
2: **para** $j = 1$ até n_{int} **faça**
3: *Perturbation()*
4: **se** $f(\boldsymbol{x}_j^{(k)}) < f(\boldsymbol{x}^*)$ **então**
5: $\boldsymbol{x}^* = \boldsymbol{x}_j^{(k)}$
6: **fim se**
7: **fim para**
8: $\boldsymbol{r}^{(k)} = (1 - \epsilon)\boldsymbol{r}^{(k-1)}$
9: **fim para**

Algoritmo 21 *Luus-Jaakola: Perturbation (modificado e hibridizado) (LJ-H3)*

1: $\boldsymbol{x}_j^{(k)} = \boldsymbol{x}^* + R\boldsymbol{r}^{(k)}$
2: $\boldsymbol{x}_j^{(k)} = \frac{1}{4}(3\ \boldsymbol{x}^* + \boldsymbol{x}_j^{(k)})$
3: **para** $w = 1$ até n **faça**
4: **se** $(x_w < \min_w)$ **então**
5: $x_w = \min_w$
6: **fim se**
7: **se** $(x_w > \max_w)$ **então**
8: $x_w = \max_w$
9: **fim se**
10: **fim para**

Nota: R é uma matriz diagonal de números aleatórios entre -0,5 e 0,5 e $\boldsymbol{x}_j^{(k)} = (x_1, \ldots, x_w, \ldots, x_n)$.

Algoritmo 22 *Luus-Jaakola: Perturbation (modificado e hibridizado) (LJ-H4)*

1: $\boldsymbol{x}_j^{(k)} = \boldsymbol{x}^* + R\boldsymbol{r}^{(k)}$
2: $\boldsymbol{x}_j^{(k)} = \frac{1}{4}(3\ \boldsymbol{x}_j^{(k)} + \boldsymbol{x}^*)$
3: **para** $w = 1$ até n **faça**
4: **se** $(x_w < \min_w)$ **então**
5: $x_w = \min_w$
6: **fim se**
7: **se** $(x_w > \max_w)$ **então**
8: $x_w = \max_w$
9: **fim se**
10: **fim para**

Nota: R é uma matriz diagonal de números aleatórios entre -0,5 e 0,5 e $\boldsymbol{x}_j^{(k)} = (x_1, \ldots, x_w, \ldots, x_n)$.

Assim como ocorrido com o método Luus-Jaakola, foram realizadas algumas hibridizações no método PCA, tomando como base algumas características do método Luus-Jaakola, em especial, a contração do intervalo de busca. A diferença do Algoritmo 2 do método PCA para o Algoritmo 23, apresentado na sequência do texto, está nas linhas de 1 e 13, onde, no início, o intervalo de busca $\boldsymbol{r}^{(k)}$ é dado por $\boldsymbol{r}^{(0)} = (\mathbf{max} - \mathbf{min})$ e, no final, um novo intervalo de busca é calculado fazendo $\boldsymbol{r}^{(k)} = (1-\epsilon)\boldsymbol{r}^{(k-1)}$. Nesta versão do método, modifica-se também a função original *Perturbation()* apresentada anteriormente no Algoritmo 3, na linha 1, onde o vetor **NewConfig** recebe um valor aleatório que é somado ao vetor **OldConfig**, conforme Algoritmo 24.

Para finalizar, outra modificação no método PCA é realizada de forma similar à modificação anterior, ou seja, no algoritmo principal tem-se as modificações descritas no Algoritmo 23, no entanto a função *Perturbation()* sofre mais uma pequena modificação adicionando ao Algoritmo 24 uma nova linha, que gera um vetor **NewConfig** com uma nova solução, conforme explicitado na linha 2 do Algoritmo 25.

Algoritmo 23 *PCA: principal (modificado e hibridizado) (PCA-H1)*

Defina o número de vezes que o PCA irá executar o processo: n_{PCA}
Defina o número de explorações do PCA em torno de uma solução: n_{EXP}
Defina o número de variáveis (dimensão): n
Defina o fator de contração: ϵ
Defina o vetor que contém o limite inferior: **min**
Defina o vetor que contém o limite superior: **max**
Gere um vetor com estimativas iniciais: **NewConfig**

faça:
OldConfig=**NewConfig**
BestConfig=**OldConfig**

1: $\boldsymbol{r}^{(0)} = \mathbf{max} - \mathbf{min}$
2: **para** $k = 1$ até n_{PCA} **faça**
3: *Perturbation()*
4: **se** (f(**NewConfig**) $<$ f(**OldConfig**)) **então**
5: **OldConfig** = **NewConfig**
6: **se** (f(**NewConfig**) $<$ f(**BestConfig**)) **então**
7: **BestConfig** = **NewConfig**
8: **fim se**
9: *Exploitation()*
10: **senão**
11: *Scattering()*
12: **fim se**
13: $\boldsymbol{r}^{(k)} = (1 - \epsilon)\boldsymbol{r}^{(k-1)}$
14: **fim para**

Algoritmo 24 *PCA: Perturbation(modificado e hibridizado) (PCA-H1)*

1: **NewConfig** = **OldConfig** + $R\ \boldsymbol{r}^{(k-1)}$
2: **para** $i = 1$ até n **faça**
3: **se** (NewConfig$_i$ $<$ min$_i$) **então**
4: NewConfig$_i$ = min$_i$
5: **fim se**
6: **fim para**
7: **para** $i = 1$ até n **faça**
8: **se** (NewConfig$_i$ $>$ max$_i$) **então**
9: NewConfig$_i$ = max$_i$
10: **fim se**
11: **fim para**

Nota: R é uma matriz diagonal de números randômicos entre -0,5 e 0,5.

Algoritmo 25 *PCA: Perturbation(modificado e hibridizado) (PCA-H2)*

1: **NewConfig** = **OldConfig** + $R\ \boldsymbol{r}^{(k-1)}$
2: **NewConfig** $=\frac{1}{4}$ (3 **OldConfig** + **NewConfig**)
3: **para** $i = 1$ até n **faça**
4: **se** (NewConfig$_i$ < min$_i$) **então**
5: NewConfig$_i$ = min$_i$
6: **fim se**
7: **fim para**
8: **para** $i = 1$ até n **faça**
9: **se** (NewConfig$_i$ > max$_i$) **então**
10: NewConfig$_i$ = max$_i$
11: **fim se**
12: **fim para**

Nota: R é uma matriz diagonal de números randômicos entre -0,5 e 0,5.

Capítulo 4

Resultados e Discussões

Neste capítulo são apresentados os resultados obtidos para a estimativa da condutividade térmica, λ, e capacidade volumétrica, ρc_p, presentes na Equação (2.12), por meio dos métodos de otimização Luus-Jaakola, Algoritmo de Colisão de Partículas e variantes de ambos. Os resultados foram gerados em um computador com processador Intel Core i5-5200U 2.2GHz, 8GB de memória RAM, sistema operacional Ubuntu GNU/Linux, sendo utilizada a linguagem de programação C e o software Gnuplot para geração dos gráficos.

4.1 Resultados dos Métodos Luus-Jaakola e Algoritmo de Colisão de Partículas nas Versões Clássicas

No presente trabalho, os métodos foram configurados para realizarem 100 execuções com o intuito de analisar alguns parâmetros estatísticos e, em cada uma dessas execuções, foi contabilizado o número de avaliações da função objetivo (NAF), dada pela Equação (3.2), necessárias para os algoritmos encontrarem os parâmetros de interesse (λ e ρc_p) de acordo com um critério de parada estabelecido *a priori* ($f(\lambda, \rho c_p) \leq 3 \times 10^{-2}$ ou NAF>20000). Logo, os parâmetros estatísticos são calculados levando em consideração esse total de execuções, os quais são: melhor resultado (menor número de avaliações da função objetivo), pior resultado (maior número de avaliações da função objetivo), média, mediana e desvio padrão da condutividade térmica, λ, da capacidade volumétrica, ρc_p, do valor e do número de avaliações da função objetivo e do tempo computacional, conforme pode ser observado nas tabelas ao longo desse Capítulo 4.

A média, μ, é dada por meio da Equação (4.1). Para cada parâmetro têm-se uma média,

$$\mu = \sum_{i=1}^{nc} \frac{y_i}{nc}, \tag{4.1}$$

onde nc representa o número de execuções realizadas (100) e y_i o valor de cada um dos parâmetros obtidos em cada execução.

Já a mediana é uma medida de tendência central que só se aplica a conjuntos ordenados. Ela é definida pelo valor que separa a metade maior da metade menor em um conjunto de dados. Se o conjunto de dados possuir um número ímpar de observações, a mediana é o valor do meio, porém se o conjunto possuir um número par, a mediana é definida como a média aritmética dos dois valores do meio.

Por fim, o desvio padrão representa o quanto os resultados se desviam da média e possui a mesma unidade de medida dos dados analisados. Uma vez conhecida a média, a Equação (4.2) apresenta o desvio padrão σ. Para cada parâmetro têm-se um desvio.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{nc}(y_i - \mu)^2}{nc - 1}} \tag{4.2}$$

Para estimar os valores de λ e ρc_p do problema de transferência de calor descrito pelas Equações (2.12) a (2.15) e resolvido numericamente pelo Método das Diferenças Finitas com a formulação implícita, adotando a malha espacial com 10 nós ($\triangle x = 0,001209$) e malha temporal com 800 nós ($\triangle t = 0,2$), uma vez que esta configuração apresentou melhor custo-benefício no que se refere ao tempo computacional e erro, conforme descrito na Seção 2.4.1. Foram utilizadas as seguintes configurações para os métodos LJ e PCA:

Luus-Jaakola (LJ):

- número de iterações externas: $n_{out} = 100$;
- número de iterações internas: $n_{int} = 50$;
- semente inicial para a geração de números aleatórios: 31.

Algoritmo de Colisão de Partículas (PCA):

- número de vezes que o processo será executado: $n_{pca} = 100$;
- número de explorações em torno de uma solução: $n_{exp} = 50$;
- semente inicial para a geração de números aleatórios: 31.

A Tabela 4.1 mostra os resultados obtidos pelo método Luus-Jaakola (Algoritmo 1),

versão clássica, ou seja, $\boldsymbol{r}^{(k)} = (1-\epsilon)\boldsymbol{r}^{(k-1)}$ e $\epsilon = 0,20$, onde pode-se observar que, no pior resultado, o NAF foi de 1232 e, o melhor resultado, foi obtido com NAF = 227. Isso significa que a tolerância pré-fixada no algoritmo faz o mesmo parar sua execução quando o valor da função objetivo se torna menor do que 3×10^{-2} ou o número de iterações ultrapassar o valor máximo definido em 20000 iterações. A média do NAF foi igual a 681, enquanto a mediana foi de 702.

Segundo Silva, Telles e Semaan [27], quando alguns resultados variam muito, a média pode não ser um parâmetro tão interessante para se avaliar, por isso calcula-se a mediana dos resultados. Já o desvio padrão é uma medida de dispersão que mensura o quanto um conjunto de dados se desvia da média, obtendo um NAF = 148. Cabe ressaltar que o desvio padrão para $f(\lambda, \rho c_p)$ foi de 0,000447, mostrando que os resultados não desviam muito da média. Para esta versão do algoritmo, o melhor (menor) tempo computacional necessário para estimar os parâmetros de interesse foi igual a 4543 ms e o pior (maior) tempo foi igual a 24100 ms. A média e a mediana foram, respectivamente, iguais a 14561 ms e 14837 ms. Já o desvio padrão foi igual a 3430 ms.

Para verificar o quão distante estão a média e o pior resultado em relação ao melhor resultado dos parâmetros λ e ρc_p, tomando como base o menor NAF, foi calculado os respectivos erros relativos, em módulo. Na versão clássica do LJ, os erros relativos da média para λ e ρc_p foram iguais a 0,025173 W/mK e 0,000063 $\times 10^6$ s/m, respectivamente, enquanto para o pior caso foram de 0,004334 W/mK e 0,000018 s/m, respectivamente.

Tabela 4.1: Resultados obtidos pelo método LJ na versão clássica para o problema de transferência de calor.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,843451	3862946,26667	0,029427	227	4543
Pior	11,792125	3862876,380008	0,029125	1232	24100
Média	11,545312	3862700,318705	0,029220	681	14561
Mediana	11,332384	3856706,222554	0,029200	702	14837
Desvio Padrão	0,246520	7238,788952	0,000447	148	3430

Na Figura 4.1 são apresentados os perfis da temperatura experimental, das temperaturas numéricas para o melhor caso, pior caso e para a média, bem como o perfil com os mesmos parâmetros utilizados por Carollo [4] (denominado, no gráfico, como Carollo (2010)), referentes ao método LJ na versão clássica (LJ-Clássico), onde é possível verificar que os resultados numéricos estão muito próximos dos resultados obtidos experimentalmente por Carollo [4]. Enquanto na Figura 4.2 são apresentados os erros absolutos

entre as temperaturas numéricas e a experimental para o melhor caso, pior caso, para a média e para os parâmetros utilizados por Carollo [4], referentes à versão clássica do método LJ (LJ-Clássico), demonstrando que o erro absoluto para esses três casos são próximos entre si, o que representa uma robustez nos resultados.

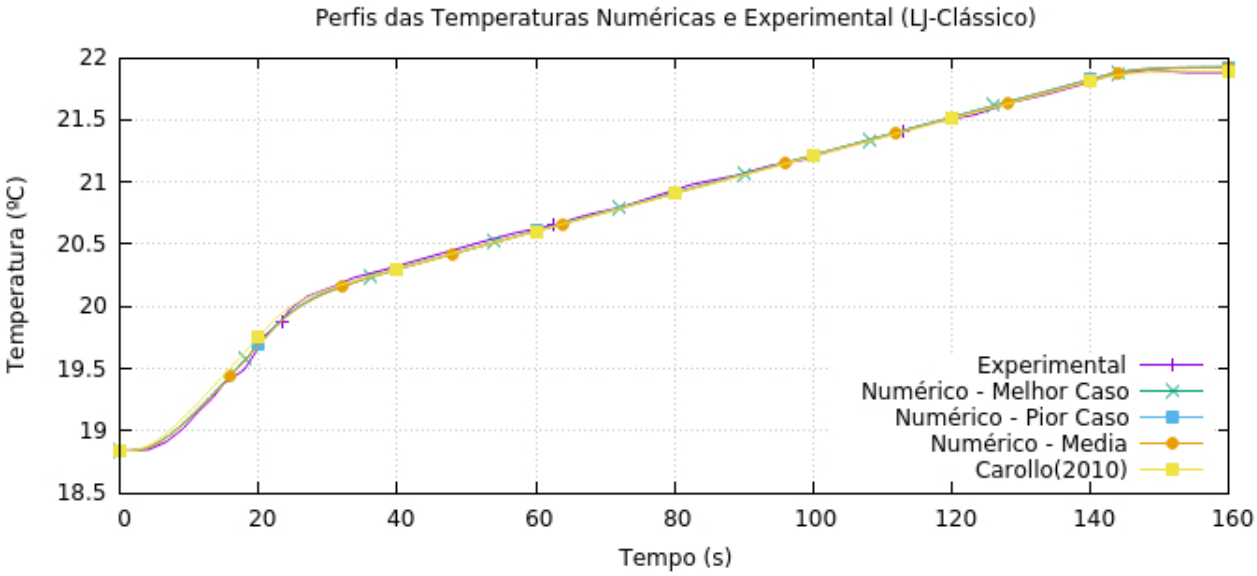


Figura 4.1: Resultados dos perfis das temperaturas obtidos com os parâmetros estimados pelo método LJ-clássico no melhor caso, pior caso e média, resultado obtido utilizando os parâmetros de Carollo [4] e dados experimentais.
Fonte: O Autor, 2019.

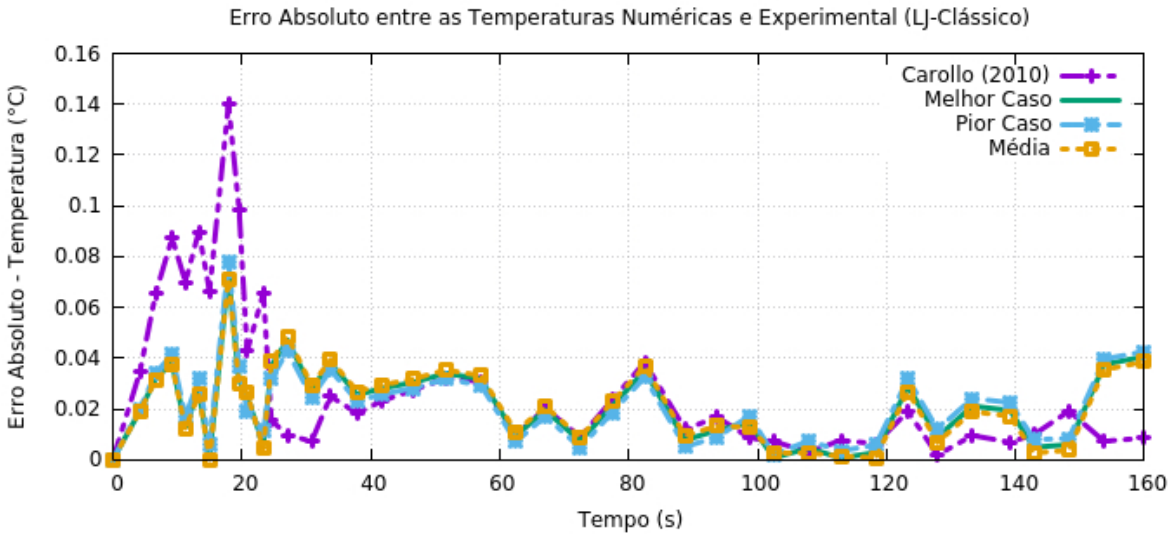


Figura 4.2: Erros absolutos obtidos com os parâmetros estimados pelo método LJ-clássico no melhor caso, pior caso e média e o erro obtido utilizando os parâmetros de Carollo [4].
Fonte: O Autor, 2019.

Já na Tabela 4.2 são mostrados os resultados obtidos pela versão clássica do método PCA (Algoritmos 2 a 7), sendo o número de avaliações da função objetivo igual a 946, para o pior caso e, NAF = 85, para o melhor caso. A média para o NAF foi de 416 e, para a mediana, NAF = 409, por outro lado, o desvio padrão resultou em NAF = 183. No que se refere ao tempo computacional, a versão clássica do PCA obteve 2787 *ms* no melhor tempo e 30242 *ms* no pior, enquanto a média do tempo computacional foi de 12948 *ms* e a mediana igual a 12524 *ms*. Concluindo, o desvio padrão do tempo foi igual a 5753 *ms*. Nessa versão clássica do PCA, os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,001988 W/mK e 0,000669 $\times 10^6$ s/m e para o pior caso foram de 0,003686 W/mK e 0,001703 $\times 10^6$ s/m, respectivamente, todos em relação ao melhor resultado obtido.

Tabela 4.2: Resultados obtidos pelo método PCA na versão clássica para o problema de transferência de calor.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,62529	3867808,499111	0,028564	85	2787
Pior	11,668145	3874394,593906	0,029394	946	30242
Média	11,648396	3865222,851587	0,029230	416	12948
Mediana	11,850866	3862856,357012	0,029547	409	12524
Desvio Padrão	0,220402	6777,322211	0,000423	183	5753

Na Figura 4.3 são apresentados os perfis da temperatura experimental, das temperaturas numéricas para o melhor caso, pior caso e para a média, bem como o perfil com os mesmos parâmetros utilizados por Carollo [4] (denominado Carollo (2010)), referentes ao método PCA na versão clássica (PCA-Clássico), onde é possível perceber que durante o tempo de simulação igual a 160 *s*, as temperaturas numéricas se aproximam da temperatura experimental. Enquanto na Figura 4.4 são apresentados os erros absolutos entre as temperaturas numéricas e a experimental para o melhor caso, pior caso, para a média e para os parâmetros utilizados por Carollo [4], tendo como base versão clássica do método PCA (PCA-Clássico).

Também foi analisada como ocorreram as avaliações da função objetivo ao longo das 100 execuções de cada método. Na Tabela 4.3 é mostrada a distribuição de frequências para os métodos LJ e PCA, nas suas respectivas versões clássicas. No LJ, o maior número de ocorrências se deu entre 700 e 880 NAF. Já o PCA apresentou o maior número de ocorrências entre 400 e 500 NAFs. Sendo assim, o PCA, na maioria das vezes, resolveu o problema com um número consideravelmente menor de iterações, quando comparado com o LJ. Na Figura 4.5, é apresentada ainda, a distribuição dos dados utilizando o boxplot,

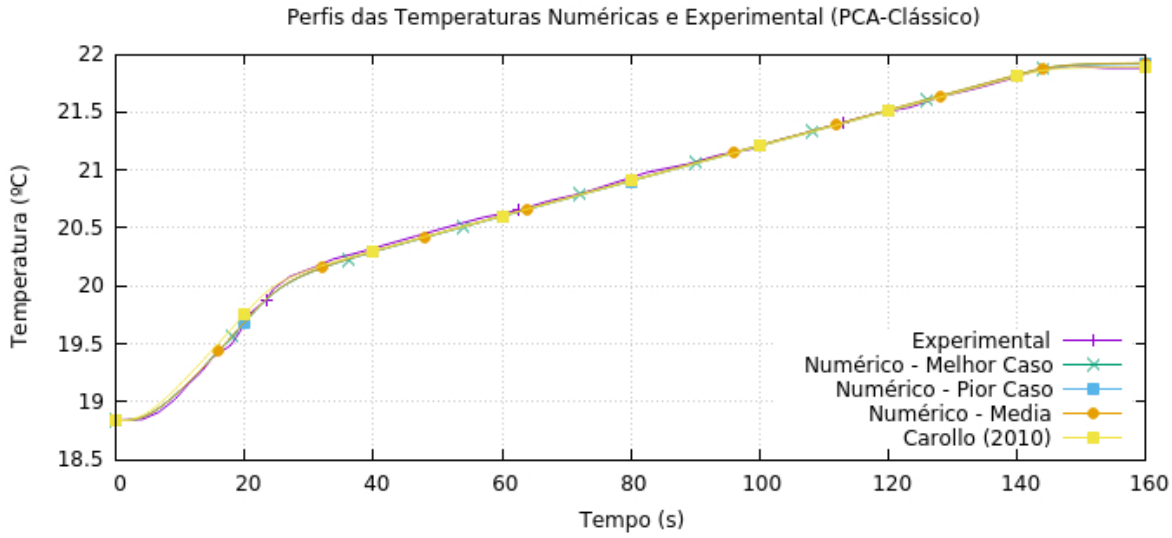


Figura 4.3: Resultados dos perfis das temperaturas obtidos com os parâmetros estimados pelo método PCA-clássico no melhor caso, pior caso e média, resultado obtido utilizando os parâmetros de Carollo [4] e dados experimentais.

Fonte: O Autor, 2019.

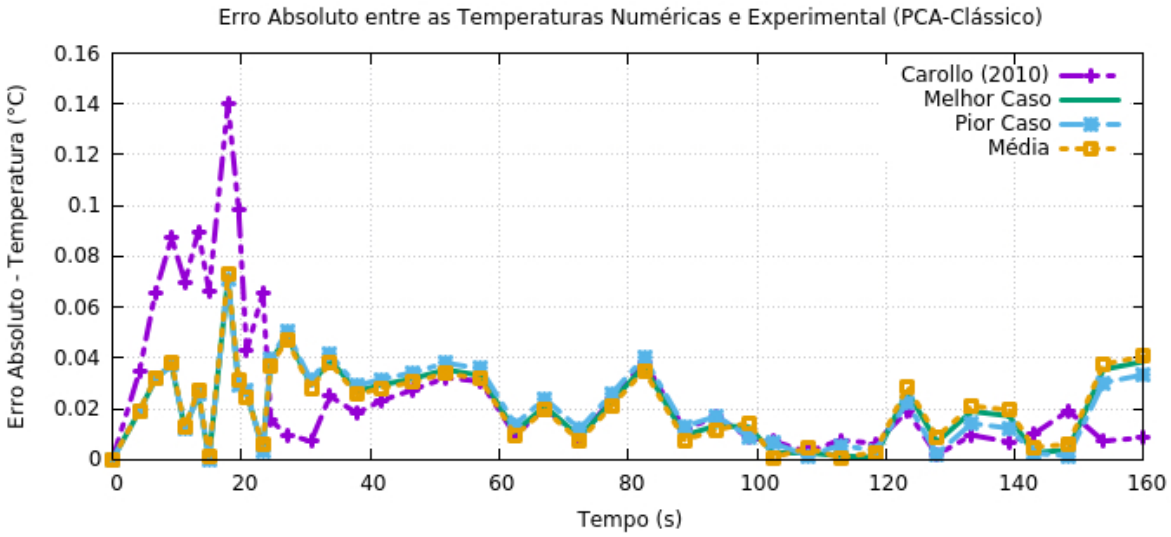


Figura 4.4: Erros absolutos obtidos com os parâmetros estimados pelo método PCA-clássico no melhor caso, pior caso e média e o erro obtido utilizando os parâmetros de Carollo [4].

Fonte: O Autor, 2019.

conforme resultados dos métodos LJ e PCA, nas versões clássicas.

Tabela 4.3: Distribuição de frequência do número de avaliações da função objetivo (NAF) para os métodos.

	Número de Ocorrências	
NAF	**LJ**	**PCA**
$1 \vdash 100$	0	2
$100 \vdash 200$	0	8
$200 \vdash 300$	2	18
$300 \vdash 400$	2	18
$400 \vdash 500$	7	27
$500 \vdash 600$	11	14
$600 \vdash 700$	26	6
$700 \vdash 800$	33	3
$800 \vdash 900$	16	2
$900 \vdash 1000$	2	2
$1000 \vdash 1100$	0	0
$1100 \vdash 1200$	0	0
$1200 \vdash 1300$	1	0
Total	100	100

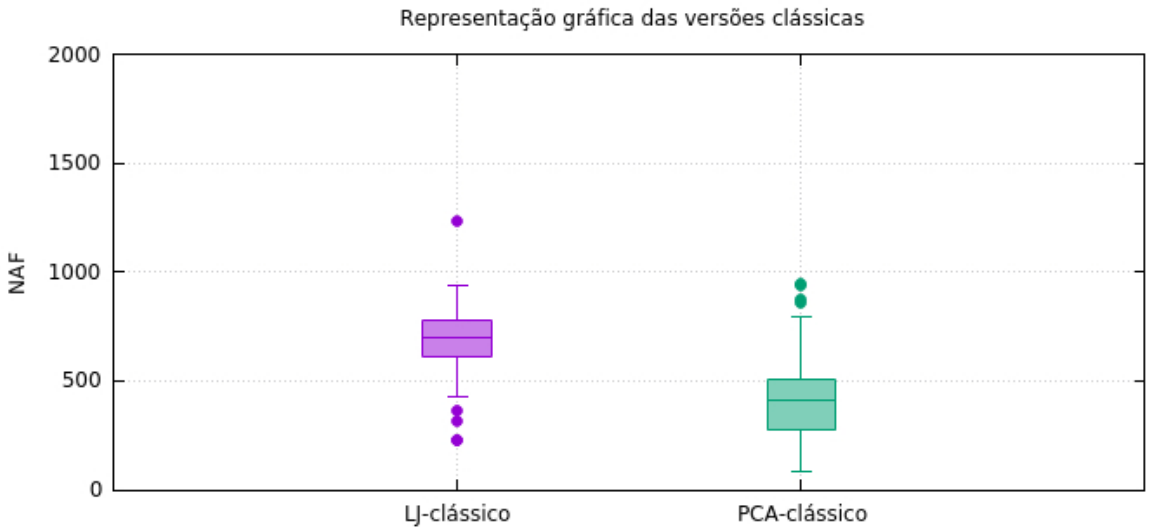


Figura 4.5: Distribuição dos dados para os métodos LJ e PCA nas versões clássicas. Fonte: O Autor, 2019.

Analisando a Figura 4.5, verifica-se que o PCA realizou um número de avaliações da função objetivo menor do que o LJ, os dados obtidos pelo PCA estão concentrados em torno da média e os parâmetros estimados estão próximos do parâmetro utilizado como base na abordagem inversa. Tanto o LJ quanto o PCA possuem *outliers*, que são valores considerados discrepantes em relação ao resultado e, nesse sentido, o LJ obteve o NAF igual a 227 para o melhor caso, sendo que esse valor é um *outlier* enquanto o PCA obteve um NAF igual a 85, e esse valor não é um *outlier*. Sob esse aspecto, o PCA na versão clássica foi melhor do que o LJ na versão clássica.

Com o intuito de verificar a influência da malha nos resultados obtidos, também foi analisada a estimativa dos parâmetros com a malha espacial contendo 10 nós ($\triangle x = 0,001209$) e malha temporal com 1600 nós ($\triangle t = 0,1$).

Na Tabela 4.4 são mostrados os resultados obtidos pela versão clássica do método LJ (Algoritmo 1), para $\triangle x = 0,001209$ e $\triangle t = 0,1$, sendo o número de avaliações da função objetivo igual a 858 para o pior caso e, NAF = 61, para o melhor caso. A média para o NAF foi igual a 601 e a mediana igual a 630. Já o desvio padrão resultou em NAF = 133. No que se refere ao tempo computacional, a versão clássica do LJ obteve 3681 *ms* no melhor tempo e 1972107 *ms* no pior, enquanto a média do tempo computacional foi de 55423 *ms* e a mediana igual a 37043 *ms*. Já o desvio padrão do tempo foi igual a 193828 *ms*.

Por outro lado, na Figura 4.6 são apresentados os perfis da temperatura experimental, das temperaturas numéricas para o melhor caso, pior caso e para a média, bem como o perfil com os mesmos parâmetros utilizados por Carollo [4] (denominado Carollo (2010)), tendo como base o método LJ na versão clássica (LJ-Clássico), com $\triangle x = 0,001209$ e $\triangle t = 0,1$. As temperaturas numéricas se aproximam da temperatura experimental ao longo dos 160 s da simulação. Enquanto na Figura 4.7 são apresentados os erros absolutos entre as temperaturas numéricas e a experimental para o melhor caso, pior caso, para a média e para os parâmetros utilizados por Carollo [4], levando em consideração esta configuração de malha.

Na versão clássica do LJ, tomando como referência o melhor resultado obtido (menor NAF), com $\triangle$ = 1600 nós, os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,029534 W/mK e 0,000777 $\times 10^6$ s/m e para o pior caso foram de 0,059084 W/mK e 0,002354 $\times 10^6$ s/m, respectivamente.

Para finalizar, na Tabela 4.5 são mostrados os resultados obtidos pela versão clássica do método PCA (Algoritmos 2 a 7), para $\triangle x = 0,001209$ e $\triangle t = 0,1$, sendo o número de

Tabela 4.4: Resultados obtidos pelo método LJ na versão clássica com $\triangle t = 1600$, para o problema de transferência de calor.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,277062	3869082,003235	0,02972	61	3681
Pior	11,943353	3878188,459226	0,029387	858	1972107
Média	11,610116	3866076,562418	0,028849	601	55423
Mediana	11,458929	3863377,083685	0,028092	630	37043
Desvio Padrão	0,266974	8431,902271	0,000669	133	193828

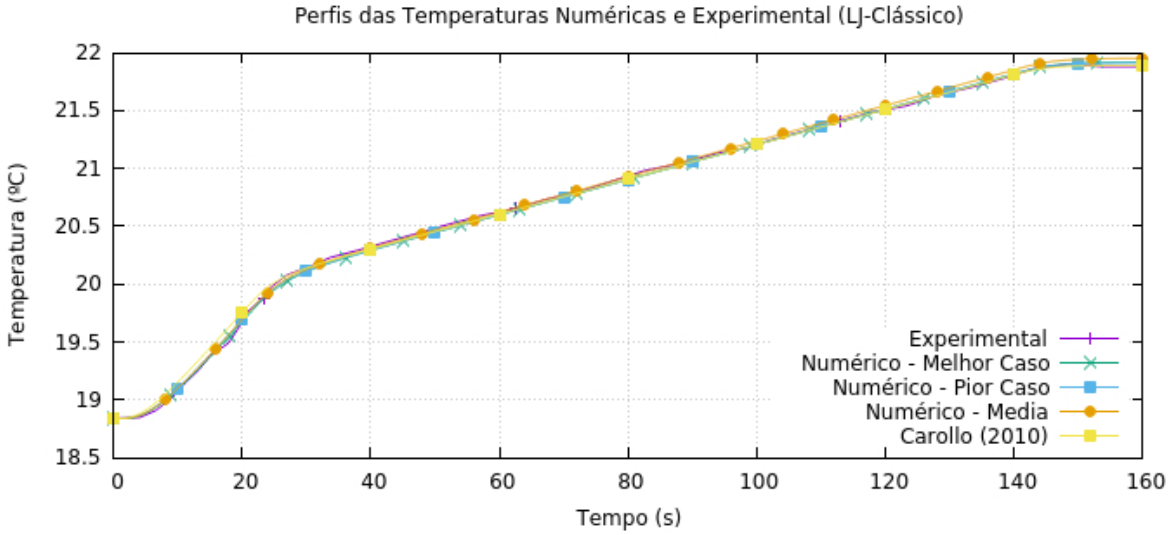


Figura 4.6: Resultados obtidos com os parâmetros estimados pelo método LJ clássico, com a formulação implícita com $\triangle t = 0,1$.
Fonte: O Autor, 2019.

avaliações da função objetivo igual a 968 para o pior caso e, NAF = 68, para o melhor caso. A média e a mediana para o NAF foram iguais a 399 e o desvio padrão resultou em NAF = 177. No que se refere ao tempo computacional, a versão clássica do PCA obteve 3431 ms no melhor tempo e 46481 ms no pior, enquanto a média do tempo computacional foi de 18431 ms e a mediana igual a 17760 ms. Concluindo, o desvio padrão do tempo foi igual a 8450 ms.

Para essa versão clássica do PCA, com $\triangle = 1600$ nós, os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,023511 W/mK e 0,004772 $\times 10^6$ s/m e para o pior caso foram de 0,009108 W/mK e 0,004262 $\times 10^6$ s/m, respectivamente, todos tendo como referência o melhor resultado obtido.

Já na Figura 4.8 são apresentados os perfis da temperatura experimental, das temperaturas numéricas para o melhor caso, pior caso e para a média, bem como o perfil com

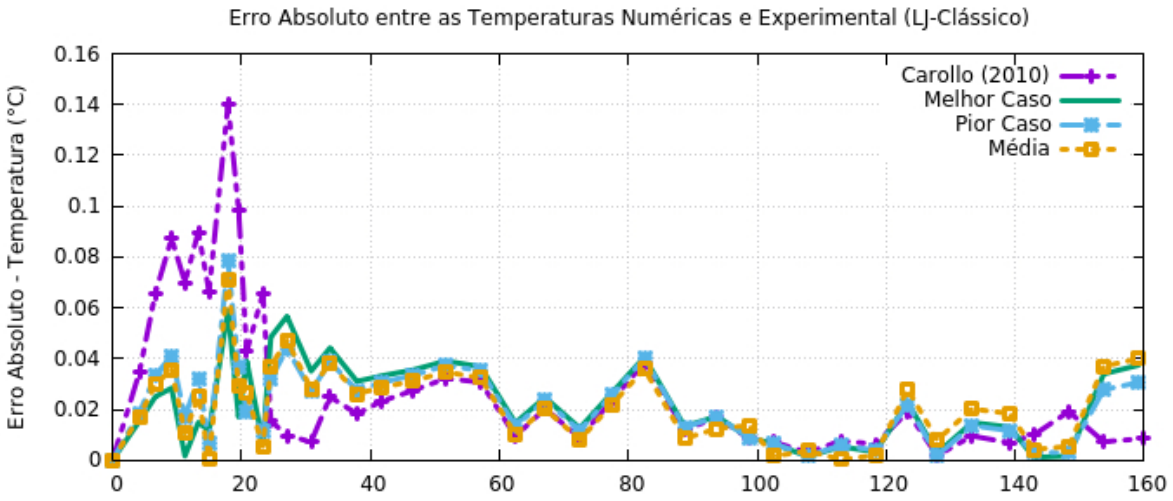


Figura 4.7: Erros absolutos obtidos com os parâmetros estimados pelo método LJ-clássico no melhor caso, pior caso e média e o erro obtido utilizando os parâmetros de Carollo [4] com $\triangle x = 0,001209$ e $\triangle t = 0,1$.

Fonte: O Autor, 2019.

os mesmos parâmetros utilizados por Carollo [4] (denominado Carollo (2010)), tomando como referência o método LJ na versão clássica (LJ-Clássico), com $\triangle x = 0,001209$ e $\triangle t = 0,1$. Durante o tempo de simulação igual a 160 s, as temperaturas numéricas se aproximam da temperatura experimental. Enquanto na Figura 4.9 são apresentados os erros absolutos entre as temperaturas numéricas e a experimental para o melhor caso, pior caso, para a média e para os parâmetros utilizados por Carollo [4], levando em consideração esta configuração de malha.

Tabela 4.5: Resultados obtidos pelo método PCA na versão clássica com $\triangle t = 1600$, para o problema de transferência de calor.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,401862	3847673,950382	0,029852	68	3431
Pior	11,505709	3864074,026721	0,027771	968	46481
Média	11,669931	3866033,682998	0,028552	399	18431
Mediana	11,744549	3865415,297716	0,028987	399	17760
Desvio Padrão	0,248216	8321,909514	0,000649	177	8450

Os resultados obtidos pelos métodos nas versões clássicas, utilizando a formulação implícita, com $\triangle x = 0,001209$ e $\triangle t = 0,1$ não foram muito melhores do que os resultados obtidos pelos mesmos métodos com $\triangle x = 0,001209$ e $\triangle t = 0,2$. Sendo assim, a configuração da malha temporal para $\triangle t = 0,2$, se apresentou mais eficiente, pois, o custo computacional reduziu devido ao menor número de nós.

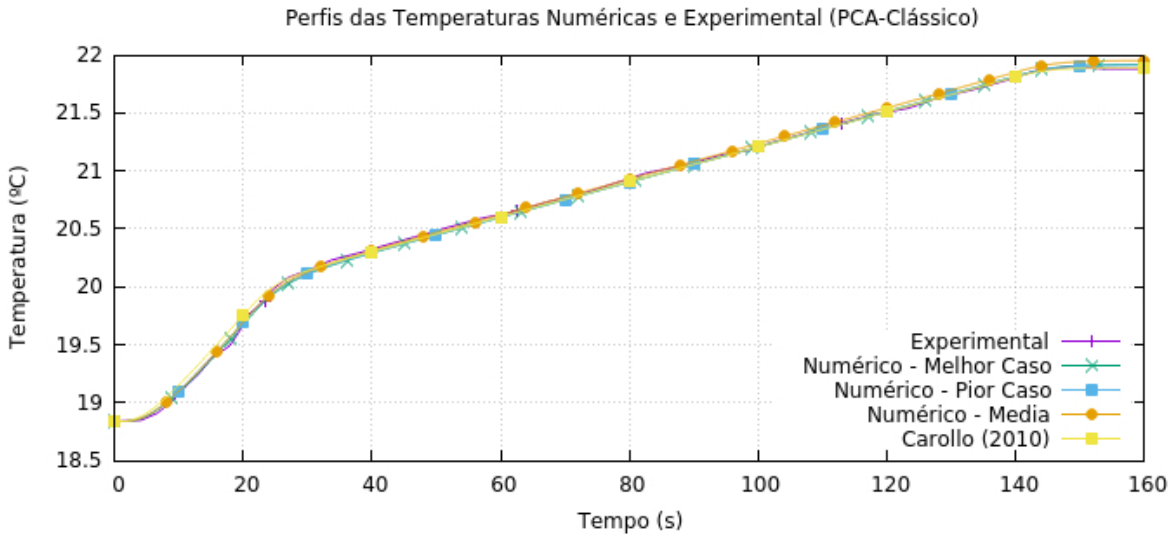


Figura 4.8: Resultados obtidos com os parâmetros obtidos pelo método PCA clássico com a formulação implícita com $\triangle t = 0,1$.

Fonte: O Autor, 2019.

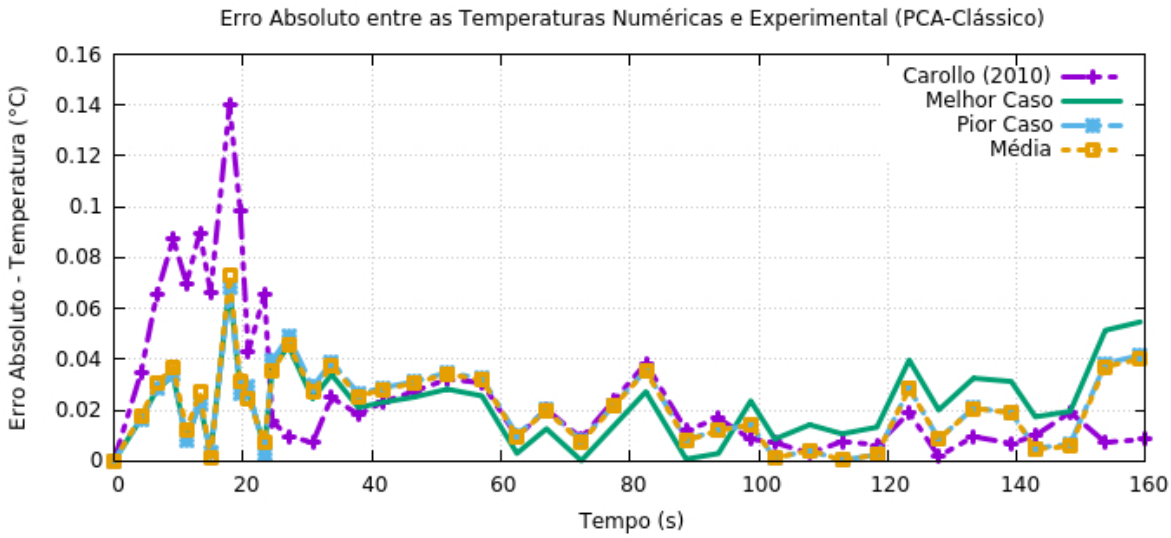


Figura 4.9: Erros absolutos obtidos com os parâmetros estimados pelo método PCA-clássico no melhor caso, pior caso e média e o erro obtido utilizando os parâmetros de Carollo [4] com $\triangle x = 0,001209$ e $\triangle t = 0,1$.

Fonte: O Autor, 2019.

Como os perfis de temperaturas e erros referentes à aplicação das modificações e hibridizações são muito próximos quando utilizada a malha computacional com 800 nós temporais, nas seções seguintes, seus respectivos gráficos serão omitidos para evitar resultados repetitivos.

4.2 Resultados dos Métodos Luus-Jaakola e Algoritmo de Colisão de Partículas Modificados

Assim como ocorrido com as versões clássicas, a condutividade térmica (λ) e a capacidade volumétrica (ρc_p) também foram estimadas utilizando as versões modificadas dos métodos LJ e PCA, as quais foram descritas na Seção 3.3.

A cada nova estimativa dos parâmetros, é feita uma combinação com a melhor estimativa até o momento, ou seja, se na iteração atual o ponto ótimo para uma função de duas variáveis for (x_1^*, x_2^*) e a nova for (x_1, x_2), então, com a combinação, se tem 3 estimativas a serem avaliadas, incluindo a que acabou de ser gerada: (x_1, x_2), (x_1^*, x_2) e (x_1, x_2^*), logo o número de loops internos tem que ser um terço da versão original. Os algoritmos foram configurados para realizarem um total de 100 execuções, com o objetivo de obter o melhor e o pior resultado, além dos cálculos da média, mediana e desvio padrão do NAF, bem como dos parâmetros a serem estimados e do valor da função objetivo.

Na Tabela 4.6 são apresentados os resultados do método LJ (Algoritmo 8), com alteração apenas na combinação de estimativas. O NAF foi de 66 no melhor resultado e NAF igual a 780 para o pior resultado. A média e a mediana foram, respectivamente, iguais a 495 e 487, enquanto o desvio padrão foi igual a 152. O melhor tempo computacional foi igual a 1464 *ms* e o pior tempo obtido foi igual a 16150 *ms*. A média e a mediana obtidas foram, respectivamente, iguais a 9962 *ms* e 9803 *ms*. O desvio padrão é igual a 3149 *ms*.

Nessa modificação LJ-M1, os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,018534 W/mK e 0,000431 $\times 10^6$ s/m e para o pior caso foram de 0,033600 W/mK e 0,001394 $\times 10^6$ s/m, respectivamente, em relação ao melhor valor obtido para os parâmetros (menor NAF).

Ao aplicar a técnica de combinação de estimativas, houve uma melhora no método no que se refere à redução do NAF, uma vez que, com essa técnica, os melhores elementos de cada vetor ($\boldsymbol{x}^{(k)}$ e $\boldsymbol{x}^*$) são aproveitados, ou seja, não são descartados, como acontece nas versões clássicas.

No método PCA (Algoritmo 9), Tabela 4.7, observa-se um NAF igual a 49, o que é uma solução satisfatória. O pior resultado tem NAF = 562, enquanto a média possui NAF = 332 e 354 para mediana. O desvio padrão obteve um NAF de 112. O melhor tempo foi igual a 2585 *ms* e, o pior, foi igual a 21518 *ms*. A média obtida foi igual a 11561 *ms*, enquanto a mediana foi igual a 11574 *ms*. O desvio padrão foi igual a 4210 *ms*. Com isso verifica-se que o tempo computacional do melhor resultado foi bem superior em relação

Tabela 4.6: Resultados obtidos pelo método LJ (LJ-M1), na versão clássica, com combinação de estimativas.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,348177	3862323,4761	0,028881	66	1464
Pior	11,729473	3867708,714182	0,028658	780	16150
Média	11,558500	3863989,728552	0,029271	495	9962
Mediana	11,814359	3872059,083399	0,029466	487	9803
Desvio Padrão	0,252759	7642,957763	0,000462	152	3149

a versão clássica, no entanto trouxe um resultado muito satisfatório, no que se refere ao NAF. Novamente, pode-se perceber a importância da combinação de estimativas, uma vez que os melhores resultados em cada vetor são aproveitados. Isso implica em dados mais concentrados em torno da média, tornando a modificação mais robusta. A malha considerada nessa modificação é a mesma considerada nas outras simulações, ou seja, malha espacial com 10 nós e malha temporal com 800 nós. Com os resultados obtidos, é possível perceber que o aumento da malha temporal, levará a um custo computacional bem maior e com pouco ganho de desempenho, no que se refere a redução do NAF, fazendo com que esta seja a configuração ideal para solucionar o problema proposto nesse trabalho.

Na modificação PCA-M1, os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,022859 W/mK e 0,000999 $\times 10^6$ s/m e para o pior caso foram de 0,036090 W/mK e 0,003974 $\times 10^6$ s/m, respectivamente.

Tabela 4.7: Resultados obtidos pelo método PCA (PCA-M1), na versão clássica, com combinação de estimativas.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,929746	3869631,894343	0,029433	49	2585
Pior	11,499198	3854252,397865	0,029204	562	21518
Média	11,657043	3865764,479263	0,029286	332	11561
Mediana	11,535788	3870681,158195	0,029096	354	11574
Desvio Padrão	0,224947	8135,798915	0,000461	112	4210

Após análise dos resultados, é possível concluir, que as modificações realizadas nos métodos LJ e PCA (respectivamente denominadas como LJ-M1 e PCA-M1) foram melhores do que as versões clássicas pois obtiveram o NAF menor em relação às mesmas. Além disso, ao se comparar as duas modificações, é possível perceber que o PCA-M1 teve vantagem em relação ao LJ-M1, por também possuir o NAF menor.

A partir da Tabela 4.8, os resultados foram obtidos com base na versão clássica dos métodos Luus-Jaakola e PCA, apresentando alterações no fator de contração dos intervalos de busca. Logo, na Tabela 4.8, o método Luus-Jaakola (Algoritmo 10) sofre alteração na linha 8 do Algoritmo 1, fazendo $r_i^{(k)} = \epsilon\ r_i^{(k-1)}$ e $\epsilon = \frac{10^0}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$. O número de avaliações da função objetivo foi de 259 no melhor resultado e NAF igual a 1846 para o pior resultado. A média e a mediana foram, respectivamente, iguais a 1263 e 1303. O desvio padrão foi igual a 289. O melhor tempo computacional foi de 5275 *ms*. O pior tempo obtido foi igual a 45993 *ms*. A média e a mediana obtidas foram, respectivamente, iguais a 27764 *ms* e 27560 *ms*. O desvio padrão foi igual a 7202 *ms*. Sendo assim, o pior resultado levou um tempo computacional muito superior em relação ao melhor resultado, embora o valor da função objetivo ($f(\lambda, \rho c_p)$) tenha sido um pouco melhor. Verifica-se que desvio padrão se aproxima de zero, permitindo concluir que os valores obtidos se dispersam pouco em relação a média.

Para a modificação LJ-M2, os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,019592 W/mK e 0,002156 $\times 10^6$ s/m e para o pior caso foram de 0,043721 W/mK e 0,002469 $\times 10^6$ s/m, respectivamente.

Tabela 4.8: Resultados obtidos pelo método LJ (LJ-M2), na versão clássica, com $r_i^{(k)} = \epsilon\ r_i^{(k-1)}$ e $\epsilon = \frac{10^0}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,348177	3855030,365734	0,028972	259	5275
Pior	11,844329	3864548,748359	0,029244	1846	45993
Média	11,570514	3863341,094623	0,029123	1263	27764
Mediana	11,634211	3866785,367805	0,029548	1303	27560
Desvio Padrão	0,212905	7230,650124	0,000464	289	7202

Já na Tabela 4.9, os resultados foram obtidos quando o método Luus-Jaakola sofre alteração na linha 8 do Algoritmo 1, ficando similar ao Algoritmo 10, com as linhas 8 e 9 alteradas, sendo $r_i^{(k)} = \epsilon\ r_i^{(k-1)}$ e $\epsilon = \frac{10^{-1}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$, o que gerou um pior resultado com NAF = 1546. Por outro lado, no melhor resultado, obteve-se um NAF igual a 360. O NAF da média foi de 1079, enquanto a mediana foi de 1110. No desvio padrão o NAF foi igual a 248. O melhor tempo obtido foi igual a 7197 *ms* e o pior, igual a 36139 *ms*. A média foi igual a 22955 *ms* e a mediana foi de 23567 *ms*. O desvio padrão foi igual a 5525 *ms*. Embora essa modificação no método tenha possibilitado resolver o problema de transferência de calor, não houve melhorias em relação ao NAF de nenhuma

das versões apresentadas até o momento.

No que se refere à modificação LJ-M3, os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,026462 W/mK e 0,000072 $\times 10^6$ s/m e para o pior caso foram de 0,012613 W/mK e 0,000513 $\times 10^6$ s/m, respectivamente.

Tabela 4.9: Resultados obtidos pelo método LJ (LJ-M3), na versão clássica, com $r_i^{(k)} = \epsilon\, r_i^{(k-1)}$ e $\epsilon = \frac{10^{-1}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,264162	3864397,751441	0,029596	360	7197
Pior	11,406241	3862413,979083	0,028631	1546	36139
Média	11,562230	3864118,778069	0,029130	1079	22955
Mediana	11,413858	3856996,447954	0,028881	1110	23567
Desvio Padrão	0,220473	6763,087747	0,000460	248	5525

Por outro lado, na Tabela 4.10 são apresentados os resultados onde o método Luus-Jaakola sofre alteração na linha 8 do Algoritmo 1, fincando similar ao Algoritmo 10, com as linhas 8 e 9 alteradas, sendo $r_i^{(k)} = \epsilon\, r_i^{(k-1)}$ e $\epsilon = \frac{10^{-3}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$, com NAF = 1130 para o pior resultado e melhor com NAF = 118. O NAF foi igual a 819 para média e NAF = 869 para mediana, enquanto o desvio padrão foi igual a 178. O melhor tempo obtido foi igual a 2532 ms e, o pior, 24234 ms. A média do tempo computacional foi igual a 17215 ms e a mediana obtida foi igual a 18102 ms. O desvio padrão foi igual a 3823 ms. Em relação à versão clássica, essa alteração no método trouxe bons resultados, diminuindo o NAF e o tempo computacional no melhor caso. A função objetivo, que representa a diferença entre as temperaturas experimental e numérica, também reduziu, obtendo um valor de 0,028696.

Nessa modificação (LJ-M4), os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,002071 W/mK e 0,001114 $\times 10^6$ s/m e para o pior caso foram de 0,004825 W/mK e 0,003732 $\times 10^6$ s/m, respectivamente.

Na sequência, na Tabela 4.11, são apresentados os resultados obtidos pelo método Luus-Jaakola, o qual foi implementado com alteração na linha 8 do Algoritmo 1, ficando, novamente, similar ao Algoritmo 10, com as linhas 8 e 9 alteradas, onde $r_i^{(k)} = \epsilon\, r_i^{(k-1)}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$. Nesse caso, tem-se NAF = 868 para pior caso e NAF = 344 como melhor resultado. A média e a mediana foram, respectivamente, iguais a 633 e 666, enquanto o desvio padrão foi de 122. O melhor tempo computacional foi igual a 7142 ms e, o pior, foi igual a 24557 ms. A média foi igual a 15057 ms, enquanto a mediana foi

Tabela 4.10: Resultados obtidos pelo método LJ (LJ-M4), na versão clássica, com $r_i^{(k)} = \epsilon\, r_i^{(k-1)}$ para $\epsilon = \frac{10^{-3}}{(max-min)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,626912	3860315,054935	0,028696	118	2532
Pior	11,683012	3874719,900652	0,029429	1130	24234
Média	11,602838	3864613,788329	0,029156	819	17215
Mediana	11,698668	3861761,333310	0,028957	869	18102
Desvio Padrão	0,223392	6826,673014	0,000427	178	3823

igual a 15029 *ms*. O desvio padrão foi igual a 3499 *ms*.

Os erros relativos da média para λ e ρc_p na modificação LJ-M5, foram respectivamente iguais a 0,034287 W/mK e 0,002603 $\times 10^6$ s/m e para o pior caso foram de 0,045969 W/mK e 0,003581 $\times 10^6$ s/m, respectivamente.

Ainda no que se refere aos resultados do método Luus-Jaakola com as modificações realizadas, nas Tabelas 4.12, 4.13, 4.14 e 4.15 são apresentados os resultados levando em consideração que $r_i^{(k)} = \epsilon\, r^{(k-1)}$ e, respectivamente, os seguintes valores para ϵ: $\epsilon = \frac{10^0}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $\epsilon = \frac{10^{-1}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $\epsilon = \frac{10^{-3}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $i = 1, 2$, tornando-se similar ao Algoritmo 11, com as respectivas alterações nas linhas 8 e 9. O NAF para os piores resultados foram, respectivamente, 1846, 1546, 1130 e 868. Já o NAF para os melhores resultados foram, respectivamente, 259, 360, 118 e 344. O NAF da média obtido pelos algoritmos foram 1263, 1079, 819 e 633, respectivamente. Para a mediana, o NAF foi 1303, 1110, 869 e 666, respectivamente. Os respectivos valores para o desvio padrão foram 289, 248, 178 e 122. Para finalizar, os melhores tempos computacionais foram iguais a 5326 *ms*, 8381 *ms*, 3043 *ms* e 7618 *ms*, respectivamente, seguido dos piores tempos obtidos, os quais foram, respectivamente, iguais a 41153 *ms*, 31376 *ms*, 27998 *ms* e 19562 *ms*. As médias foram iguais a 27381 *ms*, 22449 *ms*, 18835 *ms* e 13756 *ms*, respectivamente. Já as respectivas medianas obtidas foram iguais a 27402 *ms*, 23111 *ms*, 18992 *ms* e 13947 *ms*. Os desvios padrões foram iguais a 6557 *ms*, 5200 *ms*, 4690 *ms* e 2982 *ms*, respectivamente. Todas as modificações resolveram o problema de forma satisfatória, no entanto, a modificação que trouxe melhor resultado, em relação à versão clássica, foi a modificação do método apresentado na Tabela 4.14, cujo NAF foi igual a 118. Tanto o tempo computacional quanto a função objetivo também foram melhores em relação à versão clássica do método. Embora o desvio padrão nessa modificação tenha se aproximado de zero, o desvio padrão na versão clássica foi menor. A modificação apresentada pela Tabela 4.14 ainda foi melhor do que as outras modificações apresentadas

Tabela 4.11: Resultados obtidos pelo método LJ (LJ-M5), na versão clássica, com $r_i^{(k)} = \epsilon\, r^{(k-1)}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,233181	3854816,478295	0,029296	344	7142
Pior	11,749554	3868621,283466	0,028727	868	24557
Média	11,618337	3864850,898142	0,029185	633	15057
Mediana	11,635201	3859425,160977	0,029091	666	15029
Desvio Padrão	0,219574	6931,028080	0,000464	122	3499

Tabela 4.12: Resultados obtidos pelo método LJ (LJ-M6), na versão clássica, com $r_i^{(k)} = \epsilon\, r_i^{(k-1)}$ e $\epsilon = \frac{10^{0}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,348177	3855030,365734	0,028972	259	5326
Pior	11,844329	3864548,748359	0,029244	1846	41153
Média	11,570514	3863341,094623	0,029123	1263	27381
Mediana	11,634211	3866785,367805	0,029548	1303	27402
Desvio Padrão	0,212905	7230,650124	0,000464	289	6557

do método LJ, até o presente momento.

Os erros relativos da média para λ e ρc_p na modificação LJ-M6, foram respectivamente iguais a 0,019592 W/mK e 0,002156 $\times 10^6$ s/m e para o pior caso foram de 0,043721 W/mK e 0,002469 $\times 10^6$ s/m, respectivamente. Já na modificação LJ-M7, os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,026462 W/mK e 0,000072 $\times 10^6$ s/m e para o pior caso foram de 0,012613 W/mK e 0,000513 $\times 10^6$ s/m, respectivamente. Enquanto na modificação LJ-M8, os erros relativos da média para λ e ρc_p foram respectivamente iguais a 0,002071 W/mK e 0,001114 $\times 10^6$ s/m e para o pior caso foram de 0,004825 W/mK e 0,003732 $\times 10^6$ s/m, respectivamente. Por fim, a modificação LJ-M9, obteve os erros relativos da média para λ e ρc_p iguais a 0,034287 W/mK e 0,002603 $\times 10^6$ s/m, respectivamente e para o pior caso iguais a 0,045969 W/mK e 0,003581 $\times 10^6$ s/m, respectivamente.

Em seguida, na Tabela 4.16, são apresentados os resultados obtidos por meio da versão modificada do método LJ (Algoritmo 12), tomando-se $r_i^{(k)} = \epsilon\, r_i^{(k-1)}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$, em que a combinação das estimativas é utilizada para obtenção de λ e ρc_p, onde o resultado foi dado com o NAF = 726 para o pior caso, enquanto o melhor resultado foi obtido com NAF = 66, respeitando o critério de parada estabelecido *a priori*. O NAF

Tabela 4.13: Resultados obtidos pelo método LJ (LJ-M7), na versão clássica, com $r_i^{(k)} = \epsilon\, r^{(k-1)}$ e $\epsilon = \frac{10^{-1}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,264162	3864397,751441	0,029596	360	8381
Pior	11,406241	3862413,979083	0,028631	1546	31376
Média	11,562230	3864118,778069	0,029130	1079	22449
Mediana	11,413858	3856996,447954	0,028881	1110	23111
Desvio Padrão	0,220473	6763,087747	0,000460	248	5200

Tabela 4.14: Resultados obtidos pelo método LJ (LJ-M8), na versão clássica, com $r_i^{(k)} = \epsilon\, r^{(k-1)}$ e $\epsilon = \frac{10^{-3}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,626912	3860315,054935	0,028696	118	3043
Pior	11,683012	3874719,900652	0,029429	1130	27998
Média	11,602838	3864613,788329	0,029156	819	18835
Mediana	11,698668	3861761,333310	0,028957	869	18992
Desvio Padrão	0,223392	6826,673014	0,000427	178	4690

Tabela 4.15: Resultados obtidos pelo algoritmo LJ (LJ-M9), na versão clássica, com $r_i^{(k)} = \epsilon\, r^{(k-1)}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,233181	3854816,478295	0,029296	344	7618
Pior	11,749554	3868621,283466	0,028727	868	19562
Média	11,618337	3864850,898142	0,029185	633	13756
Mediana	11,635201	3859425,160977	0,029091	666	13947
Desvio Padrão	0,219574	6931,028080	0,000464	122	2982

para a média foi igual a 487 e, para a mediana, 514. O desvio padrão resultou em NAF = 22422. No que se refere ao tempo computacional, nesta versão, o mesmo foi de 1455 *ms*, no melhor caso e, 18826 *ms* no pior. A média e a mediana do tempo foram iguais a 10721 *ms* e 11192 *ms*, respectivamente. Por fim, o desvio padrão foi igual a 3538 *ms*. Em relação ao NAF, essa modificação se apresentou melhor do que a versão clássica e demais modificações realizadas e apresentadas até o momento.

No que se refere à modificação LJ-M10, obteve os erros relativos da média para λ e ρc_p iguais a 0,013485 W/mK e 0,001284 $\times 10^6$ s/m, respectivamente e para o pior caso iguais a 0,032148 W/mK e 0,001467 $\times 10^6$ s/m, respectivamente.

Tabela 4.16: Resultados obtidos pelo método LJ (LJ-M10) na versão modificada através de combinação da nova com a melhor estimativa e $r_i^{(k)} = \epsilon\ r_i^{(k-1)}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,731186	3859365,042997	0,029284	66	1455
Pior	11,354052	3853704,615877	0,029123	726	18826
Média	11,572992	3864318,761701	0,029325	487	10721
Mediana	11,678818	3860041,275289	0,028973	514	11192
Desvio Padrão	0,252705	8456,267505	0,000000	22422	3538

Finalizando os resultados no que refere ao método Luus-Jaakola, na Tabela 4.17, são mostrados os valores dos parâmetros obtidos com o método LJ (Algoritmo 13), o qual apresentou o melhor desempenho, uma vez que o NAF foi de 14 e $f(\lambda, \rho c_p) = 0,028974$. O pior resultado obteve NAF = 651. O NAF da média foi igual a 375 e da mediana foi 404, enquanto o desvio padrão resultou no NAF igual a 170. No que se refere ao tempo computacional, o melhor tempo foi igual a 310 *ms* e, o pior, foi igual a 14896 *ms*. A média e a mediana foram iguais a 8273 *ms* e 9102 *ms*, respectivamente. Já o desvio padrão foi igual a 3797 *ms*. Para alcançar esse resultado configurou-se $r_i^{(k)} = \epsilon\ r^{(k-1)}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $i = 1, 2$. Essa modificação implicou em um resultado melhor do que todas as versões modificadas e melhor até mesmo do que a versão clássica. Um dos motivos foi o fato, já apresentado, da combinação de estimativas aproveitar os melhores resultados do vetor "nova estimativa" $(\boldsymbol{x}_j^{(k)})$ com os melhores resultados do vetor "solução ótima" $(\boldsymbol{x}^*)$. Outro motivo é a configuração realizada no fator de contração ϵ. A junção dessas duas técnicas resultou nos melhores resultados apresentados nesse trabalho.

A modificação LJ-M11, obteve os erros relativos da média para λ e ρc_p iguais a 0,022513 W/mK e 0,000102 $\times 10^6$ s/m, respectivamente e para o pior caso iguais a

0,051021 W/mK e 0,001123 $\times 10^6$ s/m, respectivamente.

Tabela 4.17: Resultados obtidos pelo método LJ (LJ-M11) na versão modificada através de combinação da nova com a melhor estimativa e $r_i^{(k)} = \epsilon\ r^{(k-1)}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,340849	3863403,733211	0,028974	14	310
Pior	11,919467	3867743,154374	0,029446	651	14986
Média	11,596169	3863798,704388	0,029221	375	8273
Mediana	11,845996	3868875,902349	0,029074	404	9102
Desvio Padrão	0,239353	7377,680245	0,000443	170	3797

No que se refere às modificações realizadas no Algoritmo de Colisão de Partículas, na Tabela 4.18 é exibido o resultado obtido pelo método PCA (Algoritmos 2 a 7) com alteração na função *Perturbation()* (Algoritmo 14), onde $NewConfig_i = \frac{1}{4}((3\ OldConfig_i) + (min_i + rand(max_i - min_i)))$, $i = 1, 2$. O pior NAF foi de 2517, e o melhor, NAF = 78, mostrando uma melhora em relação ao PCA na versão clássica. O NAF da média foi de 562 e o NAF da mediana foi 454. Já o desvio padrão apresentou um NAF de 378. O tempo computacional, nesta versão do algoritmo, foi igual a 451 ms para o melhor tempo computacional e, 53299 ms para o pior. A média do tempo foi igual a 9652 ms e a mediana foi igual a 7735 ms. O desvio padrão obtido foi igual 7842 ms. A alteração no PCA se refere a linha 2 do Algoritmo 3. Essa modificação no método PCA trouxe resultado melhor do que a versão clássica do LJ, pois obteve um NAF igual a 78. Embora o tempo computacional e a função objetivo não tenham sido melhores, ficaram próximos da versão clássica. Em relação à modificação PCA-M1, essa modificação não foi melhor. Em contrapartida, comparando com a versão clássica do LJ, essa modificação melhorou de forma considerável os resultados no que se refere ao NAF.

Com relação à modificação PCA-M2, os erros relativos da média para λ e ρc_p foram iguais a 0,010503 W/mK e 0,001593 $\times 10^6$ s/m, respectivamente e para o pior caso iguais a 0,023038 W/mK e 0,000091 $\times 10^6$ s/m, respectivamente.

Para analisar a distribuição de frequência dos resultados obtidos com as modificações realizadas no método LJ, denominam-se tais modificações, referente às Tabelas 4.6, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, 4.15, 4.16 e 4.17, como LJ-M1, LJ-M2, LJ-M3, LJ-M4, LJ-M5, LJ-M6, LJ-M7, LJ-M8, LJ-M9, LJ-M10 e LJ-M11, respectivamente. Na Tabela 4.19 são descritas as distribuições de frequências, enquanto na Figura 4.10, é apresentada a distribuição dos dados referente a esses métodos modificados no boxplot.

Tabela 4.18: Resultados obtidos pelo método PCA (PCA-M2), na versão clássica, com alteração na função *Perturbation()*, fazendo $NewConfig_i = \frac{1}{4}((3\ OldConfig_i) + (min_i + rand(max_i - min_i)))$, $i = 1, 2$.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,780669	3859335,645697	0,029599	78	451
Pior	11,509271	3859685,673621	0,028534	2517	53299
Média	11,656941	3865485,351721	0,029253	562	9652
Mediana	11,598103	3869318,709255	0,028949	454	7735
Desvio Padrão	0,222345	6984,029204	0,000493	378	7842

Tabela 4.19: Distribuição de frequência do método LJ nas versões modificadas.

	Número de Ocorrências											
NAF	**LJ-Clássico**	**LJ-M1**	**LJ-M2**	**LJ-M3**	**LJ-M4**	**LJ-M5**	**LJ-M6**	**LJ-M7**	**LJ-M8**	**LJ-M9**	**LJ-M10**	**LJ-M11**
1 ⊢ 100	0	3	0	0	0	0	0	0	0	0	2	9
100 ⊢ 200	0	0	0	0	1	0	0	0	1	0	2	9
200 ⊢ 300	2	5	1	0	1	0	1	0	1	0	8	14
300 ⊢ 400	2	18	1	1	1	1	1	1	0	5	16	17
400 ⊢ 500	7	26	0	2	0	0	0	2	3	13	19	25
500 ⊢ 600	11	21	1	3	3	1	1	3	6	12	30	18
600 ⊢ 700	26	20	1	3	6	1	1	3	10	22	18	8
700 ⊢ 800	33	7	2	4	10	2	2	4	18	23	5	0
800 ⊢ 900	16	0	2	5	18	2	2	5	23	30	0	0
900 ⊢ 1000	2	0	7	15	23	7	7	15	25	7	0	0
1000 ⊢ 1100	0	0	11	15	25	11	11	15	12	0	0	0
1100 ⊢ 1200	0	0	11	16	12	11	11	16	1	0	0	0
1200 ⊢ 1300	1	0	12	16	1	12	12	16	0	0	0	0
1300 ⊢ 1400	0	0	16	14	0	16	16	14	0	0	0	0
1400 ⊢ 1500	0	0	13	5	0	13	13	5	0	0	0	0
1500 ⊢ 1600	0	0	13	1	0	13	13	1	0	0	0	0
1600 ⊢ 1700	0	0	6	0	0	6	6	0	0	0	0	0
1700 ⊢ 1800	0	0	2	0	0	2	2	0	0	0	0	0
1800 ⊢ 1900	0	0	1	0	0	1	1	0	0	0	0	0
Total	100	100	100	100	100	100	100	100	100	100	100	100

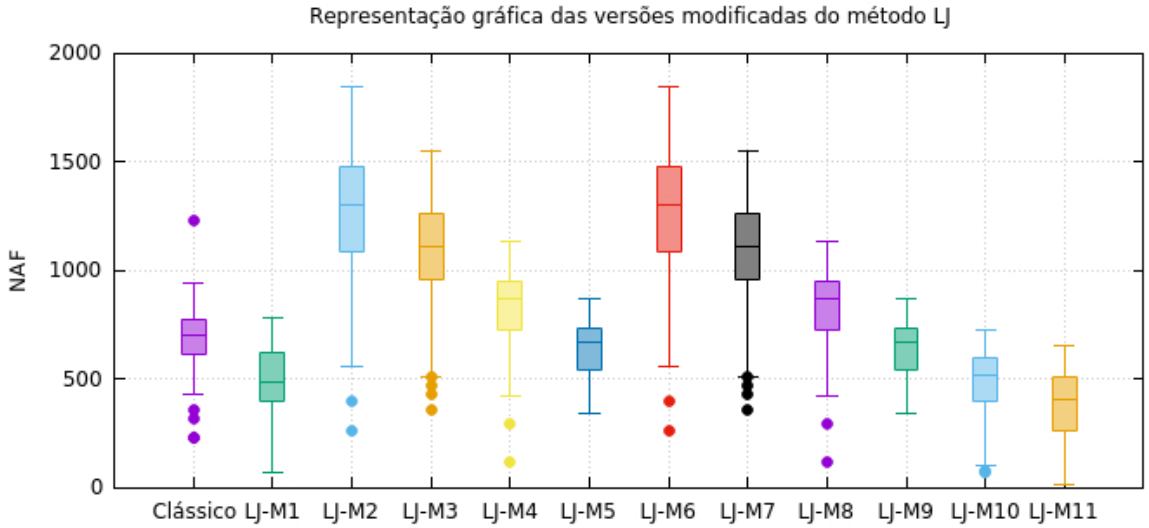


Figura 4.10: Distribuição dos dados para o método LJ nas versões modificadas.
Fonte: O Autor, 2019.

Analisando os resultados apresentados na Figura 4.10, verifica-se que a modificação LJ-M1, que utiliza a técnica de combinação de estimativas apresentou ótimos resultados quando se compara o número de avaliações da função objetivo e a concentração dos dados em torno da média. Isso aconteceu porque a combinação de estimativas aproveitou os melhores resultados armazenados em cada vetor. As modificações LJ-M2 à LJ-M5, mostraram que, quanto maior a potência do numerador no fator de contração ϵ, pior o foi resultado. Sendo assim, quando a potência foi igual a 10^{-6} (LJ-M5), o resultado foi mais satisfatório, devido à boa concentração dos dados em torno da média. O número de avaliações da modificação denominada LJ-M5 foi melhor em relação à versão clássica do LJ. A mesma análise vale para as modificações LJ-M6 à LJ-M9. A versão LJ-M10 ($r_i^{(k)} = \epsilon\ r^{(k-1)}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{1.0}{n_{out}}}$, $i = 1, 2$), embora haja um *outlier*, o mesmo ficou muito próximo à cauda inferior. Os dados estão concentrados em torno da média, o que mostra que o desvio padrão dos dados obtidos foi pequeno. Já o LJ-M11 ($r_i^{(k)} = \epsilon r^{(k-1)}$ e $\epsilon = \frac{10^{-6}}{(max_i - min_i)}^{\frac{k}{n_{out}}}$, $i = 1, 2$) apresentou o melhor resultado nesse trabalho, tanto em número de avaliação da função objetivo quanto em concentração de dados. Nessa modificação proposta, nenhum dado é um *outlier*.

As modificações realizadas no PCA, cujos resultados constam nas Tabelas 4.7 e 4.18, são denominadas como PCA-M1 e PCA-M2, respectivamente. Na Figura 4.11, é apresentada a distribuição dos dados e, na Tabela 4.20, é apresentada a distribuição de frequências dos resultados das referidas modificações. Os intervalos sem frequência foram suprimidos.

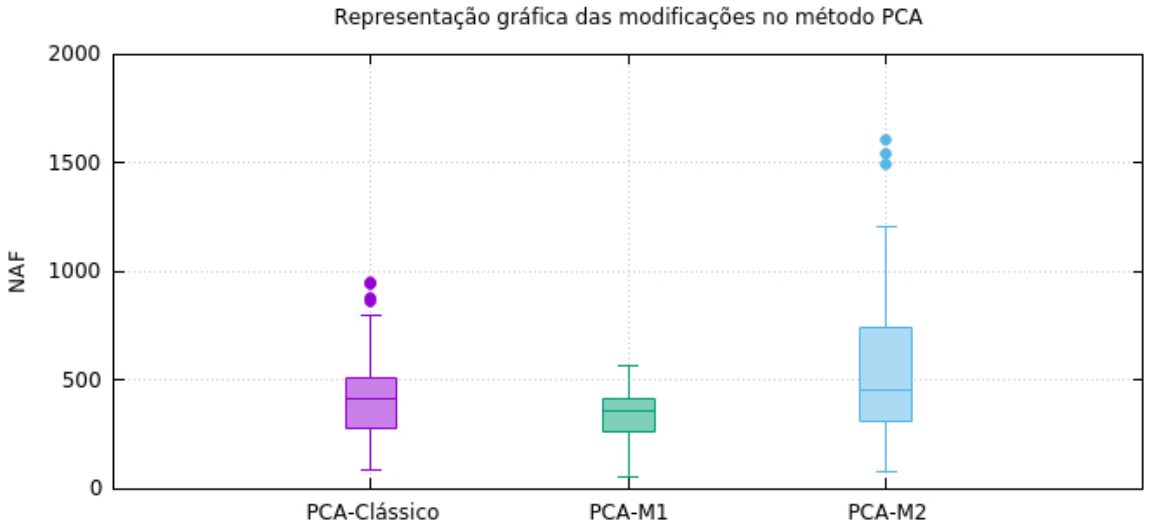


Figura 4.11: Distribuição dos dados para o método PCA nas versões modificadas.
Fonte: O Autor, 2019.

Ao analisar os resultados da Figura 4.11, constata-se que apesar de ambas as modifi-

Tabela 4.20: Distribuição de frequência do método PCA nas versões modificadas.

	Número de Ocorrências		
NAF	**PCA-Clássico**	**PCA-M1**	**PCA-M2**
$1 \vdash 100$	2	4	5
$100 \vdash 200$	8	10	6
$200 \vdash 300$	18	23	13
$300 \vdash 400$	18	31	14
$400 \vdash 500$	27	28	16
$500 \vdash 600$	14	4	10
$600 \vdash 700$	6	0	6
$700 \vdash 800$	3	0	9
$800 \vdash 900$	2	0	7
$900 \vdash 1000$	2	0	4
$1000 \vdash 1100$	0	0	4
$1100 \vdash 1200$	0	0	1
$1200 \vdash 1300$	0	0	1
$1300 \vdash 1400$	0	0	0
$1400 \vdash 1500$	0	0	1
$1500 \vdash 1600$	0	0	1
$1600 \vdash 1700$	0	0	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2500 \vdash 2600$	0	0	1
Total	100	100	100

cações terem sido melhores do que a versão clássica no que se refere ao menor número de avaliações da função objetivo, o PCA-M2 teve uma dispersão em relação aos dados, mostrando que não pode ser considerado como melhor resultado em relação à versão clássica do PCA. O PCA-M1, além de obter um número menor de avaliações da função objetivo, mostra que os resultados obtidos estão concentrados em torno da média, apresentando resultados melhores em relação à versão clássica.

4.3 Resultados da Hibridização Entre os Métodos Luus-Jaakola e Algoritmo de Colisão de Partículas Modificados

Por fim, nessa seção são apresentados os resultados da estimativa dos parâmetros λ e ρc_p utilizando as hibridizações realizadas envolvendo os métodos LJ e PCA, cujos resultados são apresentados nas Tabelas 4.21 a 4.27.

Na Tabela 4.21 é apresentado o resultado obtido pelo método LJ hibridizado com o método PCA. Nessa primeira hibridização, o algoritmo Luus-Jaakola chama a função *Scattering()*, caso a nova estimativa não seja melhor do que a estimativa anterior. Nos Algoritmos 15 e 16 são mostrados os pseudocódigos com as alterações, onde o NAF para o pior resultado foi igual a 2058 e o NAF para o melhor resultado igual a 126. A média e a mediana do NAF, respectivamente, foram iguais a 974 e 985. Já o desvio padrão do NAF foi igual a 304. Além disso, o melhor tempo foi igual a 2874 *ms* e o pior tempo igual a 44042 *ms*, seguido da média igual a 22362 *ms*, e da mediana obtida, igual a 22765 *ms*. O desvio padrão foi igual a 6603 *ms*. Essa hibridização resultou em um NAF menor do que na versão clássica do LJ, bem como a maioria das versões modificadas do mesmo.

Para a modificação LJ-H1, tomando como referência o melhor resultado encontrado, os erros relativos da média para λ e ρc_p foram iguais a 0,029826 W/mK e 0,002765 $\times 10^6$ s/m, respectivamente e para o pior caso iguais a 0,020200 W/mK e 0,000520 $\times 10^6$ s/m, respectivamente.

Tabela 4.21: Resultados obtidos pelo método LJ (LJ-H1) e alteração na função *Scattering*.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,210933	3852556,663795	0,029292	126	2874
Pior	11,43739	3850554,596136	0,029663	2058	44042
Média	11,545312	3863210,287826	0,029179	974	22362
Mediana	11,775742	3862532,170873	0,029458	985	22765
Desvio Padrão	0,246520	7228,553126	0,000471	304	6603

Por outro lado, a segunda versão hibridizada dos métodos LJ e PCA, descrita nos Algoritmos 17 a 19, obteve um NAF = 15372 no pior caso e, NAF = 288 no melhor caso, conforme Tabela 4.22. Embora o desvio padrão para $f(\lambda, \rho c_p)$ tenha sido 0,032029, ainda obteve-se um bom resultado por estar próximo de zero. Para a média, o NAF foi de 1186 e, para a mediana, NAF foi de 964. Para o desvio padrão, o NAF foi igual a 1535. Nesta versão, o melhor tempo computacional foi de 6877 *ms* e, o pior tempo, foi de 322463 *ms*. A média do tempo foi de 26112 *ms* enquanto a mediana foi igual a 22305 *ms*. O desvio padrão foi igual a 31623 *ms*. Essa versão não apresentou melhorias em relação ao NAF obtido nas versões clássicas dos métodos LJ e PCA, além de encontrar a solução em um tempo considerável.

Na modificação LJ-H2 os erros relativos da média para λ e ρc_p foram iguais a 0,029035 W/mK e 0,001407 $\times 10^6$ s/m, respectivamente e, para o pior caso, iguais a 0,032167 W/mK e 0,029159 $\times 10^6$ s/m, respectivamente.

Tabela 4.22: Resultados obtidos pelo método LJ (LJ-H2) com nova alteração na função *Scattering()*.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,942933	3870628,903333	0,028859	288	6877
Pior	11,558768	3983491,419221	0,34942	15372	322463
Média	11,596169	3865182,999233	0,032362	1186	26112
Mediana	11,592332	3863294,776167	0,029011	964	22305
Desvio Padrão	0,239353	13900,234578	0,032029	1535	31623

No Algoritmo 21, é feita uma alteração na função *Perturbation()*, que é chamada pelo método Luus-Jaakola, também alterado, conforme Algoritmo 20. Na Tabela 4.23, é apresentado o pior caso com NAF = 5000 e melhor caso com NAF = 42, representando uma redução significativa no NAF para o melhor caso em relação à versões clássicas dos métodos LJ e PCA. O pior caso não atendeu ao critério de parada estipulado para a função objetivo, pois $f(\lambda, \rho c_p) = 0{,}559937$ e, por isso, o desvio padrão para $f(\lambda, \rho c_p)$, dado por 0,077332, indica uma maior dispersão dos parâmetros obtidos nas execuções do problema sob análise. Para a média, o NAF foi de 613 e para mediana, tem-se NAF igual a 421, enquanto que, para o desvio padrão, o NAF foi igual a 919. O melhor tempo computacional, nesta versão, foi igual a 840 *ms* enquanto o pior tempo foi igual 101777 *ms*. A média, mediana e desvio padrão foram, respectivamente, iguais a 12446 *ms*, 8479 *ms* e 18581 *ms*. Cabe ressaltar que os resultados envolvendo o Algoritmo 14 também levaram aos mesmos valores apresentados na Tabela 4.23, por isso os resultados foram omitidos. Nessa modificação LJ-H3 os erros relativos da média para λ e ρc_p foram iguais a 0,050167 W/mK e 0,000626 $\times 10^6$ s/m, respectivamente e para o pior caso iguais a 2,053867 W/mK e 0,036974 $\times 10^6$ s/m, respectivamente.

Tabela 4.23: Resultados obtidos pelo método LJ (LJ-H3) hibridizado com função *Perturbation()* do PCA alterada.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,62586	3865031,654401	0,028462	42	840
Pior	35,503825	4007938,987229	0,559937	5000	101777
Média	12,209096	3867451,885590	0,043847	613	12446
Mediana	11,504567	3869892,603819	0,029775	421	8479
Desvio Padrão	3,386691	25772,034315	0,077332	919	18581

Na sequência, os resultados exibidos na Tabela 4.24 são referentes aos Algoritmos 20 e 22, onde o pior resultado obteve o NAF igual a 1338, enquanto para o melhor resultado,

obteve-se NAF igual a 221. O NAF para a média e mediana foram, respectivamente, 601 e 930, enquanto o NAF referente ao desvio padrão foi igual a 164. O melhor tempo obtido foi igual a 5199 *ms* enquanto o pior foi igual a 31018 *ms*. Já a média obtida foi igual a 13409 *ms* e a mediana foi igual a 13704 *ms*. O desvio padrão foi igual a 3806 *ms*. Ao comparar esse resultado com a versão clássica do LJ, é possível notar que em relação ao NAF, essa hibridização trouxe melhorias, já em relação à versão clássica do PCA, essa hibridização possui um NAF maior.

No que tange à modificação LJ-H4 os erros relativos da média para λ e ρc_p foram iguais a 0,028888 W/mK e 0,002928 $\times 10^6$ s/m, respectivamente e para o pior caso iguais a 0,058820 W/mK e 0,005974 $\times 10^6$ s/m, respectivamente.

Tabela 4.24: Resultados obtidos pelo método LJ (LJ-H4) hibridizado com função *Perturbation()* do PCA alterada.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,278285	3852269,776411	0,029398	221	5199
Pior	11,941679	3875284,174299	0,029729	1338	31018
Média	11,604089	3863548,787483	0,029284	601	13409
Mediana	11,624342	3860086,794465	0,029372	930	13704
Desvio Padrão	0,244515	8325,283538	0,000444	164	3806

Na versão hibridizada do método PCA, uma nova estimativa é gerada, $r_i^{(k)} = r^{(k-1)}$, $i = 1, 2$, além de se realizar uma redução no intervalo de busca para $r_i^{(k)} = (1 - \epsilon) r_i^{(k-1)}$ $i = 1, 2$, da mesma forma que ocorre no LJ e na função *Perturbation()*, $\text{NewConfig}_i = \text{OldConfig}_i + randr_i^{(k-1)}$, $i = 1, 2$, conforme Algoritmo 24. A outras funções utilizadas no método PCA estão de acordo com os Algoritmos 4, 5 e 7. Diante dessas alterações no método, na Tabela 4.25 são apresentados os dados resultantes dessa modificação, onde para o pior e melhor caso, os NAFs foram iguais a 4040 e 88, respectivamente. A média e a mediana foram iguais a 608 e 364, respectivamente. Já o NAF do desvio padrão foi igual a 762. O melhor e pior tempo obtidos foram, respectivamente, iguais a 2342 *ms* e 112938 *ms*. A média e a mediana foram iguais a 15679 *ms* e 9309 *ms*, respectivamente. Já o desvio padrão obtido foi igual a 19812 *ms*. Comparando com a versão clássica do LJ, nota-se uma melhora significativa no NAF, porém comparando com a versão clássica do PCA, essa hibridização não trouxe melhorias, mas se aproximou muito do melhor resultado, embora o tempo computacional não tenha se aproximado.

Ao analisar a modificação PCA-H1, verifica-se que os erros relativos da média para λ e ρc_p foram iguais a 0,001315 W/mK e 0,000169 $\times 10^6$ s/m, respectivamente e para o

pior caso iguais a 0,012762 W/mK e 0,003264 $\times 10^6$ s/m, respectivamente.

Tabela 4.25: Resultados obtidos pelo método PCA (PCA-H1) hibridizado com LJ e alteração na função *Perturbation()*.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,625758	3864904,479699	0,028461	88	2342
Pior	11,774125	3877520,854564	0,029889	4040	112938
Média	11,610465	3864250,296110	0,029299	608	15679
Mediana	11828833	3872713,046336	0,029201	364	9309
Desvio Padrão	0,249128	7473,559426	0,000509	762	19812

Para finalizar, na Tabela 4.26 são mostrados os resultados obtidos após a execução do método PCA hibridizado com o LJ, onde a função *Perturbation()* é similar à versão apresentada anteriormente, porém $\text{NewConfig}_i = \text{OldConfig}_i + rand\, r_i^{(k-1)}$ e $\text{NewConfig}_i = \frac{1}{4}((3\ \text{OldConfig}_i)+(\text{NewConfig}_i))$, $i = 1, 2$, tendo como enfoque a manutenção das características dos dois métodos. Para essa modificação realizada, o NAF do pior e melhor foram iguais a 4463 e 43, respectivamente. O NAF da média foi igual a 1160, enquanto o NAF da mediana foi igual a 754, seguido do desvio padrão, o qual foi igual a 1119. O melhor tempo computacional foi igual a 1068 ms e o pior foi igual a 112890 ms. A média foi igual a 29452 ms e a mediana 18572 ms. O desvio padrão obtido foi igual a 28793 ms. Ao fazer uma análise apenas do NAF, essa hibridização trouxe bons resultados, porém deve-se analisar que o desvio padrão não foi tão bom, logo, essa modificação trouxe resultados muito dispersos em relação a média, o que a torna menos recomendada para o problema apresentado.

Na modificação PCA-H2 os erros relativos da média para λ e ρc_p foram iguais a 0,008471 W/mK e 0,000003 $\times 10^6$ s/m, respectivamente e para o pior caso iguais a 0,016923 W/mK e 0,003630 $\times 10^6$ s/m, respectivamente.

As hibridizações realizadas entre os métodos LJ e PCA, cujos resultados constam nas Tabelas 4.21, 4.22, 4.23, 4.24, 4.25 e 4.26, são denominadas como LJ-H1, LJ-H2, LJ-H3, LJ-H4, PCA-H1 e PCA-H2, respectivamente. Para uma melhor análise dos resultados obtidos, na Tabela 4.27, é apresentada a distribuição de frequência das referidas hibridizações e, na Figura 4.12, é apresentada a distribuição dos dados dos métodos LJ e PCA hibridizados por meio do boxplot. Os intervalos sem ocorrência foram suprimidos.

Por último, na Figura 4.12 são apresentados os resultados das hibridizações dos métodos acompanhadas das modificações, onde é possível verificar que a hibridização LJ-H1, apesar de obter um número de avaliações da função objetivo menor do que a versão clás-

Tabela 4.26: Resultados obtidos pelo método PCA (PCA-H2) hibridizado com LJ e alteração na função *Perturbation()*.

-	λ (W/mK)	ρc_p (s/m)	$f(\lambda, \rho c_p)$ -	NAF -	Tempo (ms)
Melhor	11,543919	3866267,092721	0,028529	43	1068
Pior	11,739275	3880299,856575	0,030688	4.463	112890
Média	11,641710	3866278,347456	0,029366	1160	29452
Mediana	11,623142	3866725,119172	0,028528	754	18572
Desvio Padrão	0,264352	7964,131998	0,001743	1119	28793

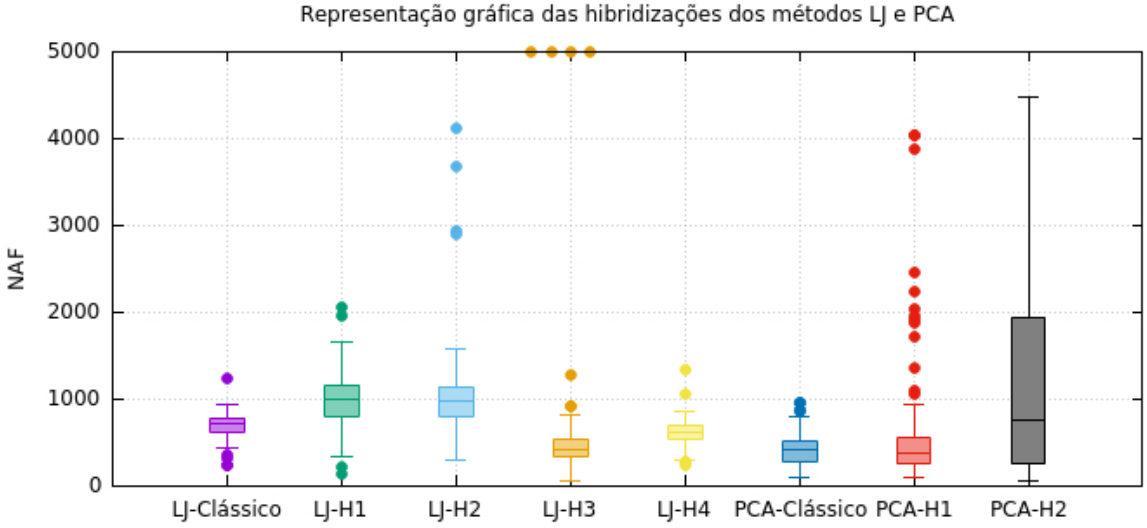


Figura 4.12: Distribuição dos dados para os métodos LJ e PCA nas versões hibridizadas. Fonte: O Autor, 2019.

sica do LJ, os dados estão mais dispersos em relação a média. A hibridização LJ-H2 também não melhorou em relação à versão clássica. Já a hibridização LJ-H3 obteve melhores resultados, pois além dos dados estarem concentrados em torno da média, obteve-se um número de avaliações da função objetivo igual a 42, contra 227 da versão clássica do LJ. A hibridização LJ-H4 obteve um número de avaliações da função objetivo um pouco melhor (221) do que a versão clássica (227) e os dados encontram-se mais concentrados em torno da média. Sendo assim, essa hibridização melhorou os resultados em relação à versão clássica.

Para as hibridizações do PCA, a versão PCA-H1 não melhorou em relação ao número de avaliações da função objetivo, além de exibir um percentual relativamente alto (em torno de 10%) de *outliers*. No PCA-H2, apesar de obter um número de avaliações da função objetivo bem menor (43) do que a versão clássica (85), é possível perceber facilmente que se trata de uma modificação onde os resultados foram bem aleatórios, deixando os

Tabela 4.27: Distribuição de frequência dos métodos LJ e PCA nas versões hibridizadas.

	Número de Ocorrências							
NAF	**LJ-Clássico**	**LJ-H1**	**LJ-H2**	**LJ-H3**	**LJ-H4**	**PCA-Clássico**	**PCA-H1**	**PCA-H2**
$1 \vdash 100$	0	0	0	1	0	2	2	4
$100 \vdash 200$	0	1	0	6	0	8	12	14
$200 \vdash 300$	2	1	1	11	4	18	24	14
$300 \vdash 400$	2	1	0	24	9	18	19	3
$400 \vdash 500$	7	1	3	29	7	27	14	6
$500 \vdash 600$	11	5	3	15	25	14	8	6
$600 \vdash 700$	26	6	8	5	32	6	3	1
$700 \vdash 800$	33	11	11	1	5	3	2	3
$800 \vdash 900$	16	13	14	1	16	2	1	6
$900 \vdash 1000$	2	15	16	2	5	2	2	2
$1000 \vdash 1100$	0	17	16	0	1	0	2	2
$1100 \vdash 1200$	0	8	8	0	0	0	0	3
$1200 \vdash 1300$	1	12	6	1	0	0	1	3
$1300 \vdash 1400$	0	4	8	0	1	0	0	1
$1400 \vdash 1500$	0	1	0	0	0	0	0	1
$1500 \vdash 1600$	0	1	1	0	0	0	0	2
$1600 \vdash 1700$	0	1	0	0	0	0	1	2
$1700 \vdash 1800$	0	0	0	0	0	0	1	0
$1800 \vdash 1900$	0	0	0	0	0	0	2	2
$1900 \vdash 2000$	0	1	0	0	0	0	1	0
$2000 \vdash 2100$	0	1	0	0	0	0	0	3
$2100 \vdash 2200$	0	0	0	0	0	0	1	2
$2200 \vdash 2300$	0	0	0	0	0	0	0	2
$2300 \vdash 2400$	0	0	0	0	0	0	1	0
$2400 \vdash 2500$	0	0	0	0	0	0	1	3
$2500 \vdash 2600$	0	0	0	0	0	0	0	4
$2600 \vdash 2700$	0	0	0	0	0	0	0	1
$2700 \vdash 2800$	0	0	0	0	0	0	0	0
$2800 \vdash 2900$	0	0	1	0	0	0	0	1
$2900 \vdash 3000$	0	0	1	0	0	0	0	1
⋮	⋮	⋮	⋮					
$3600 \vdash 3700$	0	0	1	0	0	0	0	0
⋮	⋮	⋮	⋮					
$3800 \vdash 3900$	0	0	0	0	0	0	1	0
⋮	⋮	⋮	⋮					
$4000 \vdash 4100$	0	0	1	0	0	0	1	0
⋮	⋮	⋮	⋮					
$5000 \vdash 5100$	0	0	0	4	0	0	0	0
⋮	⋮	⋮	⋮					
$15300 \vdash 15400$	0	0	1	0	0	0	0	0
Total	100	100	100	100	100	100	100	100

dados bem dispersos em relação à média.

Com o intuito de analisar as diversas implementações propostas, na Figura 4.13 é apresentado o boxplot dos resultados obtidos com as melhores modificações e hibridizações. As modificações LJ-M1, LJ-M5, LJ-M9, LJ-M10 e LJ-M11 foram melhores do que a versão clássica do LJ. Verifica-se ainda que, apesar do número de avaliações da função objetivo nas modificações LJ-M5 e LJ-M9 não serem menores do que a versão clássica, a

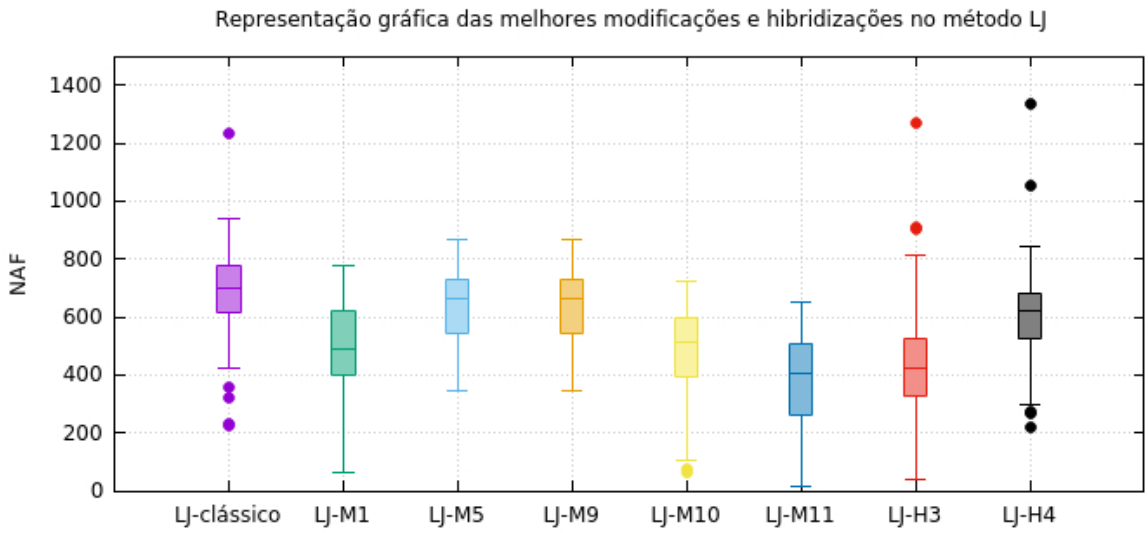


Figura 4.13: Distribuição dos dados para os melhores métodos em relação ao LJ clássico. Fonte: O Autor, 2019.

cauda inferior do diagrama de distribuição da versão clássica está acima das caudas inferiores das respectivas modificações. Já as hibridizações que obtiveram melhores resultados foram as hibridizações LJ-H3 e LJ-H4.

No que se refere as implementações relativas ao Algoritmo de Colisão de Partículas, a única modificação que melhorou os resultados em relação ao PCA clássico foi o método PCA-M1. Na Figura 4.14 é apresentada a comparação do PCA-clássico com o PCA-M1. Nenhuma hibridização do PCA foi melhor do que a versão clássica.

Para finalizar, uma comparação entre os dois melhores métodos, LJ-M11 e PCA-M1 é apresentada na Figura 4.15. Os resultados do PCA-M1 possuem uma dispersão um pouco menor do que o LJ-M11. Se não fosse considerado como melhor resultado, o menor número de avaliações da função objetivo, claramente o PCA-M1 seria a melhor modificação de todas, devido ao desvio padrão ser menor em relação ao LJ-M11. Como o que se define como melhor resultado também envolve o menor número de avaliações da função objetivo, o LJ-M11 pode ser considerado como a melhor modificação para o problema de transferência de calor considerado neste trabalho. No entanto, essas duas modificações são as mais recomendadas para o uso, devido aos resultados obtidos, os quais foram melhores, tornando os o métodos clássicos mais robustos.

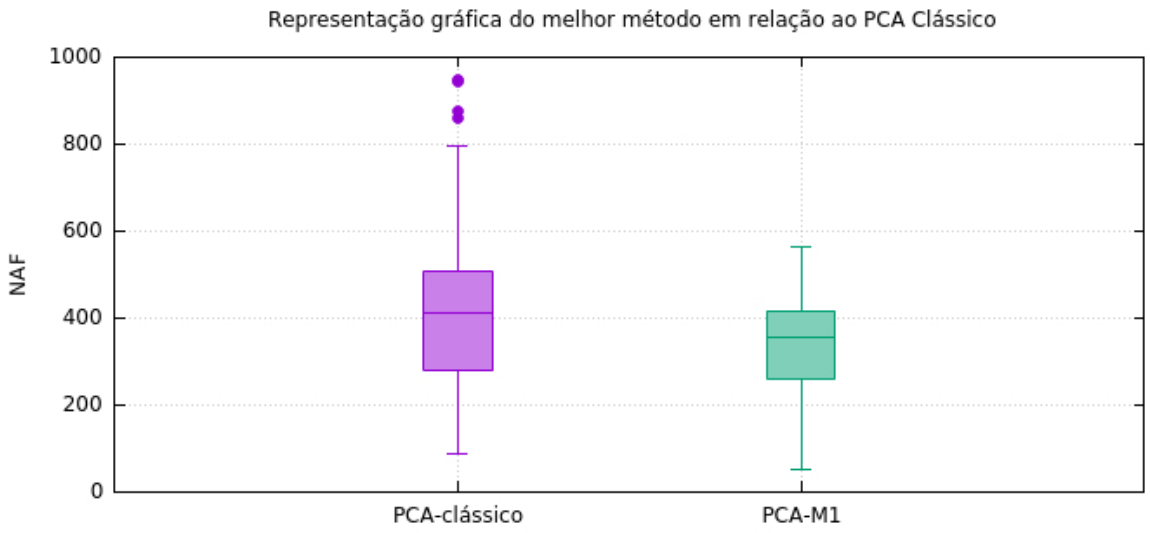


Figura 4.14: Distribuição dos dados para o melhor método em relação ao PCA clássico.
Fonte: O Autor, 2019.

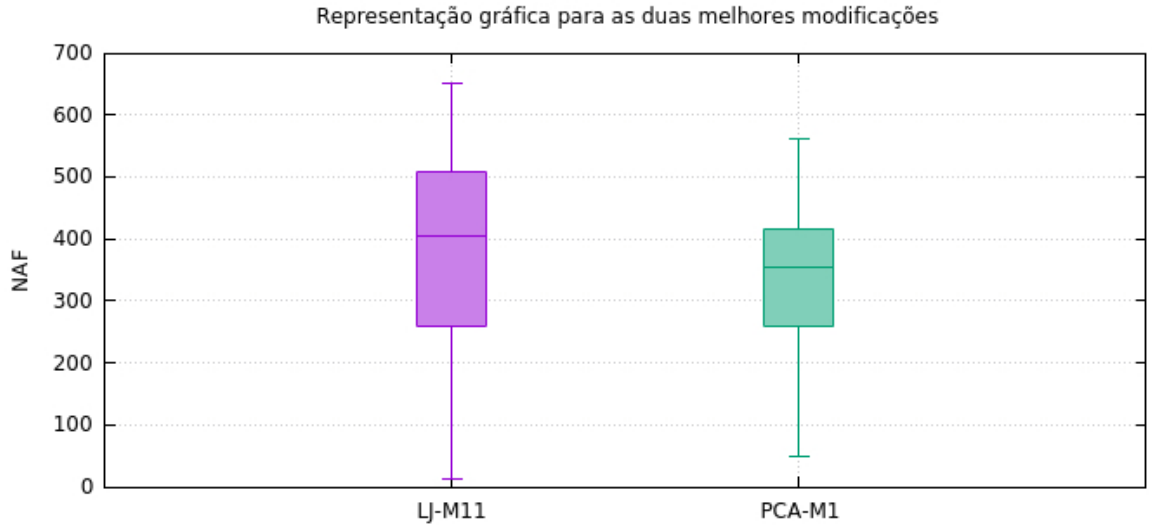


Figura 4.15: Distribuição dos dados para os dois melhores métodos.
Fonte: O Autor, 2019.

Capítulo 5

Conclusões e Trabalhos Futuros

5.1 Conclusões

No problema direto de transferência de calor, tanto na formulação explícita quanto na implícita, obteve-se resultados numéricos compatíveis e satisfatórios com os resultados experimentais obtidos por Carollo [4], levando em consideração que o erro absoluto, dado pela diferença entre os resultados numérico e experimental, foi pequeno, assim como o tempo computacional quando comparado às malhas mais refinadas. No entanto, devido ao critério de estabilidade da formulação explícita, a formulação implícita apresentou melhores resultados, considerando a discretização de 10 nós para malha espacial e 800 nós para malha temporal.

Já no problema inverso de transferência de calor, onde buscou-se estimar a condutividade térmica e a capacidade volumétrica utilizando os métodos de otimização Luus-Jaakola e Algoritmo de Colisão de Partículas, verificou-se que os referidos métodos, em suas versões clássicas (Algoritmos de 1 a 7), obtiveram como melhor resultado os NAFs de 227 e 85, respectivamente.

Algumas modificações foram feitas nos algoritmos utilizando a combinação de estimativas (Algoritmo 9) e combinação de estimativas, juntamente com a contração no intervalo de busca (Algoritmos 12 e 13), na tentativa de se obter melhores resultados (menor NAF), o que ocorreu para o método Luus-Jaakola, com NAF igual a 49, 66 e 14, respectivamente. Os resultados das Tabelas 4.16 e 4.17, obtidos pelas alterações feitas utilizando combinação de estimativa e contração no intervalo de busca, no método Luus-Jaakola, corroboram com os melhores resultados obtidos em relação às versões clássicas do LJ e PCA. Além disso, após combinação de estimativas, no PCA, também foi possível obter uma melhora nos resultados em relação à versão clássica de ambos os métodos, conforme apresentado

na Tabela 4.7. Ressalta-se ainda, que os algoritmos que geraram os resultados dessas tabelas foram implementados criando uma combinação entre as estimativas geradas.

Conforme as Tabelas 4.10 e 4.14, os resultados dos Algoritmos 10 e 11, respectivamente, baseados em alterações na contração do intervalo de busca no método Luus-Jaakola, foram melhores do que a versão clássica do LJ, obtendo NAF = 118 para ambas as modificações. Verificou-se ainda, na Tabela 4.18, dados resultantes da execução do Algoritmo 14, baseado em modificação feita no PCA, cuja função *Perturbation()* foi modificada, obtendo uma nova forma de se gerar o vetor **NewConfig**, que o NAF obtido foi igual a 78, sendo melhor do que ambas as versões clássicas dos métodos LJ e PCA.

No que se refere às hibridizações realizadas entre os métodos LJ e PCA, sendo que o algoritmo LJ faz chamada às funções *Scattering()* ou *Perturbation()*, ambas modificadas, na Tabela 4.21, resultado dos Algoritmos 15 e 16, é apresentado um NAF = 126, acarretando em uma melhoria em relação ao NAF do LJ na versão clássica. Já na Tabela 4.23, cujos resultados são baseados na saída dos Algoritmos 20 e 21, obteve-se NAF = 42, o que mostrou ser melhor do que ambos os métodos na versão clássica. Por outro lado, na Tabela 4.24, resultado baseado nos Algoritmos 20 e 22, com NAF = 221, o método foi ligeiramente melhor do que a versão clássica do LJ.

Outra alteração realizada no método PCA, mais especificamente na função *Perturbation()*, conforme Algoritmo 24, fazendo $\mathbf{NewConfig} = \mathbf{OldConfig} + R\boldsymbol{r}^{(k-1)}$, onde R é uma matriz diagonal de números randômicos entre -0,5 e 0,5, obteve-se um resultado melhor do que na versão clássica do método LJ e, embora não tenha sido melhor do que a versão clássica do PCA, se aproximou muito do seu resultado.

Ainda no que diz respeito ao método PCA, conforme explicitado no Algoritmo 25, foi realizada uma alteração similar à versão anterior (Algoritmo 24), sendo que na função *Perturbation()* uma nova configuração foi implementada fazendo $\mathbf{NewConfig} = \mathbf{OldConfig} + R\boldsymbol{r}^{(k-1)}$, onde R é uma matriz diagonal com números entre -0,5 e 0,5 e $\mathbf{NewConfig} = \frac{1}{4}(3\ \mathbf{OldConfig} + \mathbf{NewConfig})$. Com essa hibridização, o método resultou no NAF = 43, melhorando o resultado em relação à versão clássica, tanto do LJ quanto do PCA.

Algumas modificações realizadas no presente trabalho, não trouxeram uma melhorias no que se refere ao menor NAF, como foi o caso das modificações LJ-M2, LJ-M3, LJ-M5, LJ-M6 e LJ-M7, embora todas essas modificações tenham encontrado o ponto de mínimo da função. Por outro lado, a melhor modificação (LJ-M11), no que se refere ao menor NAF, se originou da mesma técnica aplicada nessas versões. As modificações realizadas

no PCA obtiveram um NAF menor do que a versão clássica do PCA.

Já em relação as hibridizações, o ponto de mínimo foi encontrado por todas as versões, embora as versões LJ-H2 e PCA-H1 não tenham obtido um NAF menor do que as versões clássicas dos métodos LJ e PCA, respectivamente.

Como os valores das temperaturas obtidos no experimento de Carollo [4] não estão disponíveis de forma explícita, ou seja, através de uma tabela de valores, foi utilizado o software "Pega-ponto" para obter mais informações sobre o gráfico de distribuição dessas temperaturas ao longo do tempo, o que implica em erros em relação aos dados reais.

Por fim, em decorrência do exposto anteriormente, verifica-se que, ao utilizar os valores de λ e ρc_p encontrados por Carollo [4], tem-se um erro entre a malha espacial com 10 nós ($\triangle x = 0,001209$) e malha temporal com 800 nós ($\triangle t = 0,2$) de aproximadamente 0,07 em relação aos dados experimentias, enquanto os erros obtidos nesse trabalho foram abaixo de 0,02 nos melhores casos, o que significa que as modificações em alguns métodos trouxeram melhorias significativas, considerando o menor NAF.

5.2 Trabalhos Futuros

Neste trabalho um problema direto de transferência de calor foi resolvido e os resultados numéricos foram comparados com os dados experimentais obtidos por Carollo [4]. A matriz resultante da formulação implícita foi resolvida utilizando o método Gauss-Seidel. Como trabalho futuro, pretende-se utilizar outros métodos para solução da matriz, como por exemplo, o Algoritmo de Thomas, também chamado de Algoritmo de Matriz Tridiagonal (do inglês *Tridiagonal Matrix Algorithm* - TDMA), por possuir um custo computacional inferior aos métodos de eliminação usualmente adotados.

Foram, ainda, analisadas diversas modificações e hibridizações dos métodos Luus-Jaakola e Algoritmo de Colisão de Partículas para estimar os parâmetros λ e ρc_p de maneira a possibilitar a obtenção de soluções numéricas para a temperatura próximas aos dados experimentais descritos em Carollo [4], após a solução do problema direto de transferência de calor. Como trabalho futuro, pretende-se ainda, realizar hibridizações com outros métodos de otimização como, Algoritmos Genéticos e Colônia de Formigas, por exemplo, com o intuito de tentar melhorar os parâmetros estimados. Uma hibridização entre métodos estocásticos e determinísticos também pode ser uma boa técnica na tentativa de estimar os parâmetros no problema apresentado nesse trabalho.

Além disso, um problema similar em três dimensões pode ser abordado com o objetivo de solucioná-lo por meio das abordagens direta e inversa, utilizando os mesmos métodos abordados nesse trabalho, assim como outros métodos sugeridos.

Referências

[1] Arenales, M., Armentano, V., Morabito, R., Yanasse, H. *Pesquisa Operacional*, 4 ed. Editora Campus, 2007.

[2] Borgnakke, C., Sonntag, E. R. *Fundamentos da Termodinâmica*, 8 ed. Blucher, 2013.

[3] Burden, R. L., Faires, D. J., Burden, A. M. *Numerical Analysis*, 10 ed. CENGAGE Learning, 2016.

[4] Carollo, L. F. S. Estimação simultânea de propriedades termofísicas de materiais metálicos. Dissertação de mestrado, Universidade Federal de Itajubá, Itajubá, MG, 2010.

[5] Carvalho, M., Pereira, O. S. Aplicação da modelagem matemática para estudo do crescimento populacional de Seropédica - RJ. *Revista Eletrônica Teccen 8* (2015), 5–9.

[6] Costa, J. F., Dias, D. G. Equação do Calor: uma comparação entre soluções analítica e computacional para uma barra de cobre finita e isolada termicamente. *Revista Eletrônica de Matemática - REMAT 4* (2018), 27–37.

[7] Cuminato, J. A., Júnior, M. M. *Discretização de Equações Diferenciais Parciais: Técnicas de Diferenças Finitas*, 1 ed. SBM, 2013.

[8] Fernandes, L. L., Cruz, J. C. R., Blanco, C. J. C., Barp, A. R. B. Modelagem Sísmica via métodos das diferenças finitas: caso da bacia do Amazonas. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería 39* (2009), 155–163.

[9] Fortuna, A. O. *Técnicas Computacionais para Dinâmica dos Fluidos: Conceitos Básicos e Aplicações*, 1 ed. USP, 2000.

[10] Incropera, F. P., Dewitt, D. P., Bergman, T. L., Lavine, A. S. *Fundamentals of Heat and Mass Transfer*, 6 ed. John Wiley and Sons, 2007.

[11] Iório, V. *EDP: Um curso de graduação*, 1 ed. IMPA, 2016.

[12] Jen, L. C., Rissetti, G., Ferreira, A. G., Yamagushi, A. H., Coelho, L. F. Aplicativo visual para problemas de transferência de calor. *Exacta 4* (2006), 385–390.

[13] Linga, P., Al-Saifi, N., Englezos, P. Comparison of the Luus-Jaakola Optimization and Gauss-Newton Methods for Parameter Estimation in Ordinary Differential Equation Models. *Industrial and Engineering Chemistry Research 45* (2006), 4716–4725.

[14] Lobato, F. S., Almeida, G. M., Fernandes, C. F., Almeida, L. M. S. Algoritmo de colisão de partículas aplicado ao projeto de sistema de engenharia. *Nono Simpósio de Mecânica Computacional, Universidade Federal de São João Del-Rei - UFSJ, São João Del-Rei, MG* (2010).

[15] Luus, R., Jaakola, T. H. I. Optimization by Direct Search and Systematic Reduction of the Size of Search Region. *American Institute Chemical Engineering 19* (1973), 760–766.

[16] Luz, E. F. P., Becceneri, J. C., Velho, H. F. C. Uma nova metaheurística baseada em algoritmo de colisão de múltiplas partículas. *Simpósio de Pesquisa Operacional e Logística da Marinha - SPOLM, Rio de Janeiro, RJ* (2008).

[17] Luz, E. F. P., Becceneri, J. C., Velho, H. F. C. Avaliação do algoritmo de colisão de múltiplas partículas na solução de problemas inversos. *Workshop de Computação Aplicada - WORCAP, Instituto Nacional de Pesquisas Espaciais - INPE, São José dos Campos, SP* (2011).

[18] Núñez, Y., Faria, C., Loula, A., Malta, S. Um método híbrido de elementos finitos aplicado a deslocamentos miscíveis em meios porosos heterogêneos. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería 17* (2017), 45–51.

[19] Oliveira, T. C. A., Barreto, G., Pasqua, A. M. Modelagem computacional de efeitos de distorções não lineares para guitarra elétrica. *Revista Brasileira de Computação Aplicada 5* (2013), 69–84.

[20] Pereira, A. J., Lisboa, N. H., Filho, J. H. D. Análise da estabilidade do método explícito para discretização de equações diferenciais parabólicas por meio de diferenças finitas. *Edição Iniciação Científica* (2017).

[21] Porto, J. C. U., Silva, J. F. A fórmula de taylor e suas aplicações. *XXV Encontro de Iniciação à Docência, Universidade Federal do Ceará - UFC, Fortaleza, CE* (2016).

[22] Resende, J. V., Jr, V. S. Medidas da condutividade térmica efetiva de modelos de polpas de frutas no estado congelado. *Food Science and Technology 22* (2002), 177–183.

[23] Retirado-Mediaceja, Y., Lamorú-Urgelles, M., Góngora-Leyva, E., Torres-Tamayo, E., de la Cruz, B. L., García-Batista, D. Transferencia de calor en el secado solar a la intemperie de menas lateríticas ferroniquelíferas. *Minería y Geología 27* (2011), 1–21.

[24] Ruggiero, M. A. G., da Rocha Lopes, V. L. *Cálculo Numérico: Aspectos Teóricos e Computacionais*, 1 ed. PEARSON, 2000.

[25] Sacco, W. F., Oliveira, C. R. E. A new stochastic optimization algorithm based on a particle collision metaheuristic. *6th World Congresses of Structural and Multidisciplinary Optimization, Rio de Janeiro, RJ* (2005).

[26] Silva, A. T., Telles, W. R., Semaan, G. S. Aplicação do método das diferenças finitas em um problema de transferência de calor utilizando as formulações explícita e implícita. *Anais do XII Encontro Acadêmico de Modelagem Computacional - EAMC, Petrópolis, RJ,* (2019), 120–129.

[27] Silva, A. T., Telles, W. R., Semaan, G. S. Avaliação dos métodos Luus-Jaakola e Algoritmo de Colisão de Partículas em um problema de transferência de calor. *Anais do XII Encontro Acadêmico de Modelagem Computacional - EAMC, Petrópolis, RJ,* (2019), 130–138.

[28] Silva, L. B., Ferreira, L. L., Moreira, F. M. B. Modelagem Matemática: Reflexões Teóricas e Aplicações. *VII Encontro Mineiro de Educação Matemática - EMEM, São João Del-Rei, MG* (2015).

[29] Silva Neto, A. J., Becceneri, J. C., Velho, H. F. C. *Inteligência Computacional Aplicada a Problemas Inversos em Transferência Radiativa*, 1 ed. EdUERJ, 2016.

[30] Telles, W. R. *Previsão do comportamento hidráulico de um rio com base na estimativa de coeficientes que controlam seu escoamento. Estudo de caso: Rio Bengalas, Nova Friburgo-RJ.* Tese de doutorado, Universidade do Estado do Rio de Janeiro - UERJ, Nova Friburgo, RJ, 2014.

[31] Telles, W. R., Gonçalves, G. I., Neto, A. J. S., Rodrigues, P. P. G. W. Estimativa de parâmetros hídricos usando técnicas de problemas inversos. *Associação Brasileira de Engenharia e Ciências Mecânicas - ABCM, Universidade Estadual de Santa Cruz - UESC, Ilhéus, BA* (2013).

[32] Çengel, Y. A., Ghajar, A. J. *Transferência de Calor e Massa: Uma abordagem prática*, 4 ed. McGrawHill, 2012.