



Universidade Federal
do Rio de Janeiro
Escola Politécnica

SOLUÇÃO DA EQUAÇÃO TRIDIMENSIONAL TRANSIENTE DO CALOR
ATRAVÉS DO MÉTODO DE ELEMENTOS FINITOS APLICADO A DISCOS
DE FREIO

Felipe Rodrigues de Mello Alves

Projeto de Graduação apresentado ao Curso
de Engenharia Mecânica da Escola Politécnica,
Universidade Federal do Rio de Janeiro, como
parte dos requisitos necessários à obtenção do
título de Engenheiro.

Orientador: Prof. Gustavo Rabello dos Anjos,
Ph.D.

Rio de Janeiro
Maio de 2021



UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Departamento de Engenharia Mecânica
DEM/POLI/UFRJ



**SOLUÇÃO DA EQUAÇÃO TRIDIMENSIONAL TRANSIENTE DO CALOR
ATRAVÉS DO MÉTODO DE ELEMENTOS FINITOS APLICADO A DISCOS
DE FREIO**

Felipe Rodrigues de Mello Alves

PROJETO FINAL SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO
DE ENGENHARIA MECÂNICA DA ESCOLA POLITÉCNICA DA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE
DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE
ENGENHEIRO MECÂNICO.

Aprovada por:

Prof. Gustavo Rabello dos Anjos, Ph.D.

Prof. Carolina Palma Naveira Cotta, D.Sc.

Prof. Roney Leon Thompson, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

MAIO DE 2021

Alves, Felipe Rodrigues de Mello

Solução da equação tridimensional transiente do calor através do método de elementos finitos aplicado a discos de freio/ Felipe Rodrigues de Mello Alves. – Rio de Janeiro: UFRJ/Escola Politécnica, 2021.

XIII, 108 p.: il.; 29, 7cm.

Orientador: Gustavo Rabello dos Anjos

Projeto de Graduação – UFRJ/ Escola Politécnica/
Curso de Engenharia Mecânica, 2021.

Referências Bibliográficas: p. 82 – 82.

1. Equação do calor. 2. Método de elementos finitos. 3. Discos de freio. I. Rabello dos Anjos, Gustavo. II. Universidade Federal do Rio de Janeiro, UFRJ, Curso de Engenharia Mecânica. III. Solução da equação tridimensional transiente do calor através do método de elementos finitos aplicado a discos de freio.

*À minha família por todo o
suporte e compreensão.*

Agradecimentos

Sou muito grato não apenas ao investimento financeiro e educacional fornecido pelos meus pais Eduardo e Isabel mas também por toda a compreensão e apoio na jornada da graduação, que teve também grande suporte da minha dinda Daize na perspectiva do mercado de trabalho e complemento na formação profissional. Agradeço ao suporte de todos os meus amigos de fora da faculdade que perceberam meus momentos difíceis e foram capazes de me apoiar mesmo sem entender bem a realidade do curso, mas principalmente a meus amigos de faculdade que foram grandes companheiros em momentos de extrema importância. Um agradecimento especial à equipe Icarus de Formula SAE por me proporcionar a paixão pela engenharia mecânica e me ensinar a ser um engenheiro mais completo, a todos os profissionais da Gerência de equipamentos de subestações de Furnas, que foram de extrema cordialidade no meu início da vida profissional e por último, a todos os colegas de trabalho no setor de Installation Analysis Subsea (IAS) da TechnipFMC, que me proporcionaram uma oportunidade completa de atuar com a engenharia, fornecendo todas as ferramentas e suportes para o desenvolvimento de um trabalho do mais alto nível.

Agradeço ao corpo docente do Departamento de Engenharia Mecânica da UFRJ que teve grande participação na minha formação, em especial ao meu orientador Gustavo, por instigar o conhecimento no tema apresentado no trabalho de maneira completa. Em uma época que o mundo começava a enfrentar grandes problemas devido ao COVID-19, insistiu em passar conhecimento em atividades complementares online com didática e paciência exemplares, que fizeram me interessar no assunto.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Mecânico

**SOLUÇÃO DA EQUAÇÃO TRIDIMENSIONAL TRANSIENTE DO CALOR
ATRAVÉS DO MÉTODO DE ELEMENTOS FINITOS APLICADO A DISCOS
DE FREIO**

Felipe Rodrigues de Mello Alves

Maio/2021

Orientador: Gustavo Rabello dos Anjos

Programa: Engenharia Mecânica

Este trabalho propõe o desenvolvimento de um código que utiliza o método de elementos finitos (MEF) capaz de calcular soluções aproximadas para o problema térmico aplicado a discos de freio submetidos à situações críticas de frenagem de um protótipo de competição tipo formula. A solução da equação transiente de calor fornece uma alternativa aos softwares de altos investimentos financeiros e capacidades computacionais no momento do estudo da dissipação de calor dos projetos, trazendo facilidade ao estudo comparativo de materiais e variações na geometria.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment
of the requirements for the degree of Mechanical Engineer

SOLUTION OF THE THREE-DIMENSIONAL TRANSIENT HEAT EQUATION
USING THE FINITE ELEMENT ANALYSIS APPLIED TO BRAKE DISCS

Felipe Rodrigues de Mello Alves

May/2021

Advisor: Gustavo Rabello dos Anjos

Department: Mechanical Engineering

This work proposes the development of a Finite Element Method (FEM) code in order to calculate approximate solutions for heat transfer in a formula student prototype brake disc under critical braking situations. The solution of the transient heat equation provides an alternative to high cost commercial softwares that require powerful computer memories, enabling the user to easily input data from the project, allowing comparison of different materials and geometries.

Sumário

Lista de Figuras	xi
Lista de Tabelas	xiii
1 Introdução	1
1.1 Motivação	2
1.2 Objetivo	2
1.3 Metodologia	2
1.4 Organização da tese	3
2 Revisão Bibliográfica	4
2.1 Transferência de calor	4
2.1.1 Condução	5
2.1.2 Convecção	5
2.1.3 Radiação	7
2.1.4 Equação do calor	7
2.2 Sistema de freios	9
2.2.1 Transmissão da força do pedal ao sistema de freios	11
2.2.2 Torque de frenagem	13
2.2.3 Frenagem ótima	14
2.2.4 Curvas de frenagem	18
2.2.5 A questão térmica	21
2.3 Método de elementos finitos	22
2.3.1 Método de resíduos ponderados	25
2.3.2 Método de Galerkin	27
2.3.3 Matriz do elemento	29

2.3.4	Assembling	31
2.3.5	Matriz de conectividade	32
2.3.6	Condições de contorno	33
2.3.7	Sistema linear	34
2.4	Método de diferenças finitas	34
3	Metodologia	35
3.1	Método de Elementos Finitos	35
3.1.1	Sequência de passos para aplicação do método	35
3.1.2	Aplicação em um problema tridimensional transiente com condição de contorno de Dirichlet	36
3.1.3	Aplicação em um problema tridimensional transiente com condição de contorno de Neumann não homogêneo e constante	39
3.1.4	Aplicação em um problema tridimensional transiente com condição de contorno de Robin	40
3.1.5	Equacionamento do caso completo de frenagem	41
3.2	Modelo de frenagem	43
3.2.1	Fluxo de calor	44
3.2.2	Convecção	44
3.3	O Algoritmo computacional	45
3.3.1	Gerador de malhas - GMSH	46
3.3.2	Módulo de montagem	50
3.3.3	Módulo de matrizes elementares	51
4	Validação	54
4.1	Validação através de solução analítica	54
4.1.1	Solução analítica proposta	54
4.1.2	Comparação de resultados	56
4.2	Comparação com software comercial	58
4.2.1	Condição de contorno de Dirichlet	58
4.2.2	Condição de contorno de Neumann e Robin	63
5	Resultados e Discussões	66
5.1	Utilização de matrizes esparsas	66

5.2	Frenagem crítica	68
5.3	Variação da espessura do disco	71
5.4	Variação no raio interno do disco	73
5.5	Variação na distribuição de calor entre pastilhas	74
5.6	Múltiplas frenagens	75
6	Conclusões	80
6.1	Sugestões para trabalhos futuros	81
Referências Bibliográficas		82
A	Código Fonte	84
A.1	Código principal	84
	A.1.1 Importação de módulos	84
	A.1.2 Módulo principal	84
A.2	Módulo da leitura do input	90
A.3	Módulo da construção e leitura da malha	93
	A.3.1 Malha de discos de freio	93
	A.3.2 Malha de paralelepípedos	97
A.4	Módulo da montagem	99
A.5	Módulo das matrizes elementares	100
A.6	Código principal da malha de paralelepípedo	103

Lista de Figuras

1.1	Modelagem de problemas reais para solução numérica	1
2.1	Camada limite térmica	6
2.2	volume infinitesimal	8
2.3	Pitch causado pela transferência de carga e de massa na frenagem . .	10
2.4	Balance bar entre sistema dianteiro e traseiro e cilindros mestre . . .	11
2.5	Sistema de freios interligado por linhas pressurizadas	12
2.6	Conjunto linha, pinça e disco de freios	13
2.7	Diagrama de corpo livre da roda	14
2.8	Medidas e forças que ocorrem no protótipo no momento da frenagem	15
2.9	Curva relacionando as forças nos eixos dianteiro e traseiro na situação de frenagem de um caminhão vazio e cheio	16
2.10	Variação no valor de coeficiente de atrito	17
2.11	Curvas de frenagem normalizadas	19
2.12	Curva de frenagem ótima de um veículo	20
2.13	Motocicleta perdendo contato entre eixo traseiro e pista	20
2.14	Pinça de freio proporcionando atrito ao disco de freios	21
2.15	Pinça de freio proporcionando atrito ao disco de freios	22
2.16	Elementos tridimensionais e seus nós	24
2.17	Malha de um cubo com elementos tetraédricos	24
2.18	Projeção de um vetor em um espaço de dimensão inferior	29
2.19	Vértices do elemento tetraédrico	30
2.20	Montagem das matrizes elementares em matrizes globais	32
2.21	Montagem das matrizes elementares em matrizes globais	33
3.1	Condições de contorno	41

3.2	Esquema geométrico do contato entre disco e pastilha	43
3.3	Diagrama de funcionamento do código	45
3.4	Planilha de input do usuário	46
4.1	Condições de contorno para solução analítica	55
4.2	Condições de contorno para solução analítica	55
4.3	Visualização da simulação numérica	56
4.4	Comparação de resultados no eixo central do plano estudado	56
4.5	Comparação de resultados em $x = 0, 25$	57
4.6	Comparação de resultados em $x = 0, 8$	57
4.7	Malha e resultado do Ansys. A foto representa o instante de 100s de duração. O gráfico mostra as temperaturas máxima, mínima, e média dos elementos	59
4.8	Comparação de resultados no instante de 10 segundos	60
4.9	Comparação de resultados no instante de 20 segundos	60
4.10	Comparação de resultados no instante de 30 segundos	61
4.11	Comparação de resultados no instante de 40 segundos	61
4.12	Comparação de resultados no instante de 50 segundos	62
4.13	Comparação de resultados no instante de 100 segundos	62
4.14	Temperatura final do código e do <i>Ansys</i>	65
5.1	Terminal mostrando a performance de simulação	67
5.2	Gradiente de temperatura nos instantes (a) $t = 0s$, (b) $t = 0.25s$, (c) $t = 0.50s$, (d) $t = 1.0s$	71
5.3	Gradiente de temperatura no instante final $t = 1.9s$	71
5.4	Disco mais espesso conduzindo calor	72
5.5	Gradiente de temperatura nos instantes (a) $t = 0s$, (b) $t = 0.30s$, (c) $t = 0.80s$, (d) $t = 1.90s$	73
5.6	Temperatura desigual nos instantes (a) $t = 0.3s$, (b) $t = 0.5s$, (c) $t = 0.7s$, (d) $t = 1.1s$	75
5.7	Temperatura nos instantes (a) $t = 3s$, (b) $t = 6s$, (c) $t = 9s$	77
5.8	Temperatura nos instantes (a) $t = 12s$, (b) $t = 15s$, (c) $t = 18s$	77
5.9	Temperatura em todos os ciclos	78

Lista de Tabelas

4.1	Erros relativos ponto a ponto	58
4.2	Propriedades térmicas utilizadas nas simulações	58
4.3	Dados da simulação	63
4.4	Fluxo de calor e coeficiente de convecção	64
4.5	Erros relativos na simulação da frenagem única	65
5.1	Tabela comparativa entre o uso de matrizes comuns e esparsas	67
5.2	Dados do protótipo considerados	68
5.3	Cinemática da frenagem	69
5.4	Propriedades físicas do material selecionado para o disco de freios . .	69
5.5	Parâmetros gerais da simulação	69
5.6	Fluxo de calor e coeficiente de convecção considerados	70
5.7	Resultados para diferentes espessuras	72
5.8	Comparação considerando o aumento do raio interno	73
5.9	Simulação de fluxo de calor desigual entre as pastilhas de freio	74
5.10	Parâmetros gerais da simulação	75
5.11	Dados propostos para 1 ciclo de frenagem	76
5.12	Temperaturas atingidas pelo disco de freio dianteiro em frenagens múltiplas	78

Capítulo 1

Introdução

Para transformar um problema real em um modelo compatível com soluções computacionais existem algumas metodologias de idealização do fenômeno físico - chamado de modelo físico - que pode ser descrito matematicamente. Em muitas aplicações, torna-se necessário transformar o modelo matemático em um modelo numérico a fim de discretizá-lo e resolvê-lo, como esquematizado na Figura 1.1.

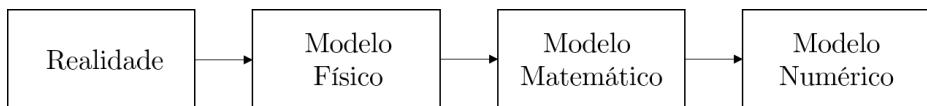


Figura 1.1: Modelagem de problemas reais para solução numérica

As simulações computacionais fazem uso de diferentes métodos numéricos e técnicas para calcular soluções envolvendo os problemas típicos da ciência e engenharia. Esse métodos se tratam de procedimentos que buscam soluções aproximadas para problemas de valores de contorno. Alguns dos métodos conhecidos são: método de diferenças finitas, método de elementos finitos e método de volumes finitos. Cada um desses métodos conhecidos têm sua vantagens e desvantagens com relação a acurácia e estabilidade. O método de elementos finitos (MEF) se aplica a uma grande variedade de problemas físicos e, portanto, é um método largamente estudado na comunidade científica, o que confere a ele uma robustez e sólida confiabilidade.

Uma aplicação do MEF bastante importante no estudo da engenharia e que será explorada neste trabalho é a solução de problema envolvendo transferência de calor em sólidos. O estudo do calor se torna necessária em diversas aplicações na

sociedade - tal como na medicina, no armazenamento de comidas e de fármacos - e consequentemente na engenharia - como no funcionamento de motores, turbinas, trocadores de calor, entre outros.

A transferência de calor ocorre de forma contínua, geralmente em geometrias complexas e passa por um período transitório até chegar em algum estado que possa ser considerado permanente. Os problemas térmicos geralmente são de difícil solução, onde os métodos numéricos ganham grande importância.

1.1 Motivação

Veículos de competição são submetidos a grandes esforços uma vez que o seu objetivo é ser mais eficiente em uma pista de corrida. As pistas planejadas para veículos de alta performance são compostas de diversos trechos curvos, principalmente as pistas de Formula SAE e a frenagem eficiente permite maior precisão nas entradas de curva. Sendo assim o sistema de freios é constantemente acionado, o que leva a um considerável aumento de temperatura dos seus componentes. O aumento descontrolado de temperatura nos discos de freio podem resultar em fadiga térmica, vaporização do fluido de freio, rachaduras e vibrações prejudiciais ao funcionamento do sistema, o que compromete a segurança e performance do protótipo.

1.2 Objetivo

Este trabalho tem como missão fornecer uma ferramenta de simulação térmica que utiliza o método de elementos finitos afim de estudar os fenômenos energéticos envolvendo os componentes dos freios de um carro de competição, sendo assim capaz de fornecer estudos sobre a influência das grandezas envolvidas no projeto.

1.3 Metodologia

Para desenvolver um algoritmo automatizado de solução da equação do calor tridimensional transitório, a linguagem de programação de alto nível Python é utilizada e organizada em diferentes módulos. A modularização permite que o código fique mais funcional, organizado e objetivo.

Afim de testar a precisão dos resultados encontrados, soluções analíticas e comparações com software comercial são feitas.

Já as condições de contorno para representar o problema de transferência de calor durante a frenagem são estimadas através de registros empíricos do uso dos discos de freio e calculadas de acordo com as grandezas físicas envolvendo os protótipos de Formula SAE da UFRJ.

1.4 Organização da tese

Este trabalho é dividido no total de 6 capítulos, dos quais os 5 capítulos além da introdução são:

- **Capítulo 2** - Trabalha todo o conceito teórico necessário para o desenvolvimento do trabalho, relembrando conhecimentos descritos por autores que são referência nos assuntos abordados;
- **Capítulo 3** - Demonstra detalhadamente o desenvolvimento do Método de Elementos Finitos na Equação do calor para diferentes condições de contorno, apresenta os cálculos do modelo de frenagem utilizado e, por último, descreve o algoritmo computacional desenvolvido;
- **Capítulo 4** - Explicita as comparações entre os resultados numéricos encontrados com solução analítica e com simulações em um software comercial consolidado;
- **Capítulo 5** - Apresenta os resultados numéricos do modelo de frenagem proposto, trazendo comparativos entre mudanças de geometria do projeto do disco de freios. Apresenta também uma análise de viabilidade do código com relação a utilização de memória computacional;
- **Capítulo 6** - Apresenta as conclusões do trabalho e traz sugestões para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

2.1 Transferência de calor

A energia dada por partículas como átomos, moléculas ou elétrons partindo de regiões mais quentes para regiões mais frias é definida, por Ozisik [1], como calor. O fenômeno natural do calor é mais observado pelo ser humano desde que ele possui domínio sobre o fogo, por volta de 1200 A.C. Entretanto apenas na civilização grega o assunto começou a ser estudado. Acredita-se que o primeiro estudo do calor do fogo foi com a finalidade de transformar diferentes materiais em ouro. Os alquimistas da época tinham a teoria de que para isso acontecer, era necessário ter um alto controle da temperatura no processo.

Os anos passaram e o calor ganhou diversas aplicações para a humanidade, principalmente quando as primeiras turbinas a vapor surgiram. Na engenharia mecânica, o calor pode ser desejado ou algo a ser evitado, dependendo da situação. Por exemplo no automobilismo é necessário uma centelha para um motor a combustão funcionar, porém se o radiador superaquecer, o motor pode subir a temperaturas altas demais para os materiais utilizados, o que leva a fusão de peças do motor.

Segundo Frank P. Incropera [2] a Transferência de calor é definida como *"energia térmica em trânsito devido a uma diferença de temperatura no espaço"*. Em um disco de freios ocorrem os três tipos de transferência de calor descrito pelo autor: condução, convecção e radiação. Quando existe um gradiente de temperatura em um meio estacionário sólido ou fluido a *condução* ocorre quando há transferência de calor através do meio. Já a *convecção* ocorre quando há transferência de calor entre

uma superfície e um fluido em movimento. Quando a troca de calor ocorre através de ondas eletromagnéticas que são emitidas por qualquer superfície com temperatura não nula (diferente de $0K$), ocorre a *radiação*. O estudo desses fenômenos ocorre através de equações de taxas que quantificam a energia transferida por unidade de tempo. Como Ozisik [1] coloca, apesar de se falar bastante em fluxo de calor, é bom registrar que este fluxo em si não é uma grandeza capaz de ser medida diretamente, entretanto esta grandeza têm significado físico bem definido por ser relacionado com a grandeza mensurável chamada Temperatura.

Cada tipo de transferência de calor tem sua descrição matemática particular, como se segue.

2.1.1 Condução

A condução pode ser interpretada como um fenômeno difusivo onde ocorre transferência de energia das partículas mais energéticas para as menos energéticas através das interações entre partículas. Em um sólido essas interações ocorrem através da combinação entre a vibração das moléculas dos retículos cristalinos e a movimentação dos elétrons livres. A equação que descreve a taxa de transferência de calor para a condução foi proposta por um físico-matemático francês chamado Joseph Fourier e é conhecida como a *lei de Fourier* - Equação 2.1.

$$\mathbf{q}_{cond} = -k\nabla T = -k \left(\mathbf{i} \frac{dT}{dx} + \mathbf{j} \frac{dT}{dy} + \mathbf{k} \frac{dT}{dz} \right) \quad (2.1)$$

sendo \mathbf{q} [W/m^2] o fluxo de calor por área e k [W/mK] a condutividade térmica. A condutividade térmica pode ser constante ou variável de acordo com a temperatura e é uma propriedade do material que está conduzindo o calor.

2.1.2 Convecção

A convecção ocorre quando existe um gradiente de temperatura entre uma superfície e um grande número de moléculas em movimento. A transferência de calor ocorre devido a superposição de dois fenômenos: o transporte das moléculas devido ao movimento do fluido e o transporte de energia devido ao movimento aleatório de cada molécula (fenômeno difusivo, como a condução). A equação que descreve a

taxa de transferência da convecção é a *lei do resfriamento de Newton*:

$$\mathbf{q}_{conv} = h(T_s - T_\infty) \quad (2.2)$$

onde h [W/m^2K] é o coeficiente de transferência de calor por convecção que depende tanto do material do sólido quanto das condições da camada limite do escoamento, o que significa que h depende também das propriedades do fluido e da geometria do sólido. Além disso \mathbf{q}_{conv} depende da diferença entre a temperatura do sólido T_s e do fluido T_∞ .

Em mecânica dos fluidos, quando se estuda escoamentos sobre superfícies, existe um conceito chamado de camada limite. Definida por Ludwig Prandtl em 1904 como uma camada de fluido nas imediações de uma superfície delimitadora, a camada limite descreve a região onde o fluido sofre influência do arrasto devido à sua viscosidade em contato com o sólido. Ou seja, partindo da superfície do sólido, o local onde a viscosidade e o arrasto começam a ser desprezíveis pode ser definido como a região fora da camada limite.

No estudo da convecção, existe a camada limite térmica. Uma vez que o fluido entra em contato com o sólido, as partículas que estão em contato direto entram em equilíbrio térmico com a superfície. Entretanto, essas partículas vão trocar energia com as partículas adjacentes do escoamento, o que vai criar um gradiente de temperatura, como mostra a Figura 2.1.

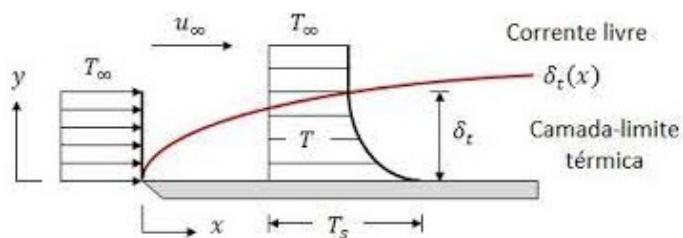


Figura 2.1: Camada limite térmica

Fonte: <https://www.respondeai.com.br>

No estudo de mecânica dos fluidos - e portanto da convecção na transferência de calor - é comum a utilização de grandezas empíricas, que foram propostas ao longo da história para solucionar problemas desta natureza. O número de Reynolds, por exemplo, é um número adimensional que permite avaliar o tipo de escoamento que

ocorre no problema físico. O número de Reynolds é dado por:

$$Re = \frac{\rho v D}{\mu} \quad (2.3)$$

onde ρ é a massa específica do fluido, v a velocidade média do fluido, μ a viscosidade dinâmica do fluido e D a dimensão característica do sólido.

Outra variável utilizada nos estudos de transferência de calor por convecção é o número de Prandt, que descreve a razão entre a viscosidade cinemática ν e a difusividade térmica α :

$$Pr = \frac{\nu}{\alpha} = \frac{c_p \mu}{k_{ar}} \quad (2.4)$$

onde c_p é o calor específico e k_{ar} é a condutividade térmica do fluido. Para se encontrar soluções de problemas térmicos, o coeficiente de transferência de calor por convecção h é proposto de acordo com os números de Reynolds e de Prandt, como será visto na seção 3.

2.1.3 Radiação

A radiação por sua vez é equacionada pela *Lei de Stefan Boltzman*:

$$\mathbf{q}_{rad} = \epsilon \sigma T^4 \quad (2.5)$$

onde σ [$W/m^2 K^4$] é a constante de Stefan Boltzman e ϵ [adimensional] é a emissividade - propriedade do material ($0 \leq \epsilon \leq 1$).

2.1.4 Equação do calor

A equação do calor aplicada neste trabalho parte da condução que ocorre no sólido. Para fazer o equacionamento tridimensional da condução considera-se um volume infinitesimal contínuo formado pelo seu domínio Ω e contorno Γ supondo um volume $dV = dx dy dz$ homogêneo e isotrópico.

Conforme a *lei de Fourier* - Equação 2.1 - é possível separar o fluxo de calor em três dimensões conforme Figura 2.2, onde:

- $\mathbf{q}_x = -k \left(\frac{dT}{dx} \right) \mathbf{i}$ - fluxo de calor na direção x, dado em $[W/m^2]$;
- $\mathbf{q}_y = -k \left(\frac{dT}{dy} \right) \mathbf{j}$ - fluxo de calor na direção y, dado em $[W/m^2]$;

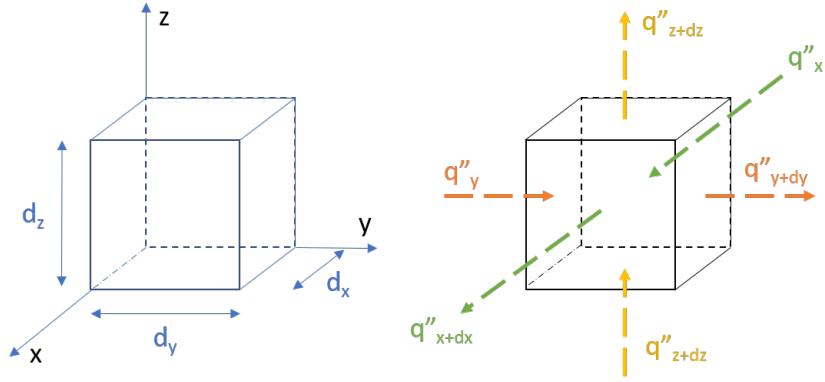


Figura 2.2: volume infinitesimal

Fonte: Elaborada pelo autor

- $\mathbf{q}_z = -k \left(\frac{dT}{dz} \right) \mathbf{k}$ - fluxo de calor na direção z, dado em $[W/m^2]$.

Além disso a taxa de energia em dV é dada por:

$$\rho c \frac{\partial T}{\partial t} [W/m^3] \quad (2.6)$$

onde $c [J/kgK]$ é o calor específico do material, $\rho [kg/m^3]$ a massa específica e $\partial T / \partial t [K/s]$ a variação da temperatura no tempo.

Considerando ainda uma possível geração de calor Q no interior do volume dV , a equação do balanço de energia de dV é dada por:

$$\rho c \frac{\partial T}{\partial t} dV = \Delta \mathbf{q}_x dy dz + \mathbf{q}_y dx dz + \mathbf{q}_z dx dy + Q_\Omega dV \quad (2.7)$$

onde:

- $\Delta \mathbf{q}_x = \mathbf{q}_{x+dx} - \mathbf{q}_x$ - fluxo de calor na direção x, dado em $[W/m^2]$;
- $dy dz$ - área da seção perpendicular ao eixo x, dado em $[m^2]$;
- $\Delta \mathbf{q}_y = \mathbf{q}_{y+dy} - \mathbf{q}_y$ - fluxo de calor na direção y, dado em $[W/m^2]$;
- $dx dz$ - área da seção perpendicular ao eixo y, dado em $[m^2]$;
- $\Delta \mathbf{q}_z = \mathbf{q}_{z+dz} - \mathbf{q}_z$ - fluxo de calor na direção z, dado em $[W/m^2]$;
- $dx dy$ - área da seção perpendicular ao eixo z, dado em $[m^2]$;

Sendo $\frac{\mathbf{q}_{x+dx} - \mathbf{q}_x}{\partial x}$ a derivada parcial de \mathbf{q} em x , temos que:

$$\begin{aligned}\mathbf{q}_{x+dx} - \mathbf{q}_x &= \frac{\partial(\mathbf{q}_x)}{\partial x} \partial x = \frac{\partial}{\partial x} \left(-k \frac{\partial T}{\partial x} \right) dx \\ \mathbf{q}_{y+dy} - \mathbf{q}_y &= \frac{\partial(\mathbf{q}_y)}{\partial y} \partial y = \frac{\partial}{\partial y} \left(-k \frac{\partial T}{\partial y} \right) dy \\ \mathbf{q}_{z+dz} - \mathbf{q}_z &= \frac{\partial(\mathbf{q}_z)}{\partial z} \partial z = \frac{\partial}{\partial z} \left(-k \frac{\partial T}{\partial z} \right) dz\end{aligned}\quad (2.8)$$

Encontra-se então a equação da condução tridimensional

$$\rho c \frac{\partial T}{\partial t} dx dy dz = \left[\frac{\partial}{\partial x} \left(-k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(-k \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left(-k \frac{\partial T}{\partial z} \right) \right] dx dy dz + Q_\Omega dx dy dz \quad (2.9)$$

Define-se o operador *Laplaciano* ∇^2 em coordenadas cartesianas como:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \quad (2.10)$$

Com isso é possível escrever a equação na sua forma diferencial:

$$\rho c \frac{\partial T}{\partial t} = -k \nabla^2 T + Q_\Omega \quad (2.11)$$

onde Q_Ω [W/m^3] é o termo de geração de calor no volume, chamado de termo fonte.

2.2 Sistema de freios

O objetivo de um sistema de freios é desacelerar um veículo para uma velocidade menor ou proporcionar a completa parada sobre as mais diversas condições previstas[3]. O sistema de freios realiza esta tarefa transformando a energia cinética de um objeto em energia térmica. A aplicação de freios é bastante plural, aparecendo em máquinas de fabricação industrial, elevadores e os muitos outros tipos de veículos. Os freios mais famosos são os freios automotivos que, assim como freios da aviação, carregam em si uma responsabilidade enorme de segurança e passa por diversos tipos de estudos e regulamentações. Além de ser um item de segurança, os freios também são estudados no meio da competição como um sistema de performance, uma vez que um carro com freios bem dimensionados tem a capacidade de frear mais perto de uma curva, ganhando então vantagem no percurso com relação a seus adversários. Os sistemas de freio automotivos mais utilizados são do tipo tambor ou a disco, ambos com acionamento hidráulico.

O freio a disco é mais utilizado em motocicletas e carros de alta performance por ser mais eficiente (em relação aos freios a tambor) já que é autolimpante, não acumula água e sujeira por meio do efeito centrífugo e por ser aberto ao mesmo exterior proporcionando uma facilidade maior de manutenção. Além disso a dispersão de calor é maior nos discos, o que causa uma menor propensão a falhas por fadiga térmica. Dada a grande vantagem do freio a disco, até os modelos de carros e motos mais populares já estão contando com esta alternativa.

Quando o motorista aciona o pedal de freios, o mesmo aciona um cilindro mestre, que por sua vez transforma o movimento do pedal em pressão hidráulica tanto no sistema dianteiro quanto no traseiro, que são independentes por motivos de segurança. A pressão que é destinada ao sistema dianteiro e traseiro não é necessariamente igual e depende do dimensionamento e dos cálculos de frenagem do projeto. Quando um veículo freia, ocorre a chamada transferência de carga - somatório de momentos causado pelas forças iniciais - e a transferência de peso - o corpo do passageiro e a gasolina se deslocando mais para a frente do veículo - que fazem o veículo ter a rotação de *Pitch*, como mostra a Figura 2.3. Com isso normalmente os freios dianteiros de um veículo recebem maior carga, uma vez que a carga dinâmica na parte dianteira do carro é maior.



Figura 2.3: Pitch causado pela transferência de carga e de massa na frenagem

Fonte: Elaborada pelo autor.

2.2.1 Transmissão da força do pedal ao sistema de freios

Em veículos de competição a distribuição de pressão hidráulica nos sistemas independentes dianteiro e traseiro geralmente é feita por uma peça chamada Balance Bar, como mostra a Figura 2.4. Esta peça tem um braço de alavanca diferente para cada sistema, B_F e B_R , destinando assim uma força específica de projeto, de forma que a força exercida na balance bar pelo pedal F_{pd} é transformada na força dianteira F_{cilF} e na força traseira F_{cilR} , onde:

$$\begin{aligned} F_{pd} &= F_{cilF} + F_{cilR} \\ F_{cilF} \cdot B_F &= F_{cilR} \cdot B_R \end{aligned} \quad (2.12)$$

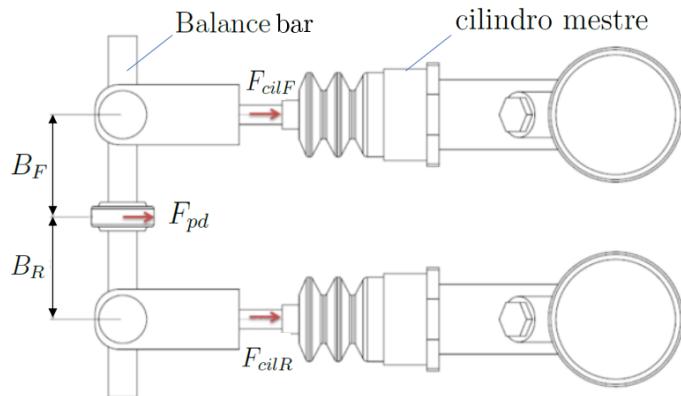


Figura 2.4: Balance bar entre sistema dianteiro e traseiro e cilindros mestre

Fonte: Elaborada pelo autor.

Com isso, é possível alterar os braços de alavanca e mudar a distribuição de pressão nos sistemas. Com as Equações 2.12, pode-se encontrar as seguintes relações:

$$\begin{aligned} F_{cilF} &= \frac{B_R}{B_F + B_R} \cdot F_{pd} \\ F_{cilR} &= \frac{B_F}{B_F + B_R} \cdot F_{pd} \end{aligned} \quad (2.13)$$

É possível, então, nomear a proporção traseira λ :

$$\lambda = \frac{B_F}{B_F + B_R} \quad (2.14)$$

Portanto:

$$\begin{aligned} F_{cilF} &= (1 - \lambda) \cdot F_{pd} \\ F_{cilR} &= \lambda \cdot F_{pd} \end{aligned} \quad (2.15)$$

Com a força exercida em cada haste dos cilindros mestre, estes pressurizam o interior das linhas de freio através de um fluido que pode ser considerado incompressível.

Denominando A_{cm} a área dos pistões dos cilindros mestre, é possível descrever a pressão que cada cilindro mestre faz no interior das linhas de freios.

$$\begin{aligned} P_F &= \frac{(1 - \lambda) \cdot F_{pd}}{A_{cm}} \\ P_R &= \frac{(\lambda) \cdot F_{pd}}{A_{cm}} \end{aligned} \quad (2.16)$$

As linhas de freio, por sua vez, transmitem a pressão até o conjunto das rodas, como mostra a Figura 2.5. Apesar de haver alguma perda de carga nas linhas, esta perda pode ser ignorada para o propósito deste trabalho.

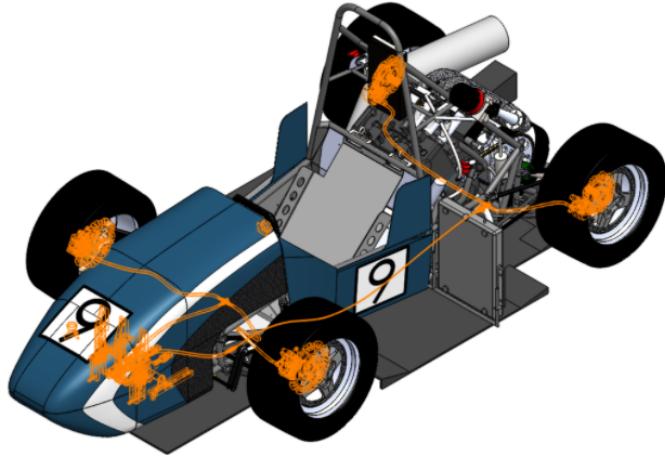


Figura 2.5: Sistema de freios interligado por linhas pressurizadas

Fonte: Elaborada pelo autor.

As linhas então entregam a pressão hidráulica a um componente chamado pinças de freio, como mostra a Figura 2.6.

As pinças têm pistões que transformam a pressão hidráulica em força. Esta força é aplicada sobre as pastilhas de freio, que são feitas de materiais especiais para atritar o disco. Pode-se, portanto, calcular a força N que os pistões da pinça fazem sobre a pastilha:

$$\begin{aligned} N_F &= P_F \cdot A_F \\ N_R &= P_R \cdot A_R \end{aligned} \quad (2.17)$$

Sendo A_F e A_R as áreas dos pistões da pinça dianteira e traseira respectivamente.

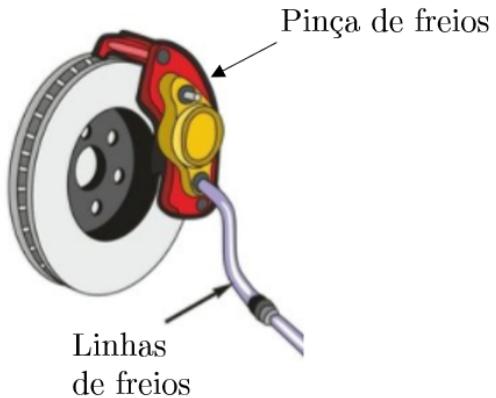


Figura 2.6: Conjunto linha, pinça e disco de freios

Fonte: Elaborada pelo autor

Esta força pressiona as pastilhas de freio contra os discos, gerando uma força de atrito que vai resultar em um torque de frenagem.

As forças de atrito F_{at} , portanto, podem ser calculadas de acordo com N e com os coeficientes de atrito μ entre disco e pastilha.

$$\begin{aligned} F_{atF} &= N_F \cdot \mu_F \\ F_{atR} &= N_R \cdot \mu_R \end{aligned} \tag{2.18}$$

2.2.2 Torque de frenagem

O objetivo da força de atrito das Equações 2.18 é gerar um torque de frenagem no disco. O torque pode então ser calculado:

$$\begin{aligned} T_F &= F_{atF} \cdot R_{efetF} \\ T_R &= F_{atR} \cdot R_{efetR} \end{aligned} \tag{2.19}$$

onde R_{efetF} e R_{efetR} são os raios efetivos dos discos dianteiro e traseiro respectivamente.

Considere agora o diagrama de corpo livre da roda do veículo - Figura 2.7. O somatório de momentos é dado por:

$$\sum M = I\alpha \tag{2.20}$$

onde I é o somatório dos momentos de inércia das partes girantes do ocnjunto (pneu, roda, disco e cubo de roda) e α a aceleração angular do disco de freio. Para

os torques envolvidos no problema, temos:

$$\sum M = -F_x R_P + T \quad (2.21)$$

onde R_P é o raio efetivo do pneu e F_x é a força de atrito entre a pista e o pneu. Portanto, é possível escrever:

$$T = F_x R_P + I\alpha \quad (2.22)$$

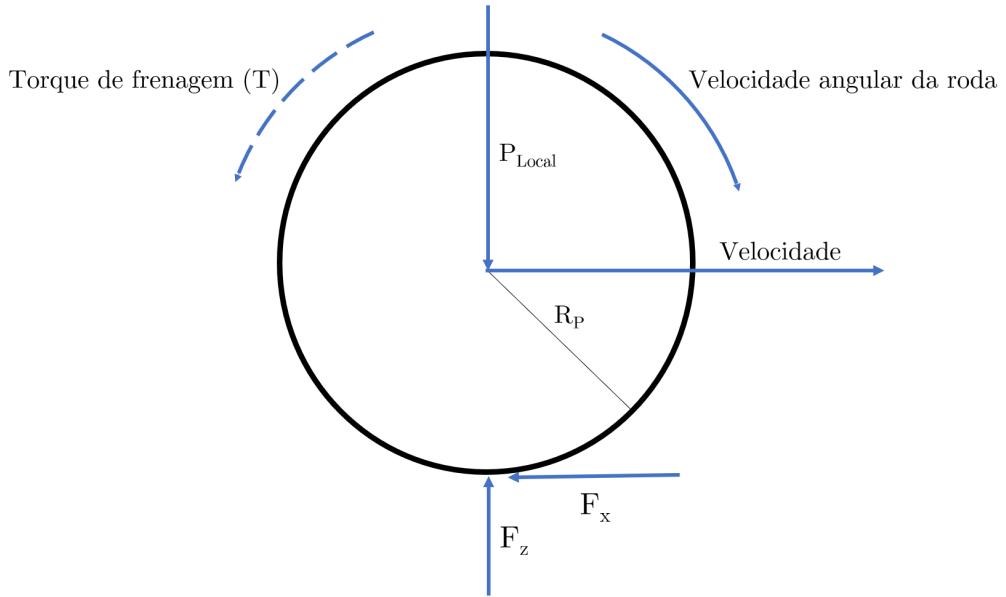


Figura 2.7: Diagrama de corpo livre da roda

Fonte: Elaborada pelo autor

2.2.3 Frenagem ótima

De acordo com demonstrações do Limpert [3], para dimensionar os valores ótimos de frenagem é necessário trabalhar com o diagrama de forças do veículo no momento da frenagem.

Na Figura 2.8 pode-se verificar a ação de forças verticais F_{zR} , F_{zF} e W (peso do veículo) e forças horizontais F_{xR} (força de atrito no eixo traseiro), F_{xF} (força de atrito no eixo dianteiro) e F_a (força de inércia resultante da desaceleração a). Fazendo o equilíbrio de momentos com relação ao eixo dianteiro, obtém-se

$$\sum M_F = WL_F - F_{zR}L - F_a h = 0 \quad (2.23)$$

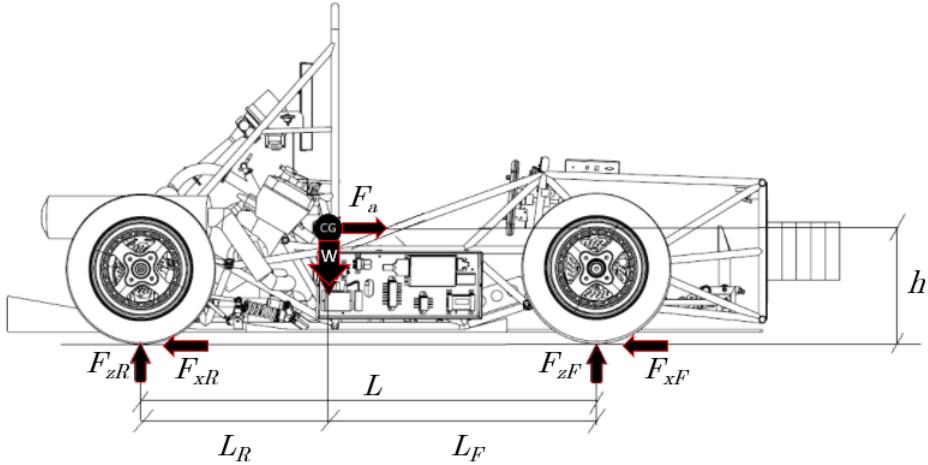


Figura 2.8: Medidas e forças que ocorrem no protótipo no momento da frenagem

Fonte: Elaborada pelo autor

$$F_{zR} = W \frac{L_F}{L} - W a \frac{h}{L} = W \left(\frac{L_F}{L} - a \frac{h}{L} \right) \quad (2.24)$$

Fazendo o equilíbrio de momentos com relação ao eixo traseiro, obtém-se

$$\sum M_R = -WL_R + F_{zF}L - F_a h = 0 \quad (2.25)$$

$$F_{zF} = W \frac{L_R}{L} + W a \frac{h}{L} = W \left(\frac{L_R}{L} + a \frac{h}{L} \right) \quad (2.26)$$

Define-se a razão entre a carga estática traseira F_{zR} e o peso total do carro W como:

$$\Psi = \frac{F_{zR}}{W} = \frac{L_F}{L} \quad (2.27)$$

Logo para a carga estática dianteira, define-se:

$$1 - \Psi = \frac{F_{zF}}{W} = \frac{L_R}{L} \quad (2.28)$$

Além disso, é definida uma relação geométrica do protótipo χ :

$$\chi = \frac{h}{L} \quad (2.29)$$

Substituindo as relações das Equações 2.27, 2.28 e 2.29 nas Equações 2.24 e 2.26, obtém-se:

$$\begin{aligned} F_{zR} &= W(\Psi - a\chi) \\ F_{zF} &= W(1 - \Psi + a\chi) \end{aligned} \quad (2.30)$$

Com isso fica demonstrada matematicamente a transferência de carga no eixo dianteiro durante a frenagem. As forças normais dinâmicas F_z variam linearmente com a desaceleração, quando a dianteira ganha carga e a traseira perde, como mostrado no gráfico da Figura 2.9.

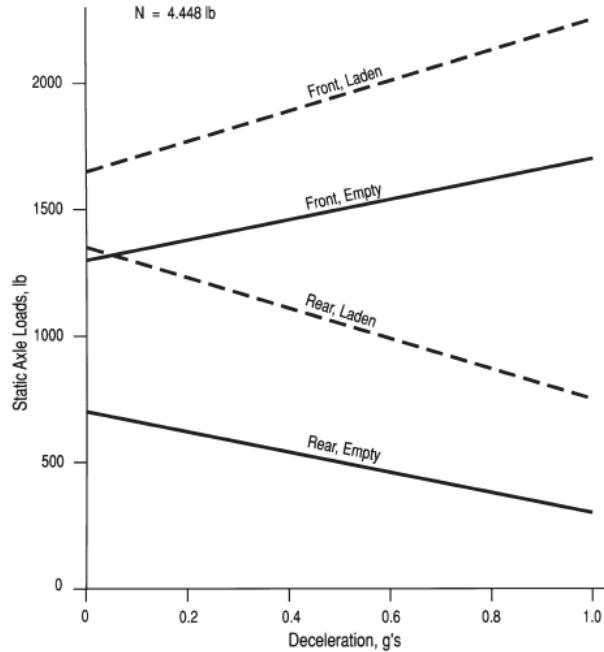


Figura 2.9: Curva relacionando as forças nos eixos dianteiro e traseiro na situação de frenagem de um caminhão vazio e cheio

Fonte: Limpert [3]

A frenagem ocorre de fato entre a pista e o pneu. Ou seja, a força de atrito F_x da Figura 2.7 é que pára efetivamente o veículo. Essa força, por sua vez é limitada. Seu limite superior é dado pelo valor máximo da força de atrito do pneu com a pista, que é dado pelo valor máximo do coeficiente de atrito estático μ_P . Portanto:

$$F_x \leq F_z \cdot \mu_{Pmax} \quad (2.31)$$

O valor máximo do coeficiente de atrito μ_{Pmax} ocorre na iminência de travamento da roda pois, após este momento o atrito dinâmico começa a ocorrer, sendo o valor do coeficiente de atrito dinâmico menor do que o estático, conforme Figura 2.10.

É válido lembrar que na situação de frenagem, a fase estática representa nenhum movimento relativo entre pista e pneu, o que significa o carro em movimento sem

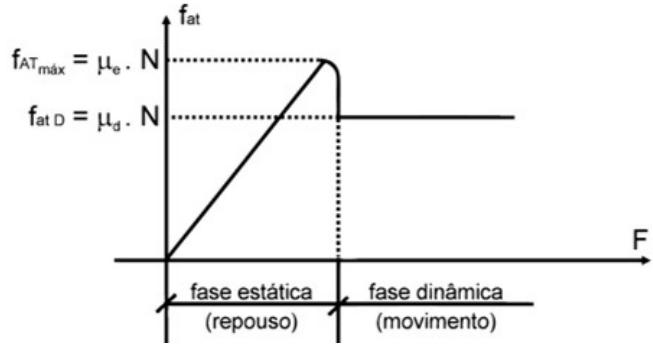


Figura 2.10: Variação no valor de coeficiente de atrito

Fonte: <https://www.proenem.com.br/enem/fisica/forca-de-atrito-e-elastica/>

deslizamento. Já a fase dinâmica é quando ocorre o deslizamento entre pneu e pista.

Define-se frenagem ótima como a desaceleração máxima possível, que ocorre na iminência do deslizamento. Em uma visão geral do veículo a Equação 2.31 pode ser igualada ao somatório de forças na frenagem máxima:

$$F_x = F_z \cdot \mu_{Pmax} = M \cdot a_{max} \quad (2.32)$$

onde M é a massa do veículo.

$$(M \cdot g) \cdot \mu_{Pmax} = M \cdot a_{max} \quad (2.33)$$

Portanto:

$$\mu_{Pmax} = \frac{a_{max}}{g} \quad (2.34)$$

Com isso, como afirmado por Limpert [3], demonstra-se que uma frenagem otimizada em termos de maximizar a desaceleração ocorre quando $a = \mu_{Pmax}$ (a medido em g's - aceleração da gravidade).

Substituindo a Equação 2.30 na Equação 2.31, obtém-se a força de atrito ótima entre pneu e pista:

$$F_{xF-otima} = W(1 - \Psi + a\chi)\mu_{Pmax} \quad (2.35)$$

$$F_{xR-otima} = W(\Psi - a\chi)\mu_{Pmax} \quad (2.36)$$

A eficiência de frenagem é definida como [3]:

$$E_F = \frac{a}{\mu_P} = \frac{1 - \Psi}{1 - \phi - \mu_F\chi} \quad (2.37)$$

$$E_R = \frac{a}{\mu_P} = \frac{\Psi}{\phi + \mu_R \chi} \quad (2.38)$$

onde ϕ é a distribuição de carga entre linha dianteira e traseira do projeto, definido como:

$$\phi = \frac{F_{xR}}{(F_{xF} + F_{xR})} = \frac{\lambda \cdot \frac{A_R}{A_F} \cdot R_{efetR}}{(1 - \lambda) \left[\frac{A_F}{A_{cm}} R_{efetF} \right] + \lambda \left[\frac{A_R}{A_{cm}} R_{efetR} \right]} \quad (2.39)$$

e λ foi definida na Equação 2.14 como a relação de transferência de força para o sistema traseiro dado pela balance bar.

Quando não se trata da frenagem ótima, as Equações 2.35 e 2.36 se tornam:

$$F_{xF} = W(1 - \Psi + a\chi)\mu_{TF} \quad (2.40)$$

$$F_{xR} = W(\Psi - a\chi)\mu_{TR} \quad (2.41)$$

onde μ_T é definido por Limpert [3] como o coeficiente de tração.

$$\mu_T = \frac{F_x}{F_z} \quad (2.42)$$

Note que o conceito de coeficiente de tração é diferente do coeficiente de atrito entre pista e pneu. Os coeficiente de atrito entre pista e pneu são definidos por Limpert [3] como um indicador da habilidade - da possibilidade - da superfície de produzir atrito no pneu. A roda continuará girando enquanto o coeficiente de tração μ_T for menor do que o coeficiente de atrito μ_P .

2.2.4 Curvas de frenagem

Retornando a frenagem ótima, as Equações 2.35 e 2.36 podem ser normalizadas com relação ao peso W do veículo:

$$\frac{F_{xF-otima}}{W} = (1 - \Psi + a\chi)a \quad (2.43)$$

$$\frac{F_{xR-otima}}{W} = (\Psi - a\chi)a \quad (2.44)$$

Avaliando as Equações 2.43 e 2.44 percebe-se uma relação quadrática com a desaceleração. Esta relação permite construir a curva de frenagem ótima, como

mostrado no exemplo de Limpert [3] na Figura 2.11. A ordenada representa a força de frenagem dianteira normalizada, enquanto que a abscissa representa a força traseira normalizada. As retas inclinadas a 45° são as curvas de desaceleração constante (perceba que o ponto $(0.5, 0.1)$ coincide com a curva de desaceleração 0.6, ou seja, qualquer ponto nessa curva tem o somatório $F_{xR} + F_{xF} = 0.6$).

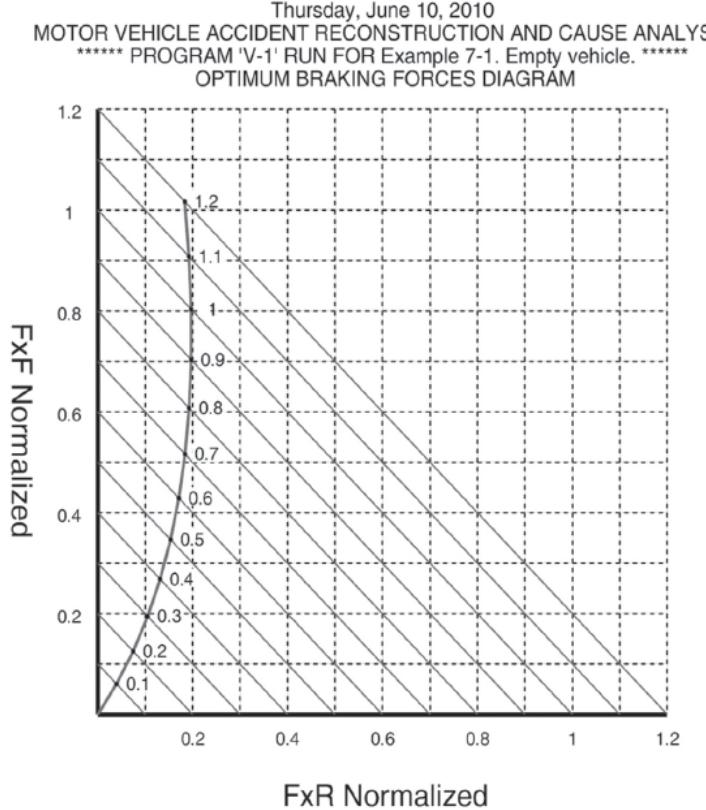


Figura 2.11: Curvas de frenagem normalizadas

Fonte: [3]

Para construir a curva de frenagem ótima do veículo, basta isolar a aceleração na Equação 2.43 e substituir na Equação 2.44, o que resulta em:

$$\frac{F_{xR-otima}}{W} = \sqrt{\frac{(1 - \Psi)^2}{4\chi^2} + \left(\frac{1}{\chi}\right) \left(\frac{F_{xF}}{W}\right)} - \frac{1 - \Psi}{2\chi} - \frac{F_{xF}}{W} \quad (2.45)$$

No gráfico da Figura 2.11 existe uma curva de frenagem ótima variando de $0.1g$ até $1.2g$.

A curva ótima de frenagem do veículo é uma parábola que pode ser traçada no gráfico, como exemplificado por Limpert [3] na Figura 2.12. Vale notar que o ponto extremo da parábola (onde a força normalizada traseira é nula e a força dianteira

vale em torno de 2.6) representa o momento que a frenagem é tão forte, que as rodas traseiras começam a perder contato com a pista - fenômeno que ocorre muito em motocicletas, como mostra a Figura 2.13. Da mesma forma o outro extremo representa a perda de contato do eixo dianteiro em uma aceleração intensa.

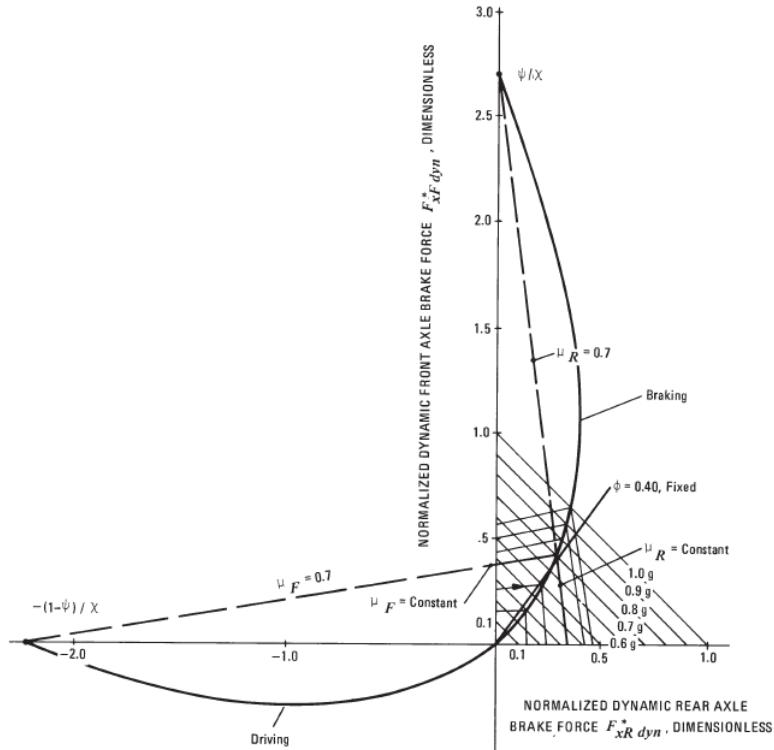


Figura 2.12: Curva de frenagem ótima de um veículo

Fonte: [3]



Figura 2.13: Motocicleta perdendo contato entre eixo traseiro e pista

Fonte: <https://viagemdemoto.com/dicas-para-viagens/3674-como-frear-uma-moto>

2.2.5 A questão térmica

Observa-se que, segundo as Equações 2.21 e 2.31, o torque de frenagem T também é limitado pelo travamento da roda, e não pela força no pedal de freios. Ao considerar a frenagem ótima significa que o torque produzido no disco é o maior possível e portanto a força de atrito nos discos são as maiores que o problema pode proporcionar. Após frenagens bruscas e sucessivas, a temperatura do disco pode se elevar consideravelmente, como mostrado na Figura 2.14.



Figura 2.14: Pinça de freio proporcionando atrito ao disco de freios

Fonte: <https://rizzofisico.wordpress.com/2012/07/04/discos-de-freio-incandescentes/>

Segundo Limpert [3], altas temperaturas trazem redução do coeficiente de atrito entre pastilha e disco. A transferência de calor nos discos de freios se torna, portanto, um assunto crítico no projeto fazendo as montadoras e equipes realizarem testes em bancadas, como mostra a Figura 2.15.

A temperatura elevada nos discos também faz outros componentes terem sua temperatura aumentada. Um dos motivos principais apontados nas causas de acidentes é a elevação exagerada da temperatura do fluido de freio [3]. Com a temperatura elevada, as moléculas de água no fluido viram vapor, tornando a mistura mais compressível e gerando uma perda de pressão nas linhas. Além da vaporização do fluido, a alta temperatura pode causar fadiga térmica, trincas e vibrações excessivas.

O calor resultante da força de atrito entre disco e pastilha não é totalmente absorvido pelo disco. Para descobrir qual a parcela de calor que vai para o disco, é necessário fazer o cálculo do coeficiente de partição de calor do disco [3].

$$\gamma = \frac{q_d}{q_d + q_p} = \frac{1}{1 + \frac{\zeta_p}{\zeta_d}} \quad (2.46)$$



Figura 2.15: Pinça de freio proporcionando atrito ao disco de freios

Fonte: <https://www.kaidon-brake.com/what-happens-to-the-brake-discs-at-high-temperatures>

onde ζ_d e ζ_p são as efusividades térmicas, que são calculadas como mostra a Equação 2.47

$$\zeta = \sqrt{k\rho c} \quad (2.47)$$

Portanto, a taxa de calor \dot{E} produzida em um conjunto de freios durante a força de atrito da frenagem é dada por [4]:

$$\dot{E} = \dot{E}_p + \dot{E}_d = (1 - \gamma)\dot{E} + \gamma\dot{E} \quad (2.48)$$

$$\dot{E} = VFat \quad (2.49)$$

2.3 Método de elementos finitos

Neste capítulo, serão abordados as questões teóricas e explicações da aplicação do método. Entretanto para um entendimento mais sólido, é indicada a leitura do Capítulo 3.1, onde os passos são efetivamente aplicados no problema proposto.

Segundo *Logan* [5], o método de elementos finitos tornou-se realmente prático na solução de problemas de engenharia a partir do anos 50 juntamente com o desenvolvimento de computadores com maior capacidade de processamento. As publicações de Hrennikoff em 1941 e McHenry em 1943 trouxeram a primeira aplicação estrutural na solução unidimensional de tensões em vigas e barras. No mesmo período

um paper de Courant foi publicado abordando a solução de tensões na forma variacional. Nesse trabalho ele propôs a utilização de funções interpoladoras em regiões triangulares de forma que toda a região possa obter uma solução numérica aproximada. Já em 1947, interessado em resolver problemas complexos em estrutura de aeronaves, Levy sugeriu o método de rigidez e deslocamento. Entretanto esse método exigia uma enorme capacidade de cálculo, que só foi satisfeita anos depois com o avanço tecnológico. A primeira aplicação realmente bidimensional veio em 1956 com Turner ao derivar matrizes de rigidez para elementos triangulares e retangulares (funcionando como treliças) no intuito de obter uma matriz de rigidez para toda a estrutura, método assim conhecido como *direct stiffness method*. O termo "elemento finito" veio em 1960 com Clough que utilizou ambos os elementos triangulares e retangulares na análise de tensão plana.

Até este momento os projetos que utilizavam o método de elementos finitos se utilizavam da premissa de pequenas deformações e deslocamentos. Contudo, em 1960 Turner foi o primeiro a considerar grandes deflexões e principalmente, a desenvolver a aplicação em estudos térmicos. Conforme classificação feita por *Logan* [5] as aplicações do MEF podem ser tanto estruturais quanto não estruturais. As aplicações estruturais típicas são:

- Estudos de tensão, incluindo análises de concentradores de tensão em furos, filetes e as mais diversas variações de geometria;
- Estudos de flambagem;
- Estudos de vibração.

Já as aplicações não estruturais são:

- Transferência de calor;
- Escoamento de fluidos;
- Estudos de potencial elétrico e magnético.

Além disso existem aplicações biomecânicas, como análises de coluna espinhal, crânio, coração, dentre outros.

A ideia por trás do método de elementos finitos é a simulação de um problema através da modelagem de partes discretas que, unidas, representam um meio contínuo, ou seja, através de uma discretização do modelo. Portanto, com o objetivo de viabilizar a solução computacional, a modelagem é feita dividindo a geometria grande e complexa em pequenas partes – denominadas *elementos* que podem ser de diferentes formas como mostrado na figura 2.16.

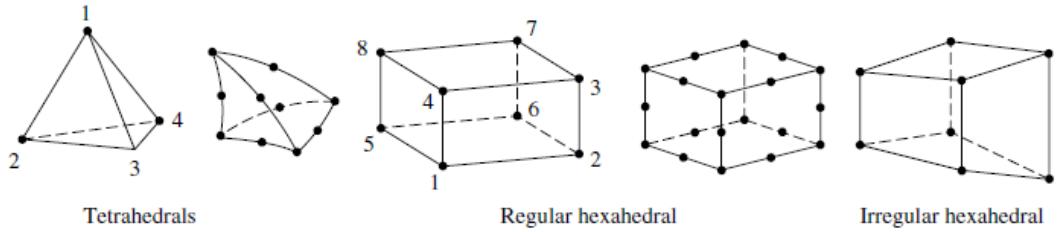


Figura 2.16: Elementos tridimensionais e seus nós

Fonte: [5]

O conjunto dos *elementos* e de seus pontos de contorno – denominados nós – é conhecido como *malha* como mostra a Figura 2.17. A *malha*, portanto, é composta por um número finito de *elementos* de comportamento bem definido. *Logan* [5] define que todo *elemento* é conectado direta ou indiretamente a todos os outros através de seus *nós*, e que um *nó* influencia no outro dando assim o comportamento da estrutura. O MEF é um método poderoso para discretização de geometrias complexas pois não exige esforço computacional adicional comparado a sua utilização em geometrias regulares e é também um método fácil de se incluir uma análise dinâmica.

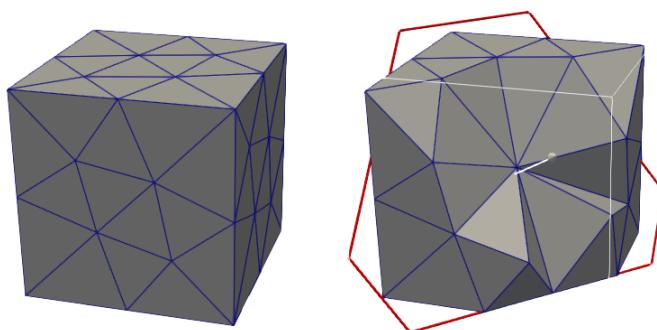


Figura 2.17: Malha de um cubo com elementos tetraédricos

Fonte: Elaborada pelo autor

O MEF parte de uma equação diferencial que descreve o problema. As equações diferenciais são divididas na literatura entre a forma fraca e a forma forte. Ambas têm soluções equivalentes - ou seja, a solução para uma delas representa a solução para a outra. A forma forte, ou forma diferencial, consiste na representação do problema através de um sistema de equações diferenciais, parciais ou ordinárias, no espaço ou no tempo, como a demonstrada na Equação 2.11. Problemas simples podem ser resolvidos analiticamente na forma forte através de técnicas de integração, porém problemas mais complexos podem ter soluções muito difíceis de serem obtidas analiticamente, se tornando uma tarefa árdua e pouco eficiente, sendo assim indicada a aplicação de métodos precisos para obtenção de soluções aproximadas como o método de elementos finitos. A forma fraca consiste na representação do problema através de uma equação integral ponderada, que é definida como uma equação que, a partir do método de solução, distribui "pesos" a determinados elementos da expressão. O termo "forma fraca" diz respeito ao fato de a equação ser resolvida adotando uma média em vez de ser solucionada ponto a ponto como na forma forte.

A aplicação do MEF se inicia justamente na transformação da equação diferencial na sua forma forte $f''(x, y, z)$ para a forma fraca. Para realizar este procedimento, é necessário integrar sua equação no domínio multiplicando por uma função peso $\omega(x, y, z)$:

$$\int_{\Omega} \omega(x, y, z) f''(x, y, z) d\Omega \quad (2.50)$$

Após isso é necessário fazer a integração por partes pelo teorema de green para se obter a redução de ordem e separar os termos do contorno Γ . O teorema de green para campos vetoriais é dado por:

$$\int_{\Omega} \phi \nabla^2 \psi d\Omega = \int_{\Gamma} \phi \nabla \psi \cdot \mathbf{n} d\Gamma - \int_{\Omega} \nabla \phi \cdot \nabla \psi d\Omega \quad (2.51)$$

Além disso aplica-se um método de resíduos ponderados para se encontrar a melhor aproximação para a função de interesse.

2.3.1 Método de resíduos ponderados

Os métodos de resíduos ponderados são definidos como procedimentos capazes de achar soluções aproximadas para equações diferenciais com um bom grau de precisão

[6]. Trata-se de encontrar uma função \mathbf{F} que aproxima a solução exata no intervalo minimizando o erro - aqui chamado de resíduo - da aproximação. A função \mathbf{F} é dada por meio do somatório de N funções de forma ϕ_i com amplitudes arbitrárias a_i .

$$\mathbf{F} = \sum_{i=1}^N a_i \phi_i \quad (2.52)$$

Como esses passos estão sendo aplicados para cada elemento particular da malha, essas funções de forma têm as seguintes características:

1. O valor da função em seu respectivo nó vale 1, e nos outros nós tem valor nulo;
2. A função entre os nós pode ser linear, quadrática, cúbica ou até de graus maiores;
3. A função apresenta continuidade no interior de cada elemento, mas não na fronteira dos nós.

Ao substituir \mathbf{F} na equação diferencial original um resíduo R é encontrado já que \mathbf{F} não é exatamente igual a solução exata.

Por exemplo, para a equação diferencial $\frac{df}{dx} - f = 0$ com condição de contorno $f(0) = 1$ propõe-se um polinômio do terceiro grau:

$$\mathbf{F} = a_1 x^3 + a_2 x^2 + a_3 x + 1$$

onde x^3 , x^2 e x são as funções de forma.

Substituindo \mathbf{F} na equação diferencial obtém-se:

$$\frac{d(a_1 x^3 + a_2 x^2 + a_3 x + 1)}{dx} - (a_1 x^3 + a_2 x^2 + a_3 x + 1) = R(x)$$

Não é igualado a zero, mas sim ao resíduo $R(x)$. Portanto:

$$3a_1 x^2 + 2a_2 x + a_3 - a_1 x^3 - a_2 x^2 - a_3 x - 1 = R(x)$$

Após isso, escolhe-se pontos arbitrários onde o resíduo será nulo. Por exemplo, escolhendo $x = 0,25$, $x = 0,50$ e $x = 0,75$ obtém-se um sistema linear para as amplitudes a_i que resulta em:

$$\mathbf{F} = 0,28x^3 + 0,42x^2 + 1,03x + 1$$

A comparação dos resultados dessa função com a solução exata ($f = e^x$) no intervalo de 0 a 1 dão um erro menor que 0,5%.

A minimização deste resíduo R é dada pela integração do mesmo no domínio Ω e multiplicando-o por uma função de ponderação Ψ_j .

$$\int_{\Omega} R \cdot \Psi_j d\Omega = 0, \text{ para } j = 1, \dots, N \quad (2.53)$$

Existem diversos tipos de métodos de resíduos ponderados. O mais utilizado no MEF é o método de Galerkin.

2.3.2 Método de Galerkin

Boris Grigorievitch Galerkin, nascido em 1871, foi um engenheiro e matemático formado pelo Instituto Tecnológico de St. Peterburgo. Ao se envolver fervorosamente com a política em um período conturbado, acabou sendo preso por um período de aproximadamente dois anos, quando teve a oportunidade de desenvolver seu conhecimento científico. Em 1915 Galerkin publicou um trabalho que propunha um método de integração aproximada para equações diferenciais parciais, que ficou conhecido como Método de Galerkin. Este trabalho trouxe grande vantagem sobre o já proposto método de Ritz uma vez que era uma regra mais abrangente.

Com isso, Galerkin desenvolvia o método de resíduos ponderados mais útil para o MEF. A particularidade do método de Galerkin está no fato de que ele escolhe a função de ponderação Ψ_j exatamente igual a função de forma ϕ_i :

$$\Psi_j = \phi_j \quad (2.54)$$

Com isso, substituindo a equação 2.54 na equação 2.53 obtém-se a equação do método de Galerkin:

$$\int_{\Omega} R \cdot \phi_i d\Omega = 0, \text{ para } i = 1, \dots, n \quad (2.55)$$

Para uma melhor compreensão do método, é necessário o conhecimento da definição de espaço de funções.

Noção de espaço de funções

O espaço de funções pode ser explicado como uma analogia ao espaço vetorial que tem as seguintes definições:

- Base: espaço vetorial de três dimensões, V^3 , pode ser dado pelos vetores unitários $e_1 = [1 0 0]$, $e_2 = [0 1 0]$ e $e_3 = [0 0 1]$.
- Combinação linear: qualquer elemento pertencente a este espaço é representado pela combinação linear dos elementos da base.

$$v = \alpha_1 e_1 + \alpha_2 e_2 + \alpha_3 e_3$$

- Base é Linearmente Independente (ou ortogonais): é impossível descrever qualquer elemento da base como combinação dos outros elementos. Isso é descrito matematicamente como o resultado nulo do produto interno:

$$\langle e_1, e_2 \rangle = 0, \quad \langle e_1, e_3 \rangle = 0, \quad \langle e_2, e_3 \rangle = 0 \quad (2.56)$$

- Base é ortonormal: além de serem ortogonais, a norma dos vetores da base também é unitária:

$$\|e_1\| = \|e_2\| = \|e_3\| = 1 \quad (2.57)$$

Com isso, um espaço é formado pela combinação linear dos elementos de sua base. A construção de um espaço de funções é análoga pois as funções da base são ortogonais entre si. O que muda é a noção de ortogonalidade de funções. Por exemplo, considere as funções g e h definidas no domínio Ω , o produto interno das duas é dado por:

$$\langle g, h \rangle = \int_{\Omega} g h d\Omega \quad (2.58)$$

Da mesma forma, a representação dos elementos de um espaço de funções é dada pela combinação linear dos elementos de sua base. Portanto, com as definições da Equação 2.58 e da Equação 2.56, é possível construir um espaço de funções.

No contexto do método de Galerkin, a analogia com o espaço vetorial se faz muito útil ao entendimento. Como descrito em [6], quando é necessário se obter uma aproximação de um vetor \mathbf{v} de dimensão $(N+1)$ em um espaço de dimensão N , uma projeção do vetor é feita, como mostra a Figura 2.18.

Com isso, a melhor aproximação do vetor \mathbf{v} em N é \mathbf{u} . A diferença entre \mathbf{v} e \mathbf{u} é ortogonal ao espaço N . A mesma ideia se aplica no método de Galerkin. No caso a diferença entre a função \mathbf{F} e a função exata f é o erro $R = f - \mathbf{F}$. Como o produto interno entre funções é dado pela Equação 2.58, isso significa que o produto interno

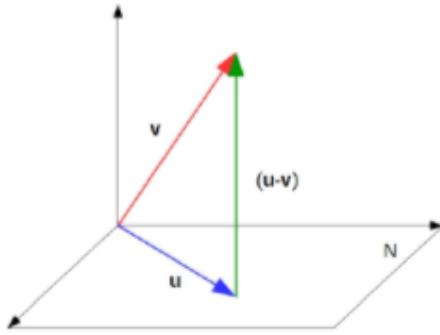


Figura 2.18: Projeção de um vetor em um espaço de dimensão inferior

Fonte: https://www.researchgate.net/publication/295546402_Uma_Introducao_ao_Metodo_dos_Residuos_Ponderados

entre o erro R e todas as funções de forma tem que ser nulo afim de minimizar o erro, como mostra a Equação 2.59. Com isso, na Equação 2.53, a função de forma toma o lugar da função de ponderação, o que define a igualdade $\Psi_j = \phi_j$.

$$\int_{\Omega} R \cdot \phi_i d\Omega = 0, \text{ para } i = 1, \dots, n \quad (2.59)$$

Com isso, de acordo com a Equação 2.52, a variável de interesse $f''(x, y, z)$ da Equação 2.50 é aproximada pelas funções de forma $\phi(x, y, z)$ e a função peso $\omega(x, y, z)$ é aproximada pelas funções de ponderação $\Psi(x, y, z)$.

$$\begin{aligned} f(x, y, z) &\approx \hat{f} = \sum_0^n a_i \phi(x, y, z) \\ \omega(x, y, z) &\approx \hat{\omega} = \sum_0^n b_j \Psi(x, y, z) \end{aligned} \quad (2.60)$$

onde $\Psi(x, y, z) = \phi(x, y, z)$.

2.3.3 Matriz do elemento

Após esses passos feitos, a equação chega a uma configuração onde é possível identificar a construção matricial, onde o lado esquerdo do problema é definido pelas matrizes de massa m^e e de rigidez k^e de cada elemento.

Um elemento tetraédrico possui seus vértices i, j, k e l , como mostra a Figura 2.19.

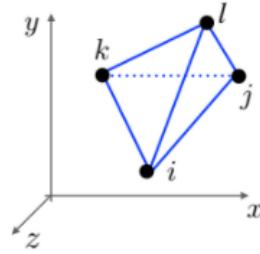


Figura 2.19: Vértices do elemento tetraédrico

Fonte: Anjos [7]

Segundo Anjos [7], a matriz de massa deste elemento pode ser calculada como:

$$m^e = \frac{V}{20} \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \quad (2.61)$$

onde o volume V é calculado da seguinte forma:

$$V = \frac{1}{6} \cdot \det \begin{bmatrix} 1 & x_i & y_i & z_i \\ 1 & x_j & y_j & z_j \\ 1 & x_k & y_k & z_k \\ 1 & x_l & y_l & z_l \end{bmatrix} \quad (2.62)$$

sendo x , y e z as coordenadas dos respectivos vértices.

A matriz de rigidez de um elemento tetraédrico de um material isotrópico com condutividade térmica k constante pode ser calculada como [7]:

$$k^e = V k B^T B \quad (2.63)$$

onde a matriz B é definida como:

$$B = \frac{1}{6V} \begin{bmatrix} b_i & b_j & b_k & b_l \\ c_i & c_j & c_k & c_l \\ d_i & d_j & d_k & d_l \end{bmatrix} \quad (2.64)$$

Os valores da matriz B são calculados da seguinte forma [7]:

$$\begin{aligned}
b_i &= (y_j - y_l)(z_k - z_l) - (y_k - y_l)(z_j - z_l) \\
b_j &= (y_k - y_l)(z_i - z_l) - (y_i - y_l)(z_k - z_l) \\
b_k &= (y_i - y_l)(z_j - z_l) - (y_j - y_l)(z_i - z_l) \\
b_l &= -(b_i + b_j + b_k) \\
c_i &= (x_k - x_l)(z_j - z_l) - (x_j - x_l)(z_k - z_l) \\
c_j &= (x_i - x_l)(z_k - z_l) - (x_k - x_l)(z_i - z_l) \\
c_k &= (x_j - x_l)(z_i - z_l) - (x_i - x_l)(z_j - z_l) \\
c_l &= -(c_i + c_j + c_k) \\
d_i &= (x_j - x_l)(y_k - y_l) - (x_k - x_l)(y_j - y_l) \\
d_j &= (x_k - x_l)(y_i - y_l) - (x_i - x_l)(y_k - y_l) \\
d_k &= (x_i - x_l)(y_j - y_l) - (x_j - x_l)(y_i - y_l) \\
d_l &= -(d_i + d_j + d_k)
\end{aligned} \tag{2.65}$$

Em problemas tridimensionais é necessário ainda ter as matrizes dos elementos bidimensionais para aplicação das condições de contorno.

A matriz de massa do elemento de contorno triangular é calculada da seguinte maneira [7]:

$$m^{ec} = \frac{A}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{2.66}$$

onde, se a superfície de contorno está inteiramente na mesma coordenada Z , a área A do triângulo de contorno pode ser calculada como:

$$A = \frac{1}{2} \cdot \det \begin{bmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{bmatrix} \tag{2.67}$$

2.3.4 Assembling

Portanto, após transformar a expressão da forma forte para a forma fraca, reduzir a ordem através do teorema de Green e substituir as aproximações da Equação 2.60 chega-se a um sistema matricial do tipo $Ax = b$ para cada elemento da malha.

Para englobar todo o domínio físico do problema, é necessário fazer a montagem das matrizes de cada elemento para formar a matriz global. Este procedimento é conhecido como *Assembling*. A Figura 2.20 mostra um exemplo do *Assembling* unidimensional.

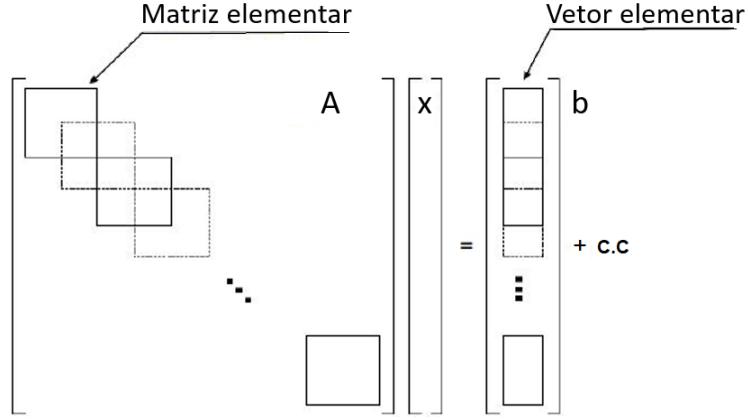


Figura 2.20: Montagem das matrizes elementares em matrizes globais

Fonte: [7]

Após o *Assembling*, chegamos a um sistema linear matricial do tipo:

$$A_{ij}x_i = b_i \quad (2.68)$$

Para realizar o *Assembling*, é necessário ter uma matriz de conectividade, denominada IEN.

2.3.5 Matriz de conectividade

Para realizar uma análise por MEF, é necessário que os elementos da malha sejam reconhecidos, portanto, seus nós têm que ser localizados. Para isto, a matriz de conectividade IEN é montada.

A matriz IEN é uma matriz em que cada linha representa um elemento da malha. Cada valor de uma dada linha representa os índices dos nós daquele elemento. Por exemplo, para os dois elementos da Figura 2.21, a matriz de conectividade é:

Elemento	Nó 1	Nó 2	Nó 3	Nó 4
1(azul)	1	2	3	4
2(verde)	4	3	1	5

Tendo o problema montado, torna-se necessário definir as condições de contorno.

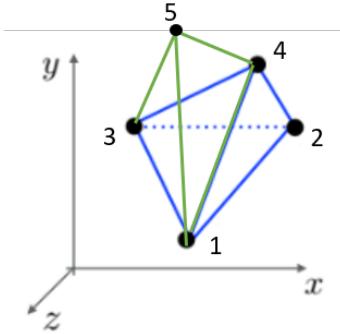


Figura 2.21: Montagem das matrizes elementares em matrizes globais

Fonte: [7]

2.3.6 Condições de contorno

Para a solução de um problema físico, é necessário o conhecimento das condições de contorno daquele problema, ou seja, um conjunto de restrições adicionais que definem a situação dos limites físicos do objeto estudado. As condições de contorno no estudo de transferência de calor são divididas em três tipos: Dirichlet (1º tipo), Neumann (2º tipo) e Robin (3º tipo). Esses tipos podem ser misturados em um mesmo problema.

As condições de contorno são definidas em 1 dimensão menor do que o problema, ou seja, em um problema tridimensional a condição de contorno diz respeito a um plano (2D). As definições de cada condição são as seguintes [7]:

1. Dirichlet - As condições do 1º tipo são quando a variável de interesse - no caso térmico, a temperatura - é definida na fronteira. Por exemplo: $T_{parede} = a_1^0 \text{C}$;
2. Neumann - As condições do 2º tipo são quando a derivada (fluxo) da variável de interesse é definida na fronteira. Por exemplo: $k \frac{dT_{parede}}{dx} = a_2 = q$;
3. Robin - As condições do 3º tipo são a combinação linear do 1º e 2º tipos. Por exemplo: $T_{parede} + k \frac{dT_{parede}}{dx} = a_3$.

É importante destacar que no equacionamento do MEF, a condição de contorno de Dirichlet gera uma função peso nula no contorno, portanto $\omega|_{\Gamma} = 0$. Para a condição de contorno de Neumann homogêneo ($a_2 = 0$), o termo de contorno também é zerado, porém para Neumann não homogêneo ($a_2 \neq 0$) e Robin a função peso não é nula.

2.3.7 Sistema linear

Com as condições de contorno aplicadas na matriz global, encontra-se um sistema linear matricial do tipo:

$$A_{ij}x_i = b_i + c.c.s \quad (2.69)$$

A solução do problema resulta no cálculo final, onde encontram-se os valores da variável de interesse para todos os nós da malha estudada.

A precisão do método de elementos finitos (MEF) depende da quantidade de nós e elementos, do tamanho dos elementos da malha e do tipo de funções de forma considerados (lineares, quadráticas, cúbicas, etc). Ou seja, quanto menor for o tamanho e maior for o número dos elementos em uma determinada malha, mais próximo da realidade está a representação e maior a precisão nos resultados da análise.

Neste trabalho, o método de elementos finitos é utilizado, portanto, na discretização espacial dos problemas. Para a discretização temporal na solução dos problema transientes, o método de diferenças finitas se faz mais prático.

2.4 Método de diferenças finitas

O método de diferenças finitas (MDF), assim como o MEF propõe soluções aproximadas em equações diferenciais, aproximando derivadas por diferenças finitas. A derivada em questão pode ser aproximada de três formas:

- Diferenças progressivas:

$$f'(t) = \frac{f(t+s) - f(t)}{s} + \text{erro}$$

- Diferenças regressivas

$$f'(t) = \frac{f(t) - f(t-s)}{s} + \text{erro}$$

- Diferenças centradas.

$$f'(t) = \frac{f(t+s) - f(t-s)}{2s} + \text{erro}$$

onde s é o passo de tempo.

Capítulo 3

Metodologia

3.1 Método de Elementos Finitos

Como descrito na seção 2.3, a aplicação do MEF tem um passo a passo bem definido. Os passos a seguir descrevem de forma objetiva como será aplicado o método.

3.1.1 Sequência de passos para aplicação do método

1. Desenvolver a equação diferencial do problema na sua forma forte;
2. Transformar a expressão da forma forte para a forma fraca, como na Equação 2.50;
3. Utilizar integração por partes - Equação 2.51 - para atingir a redução da ordem da expressão e separar os termos de contorno;
4. Uso do método de aproximação de funções (método de Galerkin) de acordo com a Equação 2.60 para montar as matrizes elementares;
5. Montagem das matrizes elementares para compor as matrizes globais - Assembling mostrado na Seção 2.3.4;
6. Imposição das condições de contorno como descrito na Seção 2.3.6;
7. Solução do sistema linear encontrado, como na Equação 2.69.

3.1.2 Aplicação em um problema tridimensional transiente com condição de contorno de Dirichlet

1. Descrição na forma forte

A forma forte da Equação do calor corresponde à Equação 2.11 encontrada na seção 2.1.4, que também pode ser escrita da seguinte forma:

$$\frac{\partial(\rho c_v T)}{\partial t} = k \nabla^2 T + Q_\Omega \quad (3.1)$$

onde consideram-se constantes as propriedades - ρ e c_v - do material.

2. Multiplicando a Equação 3.1 pela função peso w e integrando no domínio Ω :

$$\int_{\Omega} \omega \left[\frac{\partial(\rho c_v T)}{\partial t} - k \nabla^2 T - Q \right] d\Omega = 0 \quad (3.2)$$

$$\int_{\Omega} \omega \frac{\partial(\rho c_v T)}{\partial t} d\Omega - \int_{\Omega} k \omega \nabla^2 T d\Omega - \int_{\Omega} \omega Q d\Omega = 0 \quad (3.3)$$

3. Pela identidade de Green - Equação 2.51 - o segundo termo da expressão é:

$$\int_{\Omega} k \omega \nabla^2 T d\Omega = \int_{\Gamma} k \omega \nabla T d\Gamma - \int_{\Omega} k \nabla \omega \cdot (\nabla T) d\Omega \quad (3.4)$$

Portanto, obtém-se:

$$\int_{\Omega} \omega \frac{\partial(\rho c_v T)}{\partial t} d\Omega - \int_{\Gamma} k \omega \nabla T d\Gamma + \int_{\Omega} k \nabla \omega \cdot \nabla T d\Omega - \int_{\Omega} \omega Q d\Omega = 0 \quad (3.5)$$

Para a condição de contorno de Dirichlet, a função peso deve ser nula no contorno:

$$\int_{\Gamma} k \omega^0 \nabla T d\Gamma = 0$$

4. Para cada elemento da malha é usada o método de resíduos ponderados de Galerkin:

$N_i(x, y, z) = N_j(x, y, z)$, sendo:

$$T(x, y, z) = \sum_{i=0}^{n-1} N_i(x, y, z) a_i$$

$$\omega(x, y, z) = \sum_{j=0}^{n-1} N_j(x, y, z) b_i$$

$$Q(x, y, z) = \sum_{k=0}^{n-1} N_i(x, y, z) Q_i$$

Após fazer a substituição, obtém-se:

$$\begin{aligned} \int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (\rho c_v) N_i N_j \frac{da_i}{dt} d\Omega + \int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} k \nabla N_i \cdot \nabla N_j d\Omega a_i = \\ = \int_{\Omega} \sum_{j=0}^{n-1} \sum_{i=0}^{n-1} N_i N_j Q_i d\Omega \end{aligned} \quad (3.6)$$

5. Matrizes dos elementos e montagem da matriz global (*Assembling*)

Na Equação 3.6, o segundo termo pode ser escrito como:

$$\int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} k \nabla N_i \cdot \nabla N_j d\Omega = \int_{\Omega} k \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega$$

onde

$$k \frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} = kx_{ij} \quad ; \quad k \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} = ky_{ij}$$

$$k_{ij} = kx_{ij} + ky_{ij} \quad (3.7)$$

é definida como a matriz de rigidez do elemento.

Integrando essa matriz em todo o domínio, realiza-se o *Assembling*, onde encontra-se a matriz de rigidez global K_{ij} .

$$\int_{\Omega} k_{ij} d\Omega = \int_{\Omega} (kx_{ij} + ky_{ij}) d\Omega = K_{ij} \quad (3.8)$$

Da mesma maneira, temos que a matriz global de massa M_{ij} do problema é definida como:

$$\int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} N_i N_j d\Omega = M_{ij} \quad (3.9)$$

Com essas definições, pode-se escrever a equação 3.6 como:

$$(\rho c_v) M_{ij} \frac{da_i}{dt} + K_{ij} a_i = M_{ij} Q_i \quad (3.10)$$

Observação: Para a discretização temporal $\frac{da_i}{dt}$ utiliza-se o método de diferenças finitas progressivo:

$$(\rho c_v) M_{ij} \left(\frac{a_i^{n+1} - a_i^n}{\Delta t} \right) + K_{ij} a_i = M_{ij} Q_i \quad (3.11)$$

Para este passo, pode-se utilizar três métodos distintos:

- Implícito - define o termo $K_{ij}a_i$ no tempo $(n+1)$:

$$\left(\frac{(\rho c_v) M_{ij}}{\Delta t} + K_{ij} \right) a_i^{n+1} = \frac{(\rho c_v) M_{ij}}{\Delta t} a_i^n + M_{ij} Q_i$$

- Explícito - define o termo $K_{ij}a_i$ no tempo (n) :

$$\frac{(\rho c_v) M_{ij}}{\Delta t} a_i^{n+1} = \left(\frac{(\rho c_v) M_{ij}}{\Delta t} - K_{ij} \right) a_i^n + M_{ij} Q_i$$

- Crank-Nicolson - pondera os dois métodos:

$$\left(\frac{(\rho c_v) M_{ij}}{\Delta t} + \frac{1}{2} K_{ij} \right) a_i^{n+1} = \left(\frac{(\rho c_v) M_{ij}}{\Delta t} - \frac{1}{2} K_{ij} \right) a_i^n + M_{ij} Q_i$$

Por questão de facilidade de implementação de algoritmos, escreve-se:

$$\left(\frac{(\rho c_v) M_{ij}}{\Delta t} + \theta K_{ij} \right) a_i^{n+1} = \left[\frac{(\rho c_v) M_{ij}}{\Delta t} - (1 - \theta) K_{ij} \right] a_i^n + M_{ij} Q_i \quad (3.12)$$

Sendo:

- $\theta = 1$ para o método implícito;
- $\theta = 0$ para o método explícito;
- $\theta = 0.5$ para Crank-Nicolson.

6. Aplicando as condições de contorno

O lado esquerdo do problema contém a matriz A multiplicando pela incógnita a_i^{n+1} do problema, enquanto o lado direito, que é composto de valores já conhecidos, é chamado de vetor b.

O problema, portanto, pode receber as condições de contorno. Escreve-se:

$$\left(\frac{(\rho c_v) M_{ij}}{\Delta t} + \theta K_{ij} \right) a_i^{n+1} = \left[\frac{(\rho c_v) M_{ij}}{\Delta t} - (1 - \theta) K_{ij} \right] a_i^n + M_{ij} Q_i + c.c. \quad (3.13)$$

7. Com isso é possível resolver o sistema linear $A_{ij}x_i = b_i$ encontrado.

3.1.3 Aplicação em um problema tridimensional transiente com condição de contorno de Neumann não homogêneo e constante

Com o intuito de incluir na simulação o fluxo de calor constante fornecido ao disco de freio pelas pastilhas - devido a força de atrito da frenagem - considera-se a condição de contorno do 2º tipo não homogênea e constante.

Para essa condição de contorno, a Equação 3.5 não tem a função peso ω nula. Portanto, após aplicação do método de Galerkin nos termos do domínio, obtém-se:

$$\begin{aligned} \int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (\rho c_v) N_i N_j \frac{da_i}{dt} d\Omega + \int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} k \nabla N_i \cdot \nabla N_j d\Omega a_i = \\ = \int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} N_i N_j Q_i d\Omega + k \int_{\Gamma_q} \omega \nabla T d\Gamma_q \end{aligned} \quad (3.14)$$

O contorno Γ_q se refere a superfície do disco que tem contato com a pastilha. Como mostrado na Equação 3.9 o termo que multiplica as funções de forma representa a matriz global de massa M_{ij} e na Equação 3.8 o termo que multiplica as derivadas das funções de forma representa a matriz global de rigidez K_{ij} . Com isso, obtém-se:

$$(\rho c_v) M_{ij} \frac{da_i}{dt} + K_{ij} a_i = M_{ij} Q_i + k \int_{\Gamma_q} \omega \nabla T d\Gamma_q \quad (3.15)$$

O termo ∇T multiplicado por k é exatamente o fluxo de calor por área recebido pelo disco devido ao contato com a pastilha descrito pela *lei de Fourier* - Equação 2.1 - como segue:

$$k \left[\frac{W}{mK} \right] \cdot \nabla T \left[\frac{K}{m} \right] = q \left[\frac{W}{m^2} \right] \quad (3.16)$$

Com isso, a Equação 3.15 fica:

$$(\rho c_v) M_{ij} \frac{da_i}{dt} + K_{ij} a_i = M_{ij} Q_i + \int_{\Gamma_q} \omega q d\Gamma_q \quad (3.17)$$

Da mesma forma que foi feito para o termo fonte, a seguinte aproximação pelo método de Galerkin é feita:

$$\begin{aligned} q(x, y) &= \sum_{i=0}^{n-1} N_i(x, y) q_i \\ \omega(x, y) &= \sum_{j=0}^{n-1} N_j(x, y) b_i \end{aligned} \quad (3.18)$$

Com isso, a matriz de massa pode ser utilizada. É importante ressaltar que a matriz de massa neste instante é a matriz de massa do triangulo (bidimensional) pois é aplicada apenas ao contorno, portanto não pode ser a matriz de massa global (do tetraedro). Denomina-se matriz de massa do elemento triangular do contorno M^C . A Equação 3.17 se torna:

$$(\rho c_v) M_{ij} \frac{da_i}{dt} + K_{ij} a_i = M_{ij} Q_i + M_{ij}^C q_i \quad (3.19)$$

Com isso, apenas o vetor b do lado direito da Equação 3.30 sofreu um alteração, onde é somado o fluxo de calor recebido pela geometria.

3.1.4 Aplicação em um problema tridimensional transiente com condição de contorno de Robin

No processo de frenagem a troca de calor por convecção com o ar é importante no estudo de transferência de calor com disco de freio. Para equacionar este fenômeno, neste momento apenas a condição de contorno de Robin será aplicada.

Da mesma forma que na condição de contorno de Neuman, a Equação 3.5 não tem a função peso ω nula. Portanto, após aplicação do método de Galerkin nos termos do domínio obtém-se:

$$\begin{aligned} & \int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (\rho c_v) N_i N_j \frac{da_i}{dt} d\Omega + \int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} k \nabla N_i \cdot \nabla N_j d\Omega a_i = \\ & = \int_{\Omega} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} N_i N_j Q_i d\Omega + \int_{\Gamma_h} \omega k \nabla T d\Gamma_h \end{aligned} \quad (3.20)$$

O contorno Γ_h se refere a superfície do disco que troca calor com o ar. Para condição de contorno do terceiro tipo, o termo $k \nabla T$ da Equação 3.20 é o fluxo térmico dado pela Equação 2.2:

$$k \nabla T = h(T_{\infty} - T) \quad (3.21)$$

Portanto:

$$\int_{\Gamma_h} \omega k \nabla T d\Gamma_h = \int_{\Gamma_h} \omega (h(T_{\infty} - T)) d\Gamma_h = \int_{\Gamma_h} h \omega T_{\infty} d\Gamma_h - \int_{\Gamma_h} h \omega T d\Gamma_h \quad (3.22)$$

O primeiro termo é similar ao caso de Neumann, onde será utilizado a matriz de massa no contorno, porém desta vez, multiplicada por h . Denomina-se:

$$M_{ij}^h = h \cdot M_{ij}^C$$

Como este termo acompanha um valor conhecido do problema, ficará ao lado direito da equação.

Para o segundo termo, também será utilizada a matriz de massa do contorno, porém atrelada a incógnita do problema no lado esquerdo da equação. Assim como na condição do segundo tipo, é importante ratificar que a matriz de massa citada representa a matriz de massa do triângulo (bidimensional) pois é aplicada apenas ao contorno do sólido.

Portanto, a Equação 3.20 fica:

$$(\rho c_v) M_{ij} \frac{da_i}{dt} + K_{ij} a_i + \int_{\Gamma_h} h \omega T d\Gamma_h = M_{ij} Q_i + \int_{\Gamma_h} h \omega T_\infty d\Gamma_h \quad (3.23)$$

$$(\rho c_v) M_{ij} \frac{da_i}{dt} + (K_{ij} + M_{ij}^h) a_i = M_{ij} Q_i + M_{ij}^h T_\infty|_{\Gamma_h} \quad (3.24)$$

3.1.5 Equacionamento do caso completo de frenagem

No presente trabalho, a radiação será negligenciada, portanto, com o desenvolvimento feito até aqui, é possível equacionar a situação completa da frenagem, onde existe a condição de contorno do 2º e do 3º tipo representando as condições externas ao disco de troca de calor, como mostrado na Figura 3.1.

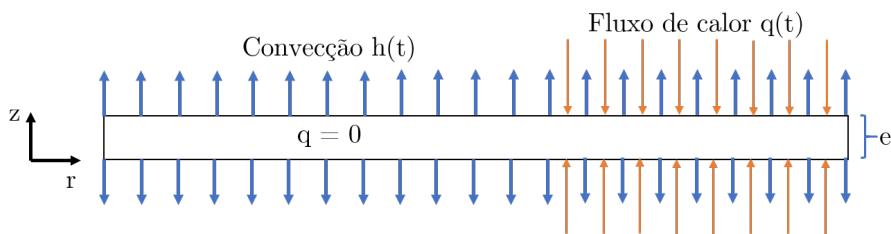


Figura 3.1: Condições de contorno

Fonte: Elaborada pelo autor

Para aplicação de ambas as condições a superfície do disco é dividida em duas partes: a parte que sofre convecção (Γ_h) formado pelas superfícies inferior ($z = 0$) e superior ($z = e$) do disco e a parte que mantém contato com a pastilha (Γ_q).

Conforme as Equações 3.19 e 3.24 pode-se obter o equacionamento geral:

$$(\rho c_v) M_{ij} \frac{da_i}{dt} + K_{ij} a_i + M_{ij}^h a_i = M_{ij} Q_i + M_{ij}^C q_i + M_{ij}^h T_\infty \quad (3.25)$$

Como feito no caso da condição de contorno de Dirichlet, o termo temporal é aproximado pelo MDF progressivo, resultando na equação:

$$(\rho c_v) M_{ij} \left(\frac{a_i^{n+1} - a_i^n}{\Delta t} \right) + K_{ij} a_i + M_{ij}^h a_i = M_{ij} Q_i + M_{ij}^C q_i + M_{ij}^h T_\infty \quad (3.26)$$

Para este passo, pode-se utilizar três métodos distintos:

- Implícito - define o termo $K_{ij} a_i$ e $M_{ij}^h a_i$ no tempo $(n+1)$:

$$\left(\frac{(\rho c_v) M_{ij}}{\Delta t} + K_{ij} + M_{ij}^h \right) a_i^{n+1} = \frac{(\rho c_v) M_{ij}}{\Delta t} a_i^n + M_{ij} Q_i + M_{ij}^C q_i|_\Gamma + M_{ij}^h T_\infty$$

- Explícito - define o termo $K_{ij} a_i$ e $M_{ij}^h a_i$ no tempo (n) :

$$\frac{(\rho c_v) M_{ij}}{\Delta t} a_i^{n+1} = \left(\frac{(\rho c_v) M_{ij}}{\Delta t} - K_{ij} - M_{ij}^h \right) a_i^n + M_{ij} Q_i + M_{ij}^C q_i + M_{ij}^h T_\infty$$

- Crank-Nicolson - pondera os dois métodos:

$$\begin{aligned} \left(\frac{(\rho c_v) M_{ij}}{\Delta t} + \frac{1}{2} (K_{ij} + M_{ij}^h) \right) a_i^{n+1} &= \left(\frac{(\rho c_v) M_{ij}}{\Delta t} - \frac{1}{2} (K_{ij} + M_{ij}^h) \right) a_i^n \\ &\quad + M_{ij} Q_i + M_{ij}^C q_i + M_{ij}^h T_\infty \end{aligned} \quad (3.27)$$

Pode-se então, escrever em termos de θ :

$$\begin{aligned} \left(\frac{(\rho c_v) M_{ij}}{\Delta t} + \theta (K_{ij} + M_{ij}^h) \right) a_i^{n+1} &= \left(\frac{(\rho c_v) M_{ij}}{\Delta t} - (1 - \theta) (K_{ij} + M_{ij}^h) \right) a_i^n \\ &\quad + M_{ij} Q_i + M_{ij}^C q_i + M_{ij}^h T_\infty \end{aligned} \quad (3.28)$$

Sendo:

- $\theta = 1$ para o método implícito;
- $\theta = 0$ para o método explícito;
- $\theta = 0.5$ para Crank-Nicolson.

No código, a equação 3.28 é escrita da forma:

$$\begin{aligned} \rho c_v M_{ij} + \theta \Delta t (K_{ij} + M_{ij}^h) a_i^{n+1} &= [\rho c_v M_{ij} - \Delta t (1 - \theta) (K_{ij} + M_{ij}^h)] a_i^n \\ &\quad + M_{ij} \Delta t Q_i + M_{ij}^C \Delta t q_i + M_{ij}^h \Delta t T_\infty \end{aligned} \quad (3.29)$$

onde:

$$A = \rho c_v M_{ij} + \theta \Delta t (K_{ij} + M_{ij}^h)$$

$$b = [\rho c_v M_{ij} - \Delta t (1 - \theta) (K_{ij} + M_{ij}^h)] a_i^n + M_{ij} \Delta t Q_i + M_{ij}^C \Delta t q_i + M_{ij}^h \Delta t T_\infty$$

$$a = T$$

Portanto o sistema resolvido é do tipo:

$$A_{ij} \cdot T_i^{n+1} = b_i \quad (3.30)$$

3.2 Modelo de frenagem

Neste trabalho, o sistema será modelado conforme as dimensões da Figura 3.2, portanto consideram-se as grandezas que definem a geometria: os raios externo r_e e interno r_i do disco e o raio interno da pastilha r_t e a espessura e do disco.

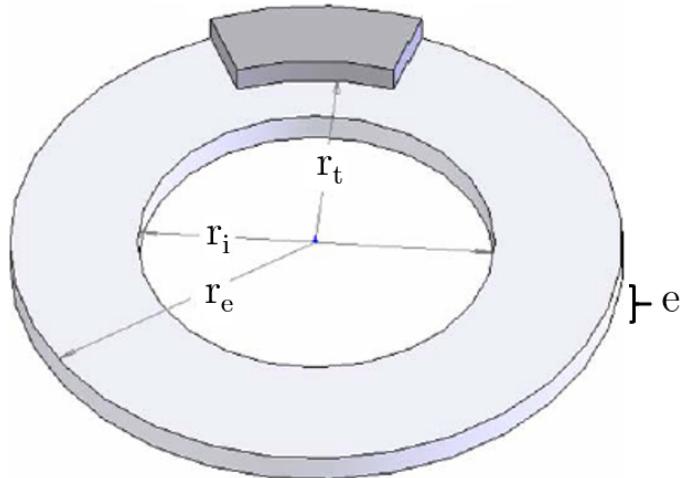


Figura 3.2: Esquema geométrico do contato entre disco e pastilha

Fonte: Elaborada pelo autor

As hipóteses adotadas para simular o momento da frenagem são:

- A pressão que a pastilha faz no disco será considerada constante entre r_t e r_e , onde assume-se um raio médio r_m ;
- A desaceleração do veículo é considerada constante;

- Os materiais são considerados isotrópicos, ou seja, as propriedades dos materiais são constantes em todas as direções. Além disso as propriedades também são independentes da variação de temperatura;
- O fluxo de calor varia com a velocidade de rotação do disco;
- O valor da velocidade do ar utilizada para o cálculo do coeficiente de convecção considera a mesma velocidade do veículo;
- A dissipação de calor por radiação não é incluída por ser considerada pouco influente, já que o período de tempo de frenagem é curto [8] [9].

3.2.1 Fluxo de calor

A taxa de calor \dot{E} produzida em um conjunto de freios durante a força de atrito da frenagem foi definida na Equação 2.49. Portanto, o fluxo de calor nos discos de freio é dado por:

$$\dot{E} = VFat = (\omega r_m) \cdot 2\mu N \quad (3.31)$$

onde r_m é o raio médio da pastilha (considerando pressão de contato uniforme), N é a força normal exercida pelo pistão da pinça calculada conforme Equação 2.17 e μ é o coeficiente de atrito entre disco e pastilha. Com isso, o calor fornecido ao disco dianteiro é dado por:

$$\dot{E}_{dF} = \gamma \dot{E} = \gamma (\omega r_m) \cdot 2\mu P_F A_F \quad (3.32)$$

onde P_F e A_F são a pressão nas linhas de freio dianteiras e a área do pistão da pinça de freios dianteira, respectivamente. O fluxo de calor na área de contato entre pastilha e disco dianteiro A_d é, então, calculado por:

$$q(t) = \frac{\dot{E}_{dF}}{A_d} = \gamma (\omega r_m) \cdot 2\mu P_F \left(\frac{A_F}{A_d} \right) [W/m^2] \quad (3.33)$$

3.2.2 Convecção

Como proposto por Jianyong [10], o coeficiente de convecção de um disco de freios pode ser calculado considerando-o como um tubo que sofre convecção forçada na sua seção transversal. Com isso o coeficiente de convecção pode ser calculado de acordo com as Equações 3.34, onde Re é o número de Reynolds, Pr é o número de Prandt, k_{ar} é a condutividade térmica do ar e D o diâmetro do disco de freios.

$$\begin{aligned}
h_{tubo} &= 0.989 \cdot Re^{0.33} \cdot Pr^{0.33} \cdot \frac{k_{ar}}{D}, \quad 0.4 < Re < 4 \\
h_{tubo} &= 0.911 \cdot Re^{0.385} \cdot Pr^{0.33} \cdot \frac{k_{ar}}{D}, \quad 4 < Re < 40 \\
h_{tubo} &= 0.683 \cdot Re^{0.466} \cdot Pr^{0.33} \cdot \frac{k_{ar}}{D}, \quad 40 < Re < 4000 \\
h_{tubo} &= 0.193 \cdot Re^{0.618} \cdot Pr^{0.33} \cdot \frac{k_{ar}}{D}, \quad 4000 < Re < 40000 \\
h_{tubo} &= 0.027 \cdot Re^{0.805} \cdot Pr^{0.33} \cdot \frac{k_{ar}}{D}, \quad 40000 < Re < 400000
\end{aligned} \tag{3.34}$$

3.3 O Algoritmo computacional

O código foi construído de uma forma que, mesmo sendo um algoritmo tridimensional com condições de contorno complexas, seja intuitivo e bem particionado, com o objetivo de fácil leitura e entendimento. Para isso módulos do python foram utilizados de forma a separar as funcionalidades do código principal, como mostra o diagrama da Figura 3.3.

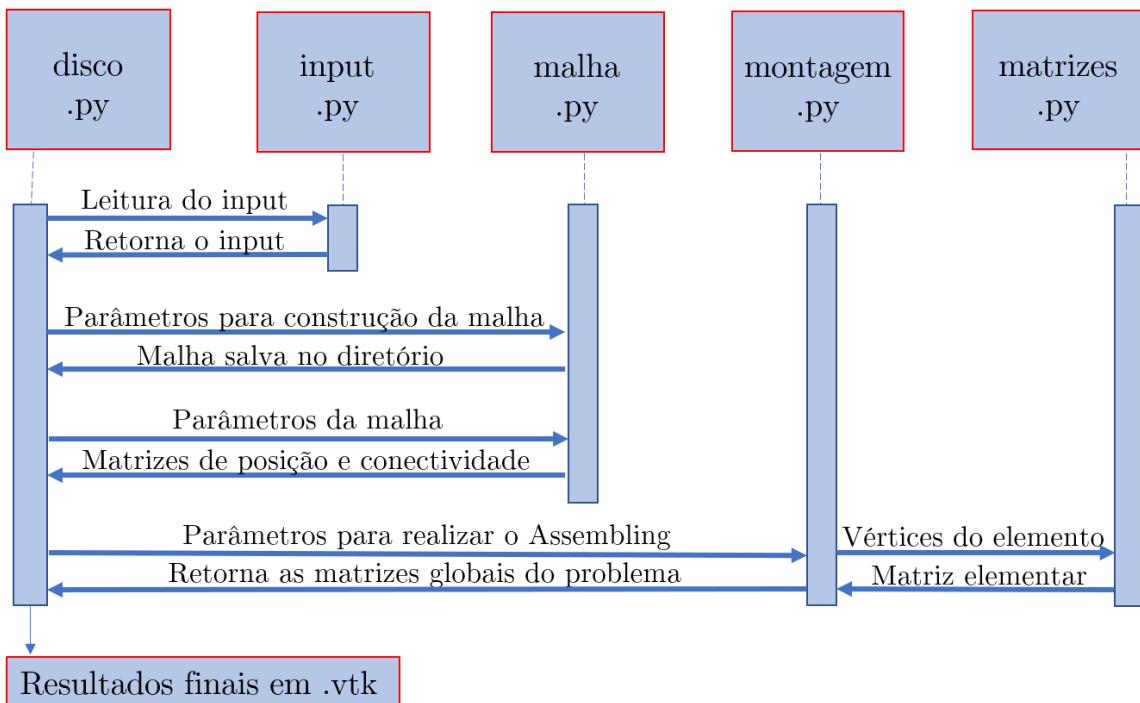


Figura 3.3: Diagrama de funcionamento do código

Fonte: Elaborada pelo autor.

Neste seção, serão trabalhadas as funcionalidades principais do código proposto. Os códigos completos podem ser encontrados no Apêndice A.

O código principal *disco.py* é um *solver* que reúne todas as informações do problema e pode ser encontrado no Apêndice A.1.

Para a leitura dos parâmetros do disco e das condições físicas da simulação, o código faz a leitura de uma planilha de input do usuário - Figura 3.4 - através do módulo *input.py*. Este módulo pode ser encontrado no Apêndice A.2.

Planilha de input												Desenvolvido por: Felipe Alves	Checado por: Gustavo R.	Data: 02/05/2021
Projeto		Temporada		Revisão	Tipo de frenagem		Nº de ciclos		Material utilizado			Observação		
Disco de freios		20-21		0	Multipla		40		Aço SAE 1045					
#	Parâmetros da simulação					Geometria				Propriedades do material			Parâmetros do	
-	Time step	Iterações	θ (MDF)	Tamanho médio do elemento	Espessura	Raio externo	Raio interno	Raio interno pastilha	Raio dos furos	Condut. térmica	Calor específico	Massa específica	Temp. inicial	Temp. do ar
1	[s]	[-]	[-]	[mm]	[mm]	[mm]	[mm]	[mm]	[mm]	[W/m.K]	[J/kg.K]	[kg/m^3]	[°C]	[°C]
1	0.01	960	0.5	1.33	4	109.0	68.2	80.8	2.0	49.8	486.0	7850.0	35	35

Figura 3.4: Planilha de input do usuário

Fonte: Elaborada pelo autor.

Para a construção da malha, o código chama o módulo *malha.py*, que também é utilizado para retornar vetores de interesse para a solução do problema, como as matrizes IEN principal e de contorno.

Após toda a construção geométrica, o módulo *montagem.py* é utilizado para realizar o *Assembling* e retornar as matrizes globais do problema. Esse módulo chama outro módulo - *matrizes.py* - reservado para conter a construção das matrizes elementares.

3.3.1 Gerador de malhas - GMSH

O módulo *malha.py* faz utilização do API (Application Programming Interface) do gerador de malhas GMSH. O API permite que, apenas com a linguagem em Python, seja construída a malha tridimensional desejada.

O GMSH trabalha com dois módulos distintos que tem relação entre si: o da geometria e o da malha. Segundo o tutorial online do GMSH [11], Os dados geométricos são feitos de entidades, chamadas de pontos (entidade de dimensão 0), curvas (entidade de dimensão 1), superfícies (entidade de dimensão 2) ou volumes (entidade de dimensão 3). As entidades são registradas junto com seus contornos, onde os volumes são contornados por um conjunto de superfícies, uma superfície é contor-

nada por um conjunto de curvas e uma curva é contornada por dois pontos. Essas entidades são identificadas por suas *tags*. Os chamados *Physical groups* são grupos de entidades e também são registrados com uma *tag*.

Para montar a malha de um disco de freios, a função *disc* foi criada. Para iniciar o API, o código deve importar a biblioteca e dar o comando de inicialização:

```

1 import gmsh
2 def disc(re, rt, ri, e, le_min, le_max, filename, furos):
3     # Before using any functions in the API, Gmsh must be initialized:
4     gmsh.initialize()
5     gmsh.model.add("minha_malha")

```

onde r_e , r_t , r_i e e são os parâmetros físicos do disco de freio mostrados na Figura 3.2 e le_{min} e le_{max} são os tamanhos mínimo e máximo dos elementos da malha.

Para modelar um disco de freios, um cilindro pode ser adicionado utilizando a função abaixo:

```

# addCylinder(x, y, z, dx, dy, dz, raio, tag)
2 cylinder = gmsh.model.occ.addCylinder(0,0,0,0,0,e,re,tag=1)

```

Para modelar o furo do centro do disco, basta adicionar outro cilindro. Este cilindro será subtraído do cilindro principal através da função *cut*.

```
cylinder = gmsh.model.occ.addCylinder(0,0,0,0,0,e,ri,tag=2)
```

Para fazer os furos do disco, a mesma ideia é usada em um *loop*.

```

if furos != 'no':
    # furos eh um lista com [raio do furinho, angulo entre furos]
2     r    = furos[0]
4     ang = furos[1]
     d = re-(13.6/1000)
6     p = ri+(13.6/1000)
     coordIn = [d,p]

```

```

8
9     coord = []
10    for i in range(1,9):
11        coord.append([d*np.cos(i*ang), d*np.sin(i*ang)])
12    for i in range(1,9):
13        coord.append([p*np.cos(i*ang), p*np.sin(i*ang)])
14    t = 4
15    for c in coordIn:
16        gmsh.model.occ.addCylinder(c,0,0,0,0,e,r,tag=t)
17        gmsh.model.occ.addCylinder(-c,0,0,0,0,e,r,tag=t+1)
18        gmsh.model.occ.addCylinder(0,c,0,0,0,e,r,tag=t+2)
19        gmsh.model.occ.addCylinder(0,-c,0,0,0,e,r,tag=t+3)
20        t = t+4
21    for c in coord:
22        gmsh.model.occ.addCylinder(c[0],c[1],0,0,0,e,r,tag=t)
23        gmsh.model.occ.addCylinder(-c[0],c[1],0,0,0,e,r,tag=t+1)
24        gmsh.model.occ.addCylinder(-c[0],-c[1],0,0,0,e,r,tag=t+2)
25        gmsh.model.occ.addCylinder(c[0],-c[1],0,0,0,e,r,tag=t+3)
26        t = t+4

27
28    # Funcao cut para subtrair os furos
29    # furinhos
30    gmsh.model.occ.cut([(3, 1)], [(3, i) for i in range(4,t)], 3)
31    # miolo do disco
32    domain = gmsh.model.occ.cut([(3, 3)], [(3, 2)], 4)

```

Agora é necessário transformar esa geometria em malha. Para isso é necessário sincronizar o modelo, salvar suas entidades e definir o tamanho dos elementos da malha.

```

# sincronizar o modelo
2 gmsh.model.occ.synchronize()
3 volumes = gmsh.model.getEntities(dim=3)
4
5 # Definir os tamanhos dos elementos da malha
6 gmsh.option.setNumber("Mesh.CharacteristicLengthMin", le_min)
7 gmsh.option.setNumber("Mesh.CharacteristicLengthMax", le_max)

```

Dados de uma malha são formados por elementos (pontos, linhas, triângulos, tetraedros, etc.) definidos por uma listagem ordenada de seus nós. Elementos e nós são identificados por *tags*. Uma vez que a geometria foi transformada em malha, uma entidade de dimensão 0 irá conter um elemento do tipo nó na malha. Uma curva da geometria vai conter elementos de linha da malha e os nós de seu interior, enquanto que os nós do seu contorno são armazenados como pontos de contorno do modelo. Uma superfície da geometria irá conter elementos triangulares e / ou quadrangulares assim como um volume da geometria irá conter tetraedros ou hexaedros.

Para gerar a malha do modelo, salvar o arquivo e finalizar o API, basta escrever:

```

# Gerar malha
2 gmsh.model.mesh.generate(3) # 3D
# Salvar arquivo .msh
4 gmsh.write(filename)
gmsh.finalize()

```

Como demonstrado anteriormente, para realizar o *Assembling* é necessário ter as matrizes de conectividade principal, e de contorno. O módulo *malha.py* ainda faz a leitura do arquivo salvo e retorna estas matrizes para o código principal através da função *boundf*:

```

import meshio
2 def boundf(filename):
    msh = meshio.read(filename)
    IENbound = [] # IEN de nohs do contorno
    4 for elem in msh.cells:
        if elem[0] == 'triangle': # Elementos triangulares
            IENbound.append(elem[1])
        6 elif elem[0] == 'tetra': # Elementos tetraedricos
            IEN = elem[1] # Matriz de conectivide IEN
8
10 return IENbound, IEN

```

Para os casos em que o programa utilizou paralelepípedos, outra função do API foi utilizada e pode ser encontrada no Apêndice A.3.2. Além disso o usuário pode

construir uma malha através da interface gráfica do GMSH e usar o código da mesma forma após desabilitar o comando de construção da malha (chamada da função *disc*).

3.3.2 Módulo de montagem

Para realizar o *Assembling* das matrizes globais tridimensionais do problema o módulo *montagem.py* tem a função que é apresentada a seguir.

```

1  from tqdm import tqdm
2  from modulos.matrizes import matriz3D, matriz2D
3  from scipy.sparse import lil_matrix, csr_matrix, issparse
4
5  def assembling3D(IEN, npoints, ne, X, Y, Z, k):
6      # LIL is a convenient format for constructing sparse matrices
7      K = lil_matrix((npoints, npoints), dtype='double')
8      M = lil_matrix((npoints, npoints), dtype='double')
9      for e in tqdm(range(ne)):
10          # construir as matrizes do elemento
11          v1 = IEN[e, 0]           # vertice 1
12          v2 = IEN[e, 1]           # vertice 2
13          v3 = IEN[e, 2]           # vertice 3
14          v4 = IEN[e, 3]           # vertice 4
15
16          # importando matrizes do modulo matrizes
17          m = matriz3D(v1=v1, v2=v2, v3=v3, v4=v4, X=X, Y=Y, Z=Z)
18          melem = m.matrizm()
19          kelem = m.matrizk(k)
20          for ilocal in range(0, 4):
21              igoal = IEN[e, ilocal]
22              for jlocal in range(0, 4):
23                  jgoal = IEN[e, jlocal]
24                  K[igoal, jgoal] = K[igoal, jgoal] + kelem[ilocal, jlocal]
25                  M[igoal, jgoal] = M[igoal, jgoal] + melem[ilocal, jlocal]
26
27      return K, M

```

As matrizes bidimensionais, por sua vez, são montadas com o código a seguir:

```

1  def assembling2D (IENboundG , npoints ,X,Y,Z,c) :
2      # Matriz de massa do contorno (de Neumann e Robin)
3      MC = lil_matrix (( npoints , npoints ) , dtype='double')
4      for e in tqdm(IENboundG) :
5          # construir as matrizes do elemento
6          v1 = e [0]
7          v2 = e [1]
8          v3 = e [2]
9          v = [v1 ,v2 ,v3 ]
10
11
12          # importando matrizes do modulo matrizes
13          # obs: o codigo pode ser usado pois os elementos estao sempre na
14          # mesma
15
16          # coordenada Z. Caso nao estivesse , teria que fazer diferente.
17          sq = matriz2D (v1=v1 , v2=v2 , v3=v3 , X=X, Y=Y)
18          melemg = sq .matrizm ()
19
20
21          for ilocal in range (0 , 3) :
22              igoal = v [ ilocal ]
23              for jlocal in range (0 , 3) :
24                  jgoal = v [ jlocal ]
25                  MC [ igoal , jgoal ]=MC [ igoal , jgoal ]+c*melemg [ ilocal , jlocal ]
26
27
28      return MC

```

3.3.3 Módulo de matrizes elementares

Como mostrado no código de montagem, as funções chamam as funções *matriz3D* e *matriz2D*. Essas funções estão no módulo *matrizes.py* que é responsável por construir as matrizes elementares. A função com as matrizes elementares tridimensionais apresentadas na Equação 2.61 é:

```
2 class matriz3D():
3     def __init__(self, X, Y, Z, v1, v2, v3, v4):
4         self.b1=(Y[v2]-Y[v4])*(Z[v3]-Z[v4])-(Y[v3]-Y[v4])*(Z[v2]-Z[v4])
```

```

4      self.b2=(Y[v3]-Y[v4])*(Z[v1]-Z[v4])-(Y[v1]-Y[v4])*(Z[v3]-Z[v4])
5      self.b3=(Y[v1]-Y[v4])*(Z[v2]-Z[v4])-(Y[v2]-Y[v4])*(Z[v1]-Z[v4])
6      self.b4 = -(self.b1 + self.b2 + self.b3)

8      self.c1=(X[v3]-X[v4])*(Z[v2]-Z[v4])-(X[v2]-X[v4])*(Z[v3]-Z[v4])
9      self.c2=(X[v1]-X[v4])*(Z[v3]-Z[v4])-(X[v3]-X[v4])*(Z[v1]-Z[v4])
10     self.c3=(X[v2]-X[v4])*(Z[v1]-Z[v4])-(X[v1]-X[v4])*(Z[v2]-Z[v4])
11     self.c4 = -(self.c1 + self.c2 + self.c3)

12     self.d1=(X[v2]-X[v4])*(Y[v3]-Y[v4])-(X[v3]-X[v4])*(Y[v2]-Y[v4])
13     self.d2=(X[v3]-X[v4])*(Y[v1]-Y[v4])-(X[v1]-X[v4])*(Y[v3]-Y[v4])
14     self.d3=(X[v1]-X[v4])*(Y[v2]-Y[v4])-(X[v2]-X[v4])*(Y[v1]-Y[v4])
15     self.d4 = -(self.d1 + self.d2 + self.d3)

18     self.vol=(1/6)*np.linalg.det(np.array([[1,X[v1],Y[v1],Z[v1]],
19                                         [1,X[v2],Y[v2],Z[v2]],
20                                         [1,X[v3],Y[v3],Z[v3]],
21                                         [1,X[v4],Y[v4],Z[v4]]]))
22
23     def volCalc(self):
24         return self.vol

26     def matrizm(self):
27         melem = (self.vol/20)*np.array([[2.0, 1.0, 1.0, 1.0],
28                                         [1.0, 2.0, 1.0, 1.0],
29                                         [1.0, 1.0, 2.0, 1.0],
30                                         [1.0, 1.0, 1.0, 2.0]])
31
32     def matrizk(self, k):
33         # condutividade termica k
34         # Para material isotropico (kx=ky=kz=k)
35         B = (1/(6* self.vol))*np.array([[self.b1, self.b2, self.b3, self.b4],
36                                         [self.c1, self.c2, self.c3, self.c4],
37                                         [self.d1, self.d2, self.d3, self.d4]])
38
39         BT = np.transpose(B)
40         kelem = self.vol * k * np.dot(BT, B)
41
42         return kelem

```

A função com as matrizes elementares bidimensionais apresentadas a Equação 2.66 é:

```

1  class matriz2D():
2      def __init__(self, X, Y, v1, v2, v3):
3          self.b1 = Y[v2] - Y[v3]
4          self.b2 = Y[v3] - Y[v1]
5          self.b3 = Y[v1] - Y[v2]
6          self.c1 = X[v3] - X[v2]
7          self.c2 = X[v1] - X[v3]
8          self.c3 = X[v2] - X[v1]
9          self.area=(1/2)*np.linalg.det(np.array([[1.0,X[v1],Y[v1]],
10                                         [1.0, X[v2], Y[v2]],
11                                         [1.0, X[v3], Y[v3]]]))
12
13      def areaCalc(self):
14          return self.area
15
16      def matrizm(self):
17          melem = (self.area/12.0)*np.array([[2.0, 1.0, 1.0],
18                                         [1.0, 2.0, 1.0],
19                                         [1.0, 1.0, 2.0]])
20
21      def matrizk(self):
22          B = (1/(2*self.area))*np.array([[self.b1, self.b2, self.b3],
23                                         [self.c1, self.c2, self.c3]])
24
25          BT = np.transpose(B)
26          kelem = self.area * np.dot(BT, B)
27
28      return kelem

```

Com isso, o código principal *disco.py* reúne todas as informações para realizar as iterações temporais da equação transiente.

Os resultados são salvos no formato *.vtk* e podem ser lidos no software aberto de visualização denominado *Paraview*.

Capítulo 4

Validação

Para validar os resultados encontrados no código Python proposto, foram feitos três estudos: uma comparação com uma solução analítica e duas comparações com simulações feitas no software comercial consolidado *Ansys*.

4.1 Validação através de solução analítica

A validação através de solução analítica foi feita no regime permanente, onde o código transiente proposto foi utilizado com tempo de simulação suficientemente grande.

4.1.1 Solução analítica proposta

A solução analítica proposta foi realizada em um quadrado de lados unitários onde três arestas têm uma temperatura T_1 e a aresta superior tem outra temperatura T_2 , como mostrado na Figura 4.1.

A solução é comparada com o resultado do código tridimensional, onde apenas o plano $Z = 0$ (plano lateral conforme eixo mostrado na Figura 4.3) é levado em consideração e todas as propriedades do material são unitárias - densidade, condutividade térmica e calor específico.

A equação que resulta na solução analítica é dada por:

$$\frac{T(x, y) - T_1}{T_2 - T_1} = \frac{2}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^{n+1} + 1}{n} \cdot \sin\left(\frac{n\pi x}{L_x}\right) \cdot \frac{\sinh\left(\frac{n\pi y}{L_x}\right)}{\sinh\left(\frac{n\pi L_y}{L_x}\right)}$$

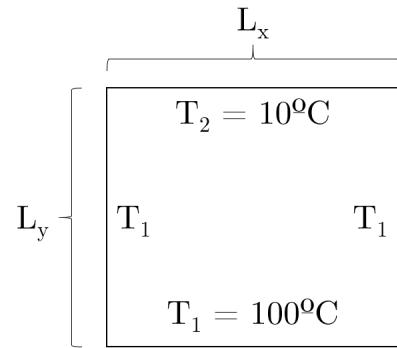


Figura 4.1: Condições de contorno para solução analítica

Fonte: Elaborada pelo autor.

onde o somatório é truncado em um número razoável de iterações.

Após fazer o cálculo de $T(x, y)$ truncando o somatório em $n = 200$ o resultado foi plotado, como mostra a Figura 4.2

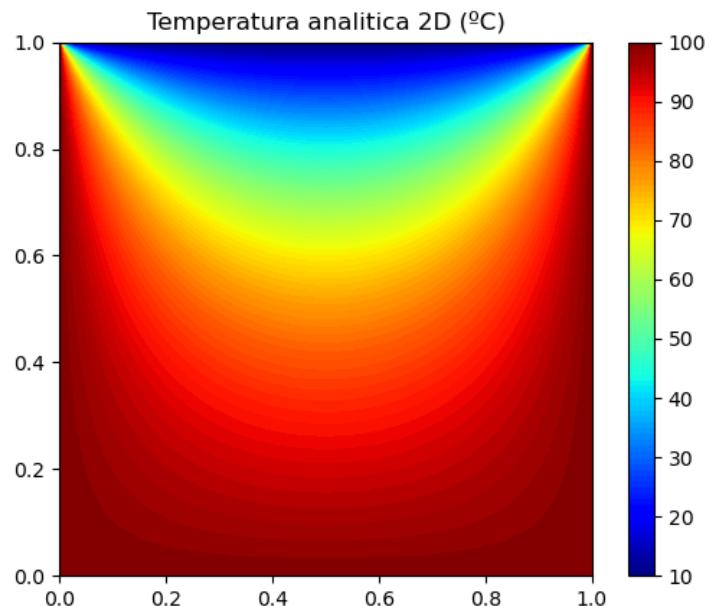


Figura 4.2: Condições de contorno para solução analítica

Fonte: Elaborada pelo autor.

4.1.2 Comparação de resultados

O resultado do código proposto neste trabalho é visualizado a partir do Paraview, como mostra a Figura 4.3

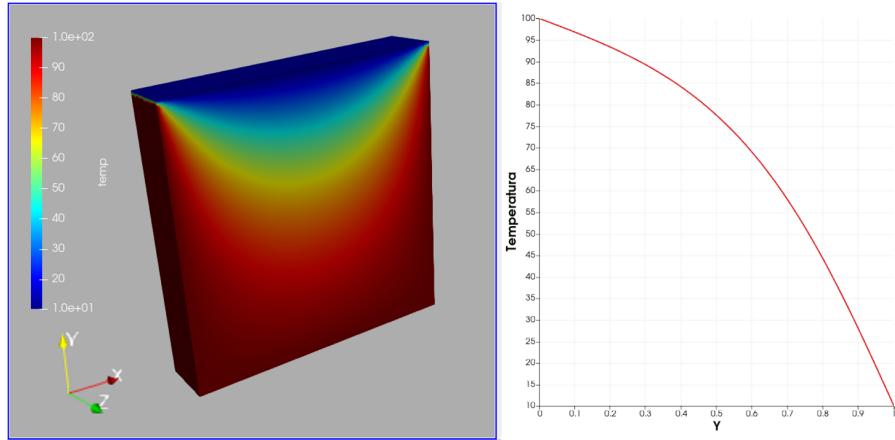


Figura 4.3: Visualização da simulação numérica

Fonte: Elaborada pelo autor.

A título de comparação, um gráfico das temperaturas em função do eixo y é traçado com os resultados analíticos e numéricos em diferentes valores de coordenadas x.

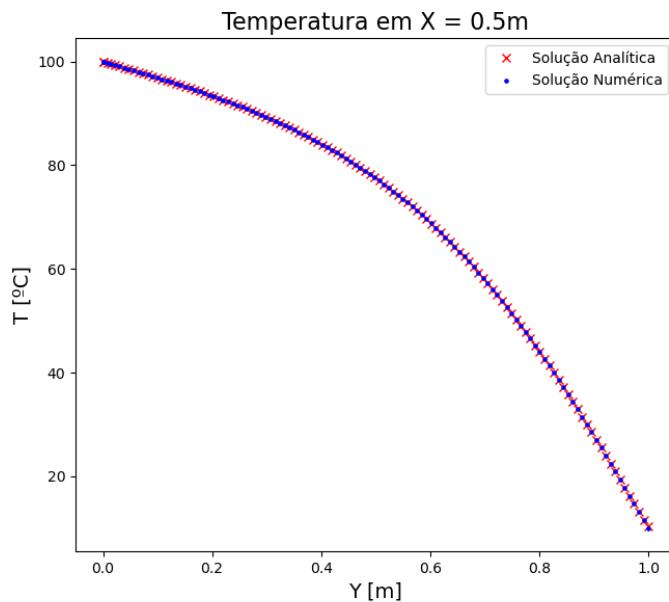


Figura 4.4: Comparação de resultados no eixo central do plano estudado

Fonte: Elaborada pelo autor.

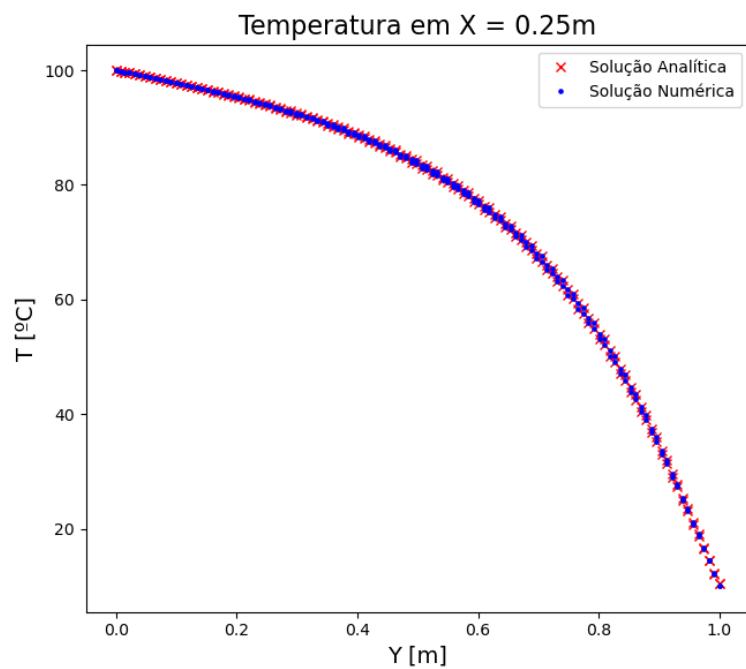


Figura 4.5: Comparação de resultados em $x = 0,25$

Fonte: Elaborada pelo autor.

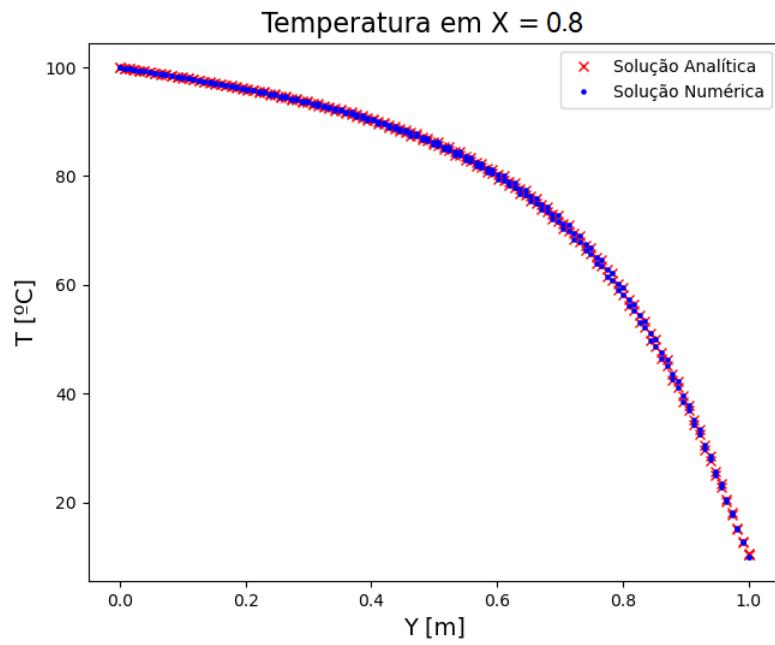


Figura 4.6: Comparação de resultados em $x = 0,8$

Fonte: Elaborada pelo autor.

Foi feita uma comparação ponto a ponto onde foi possível calcular os erros entre o resultado numérico e analítico para os nós internos do plano estudado da seguinte maneira:

$$erro (\%) = 100 \cdot \frac{Temperatura\ analitica - Temperatura\ numerica}{Temperatura\ analitica}$$

Os dados são mostrados na Tabela 4.1.

Tabela 4.1: Erros relativos ponto a ponto

Maior erro	Média dos erros	Desvio padrão
1.98%	0.01%	0.08%

4.2 Comparação com software comercial

Para estudar a confiabilidade dos resultados do código proposto, foram feitas duas simulações térmicas que foram comparadas com o Software comercial consiliado *Ansys*: a primeira avaliando uma geometria simples submetida a temperaturas fixas em suas extremidades (condição de contorno de 1º tipo) e a segunda simulando a geometria de um disco de freios com as condições de 2º e 3º tipos.

4.2.1 Condição de contorno de Dirichlet

A geometria proposta foi um paralelepípedo de arestas [25mm, 25mm, 100mm], como mostra a Figura 4.7, com as seguintes propriedades térmicas (Propriedades padrão do Structural Steel do Ansys):

Tabela 4.2: Propriedades térmicas utilizadas nas simulações

Densidade	Calor específico	Condutividade térmica
kg/m^3	$J/kg.K$	$W/m.K$
7850	434	60.5

A condição inicial foi a seguinte:

- Plano $z = 0$ mm com temperatura fixa de 10ºC;
- Plano $z = 100$ mm com temperatura fixa de 100ºC;

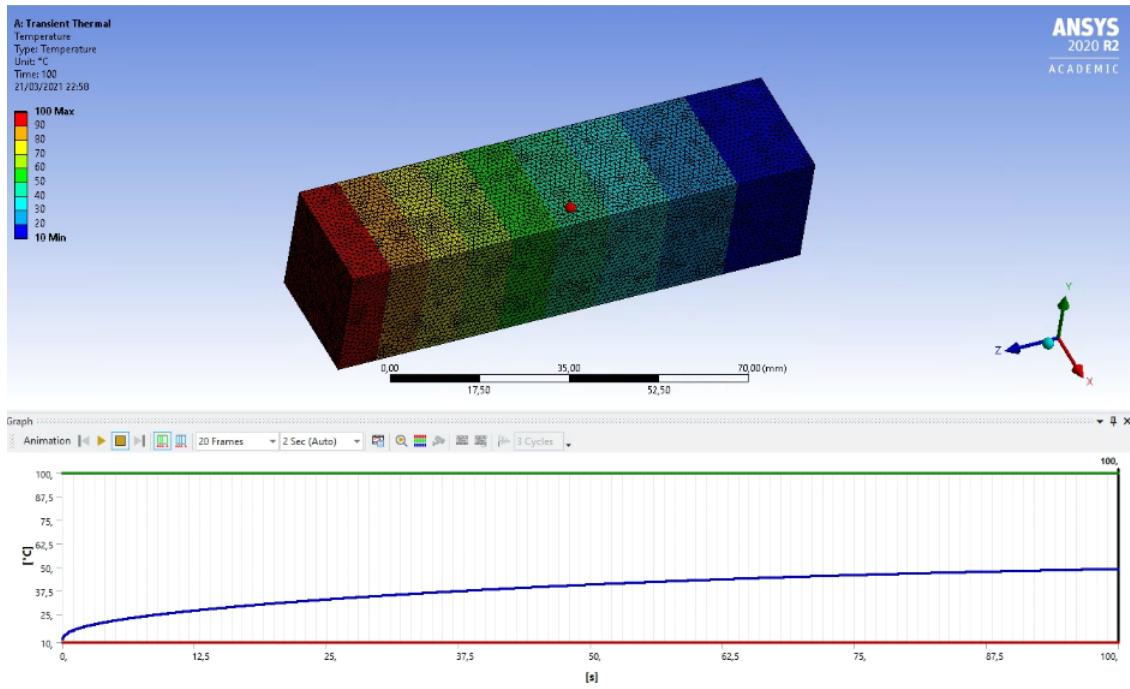


Figura 4.7: Malha e resultado do Ansys. A foto representa o instante de 100s de duração. O gráfico mostra as temperaturas máxima, mínima, e média dos elementos

Fonte: Elaborada pelo autor.

- Todos os demais pontos com temperatura inicial de 10°C, variando ao longo do tempo devido à condução do calor causada pela parede quente.

O *time step* usado foi de 0,1 segundos e a discretização temporal foi realizada considerando o método implícito de diferenças finitas progressivo. Em ambas as simulações, todas as coordenadas (x,y,z) dos nós foram salvas junto com sua respectiva temperatura. Como se trata de uma análise transiente, os resultados foram comparados em alguns instantes de tempo específicos (10s, 20s, 30s, 40s, 50s, 100s). Todos os pontos foram plotados, de forma que a coordenada Z de cada ponto ocupa o eixo das abscissas e a temperatura dos respectivos pontos foi plotado nas ordenadas, como mostrado da Figura 4.8 até a Figura 4.13. O erro absoluto dos valores de temperatura foram plotados no eixo das ordenadas a direita do gráfico. Apesar de ser um gráfico que relaciona duas grandezas, todos os pontos da malha tridimensional foram incluídos.

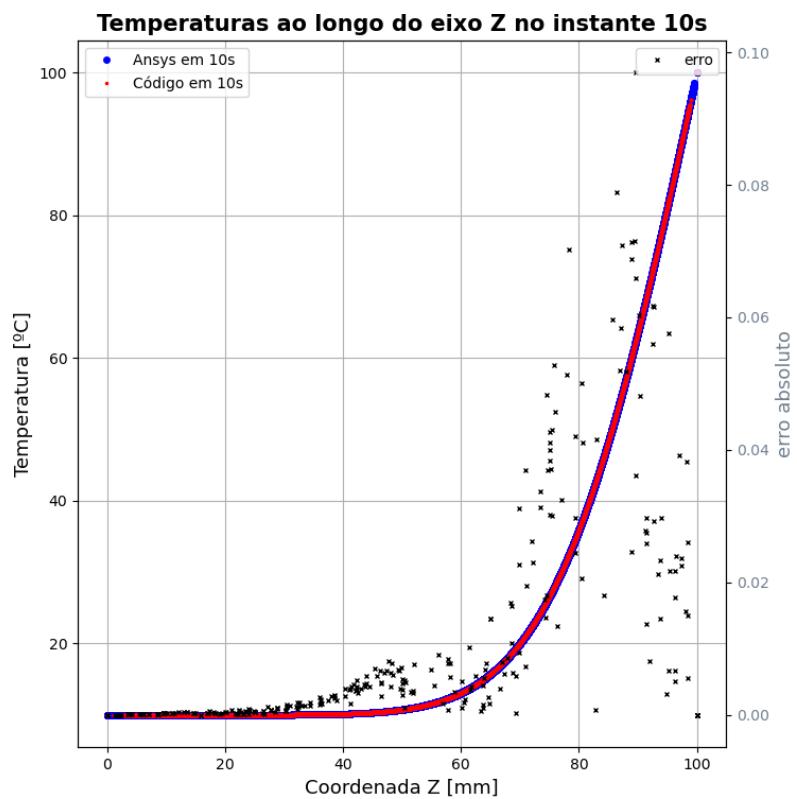


Figura 4.8: Comparação de resultados no instante de 10 segundos

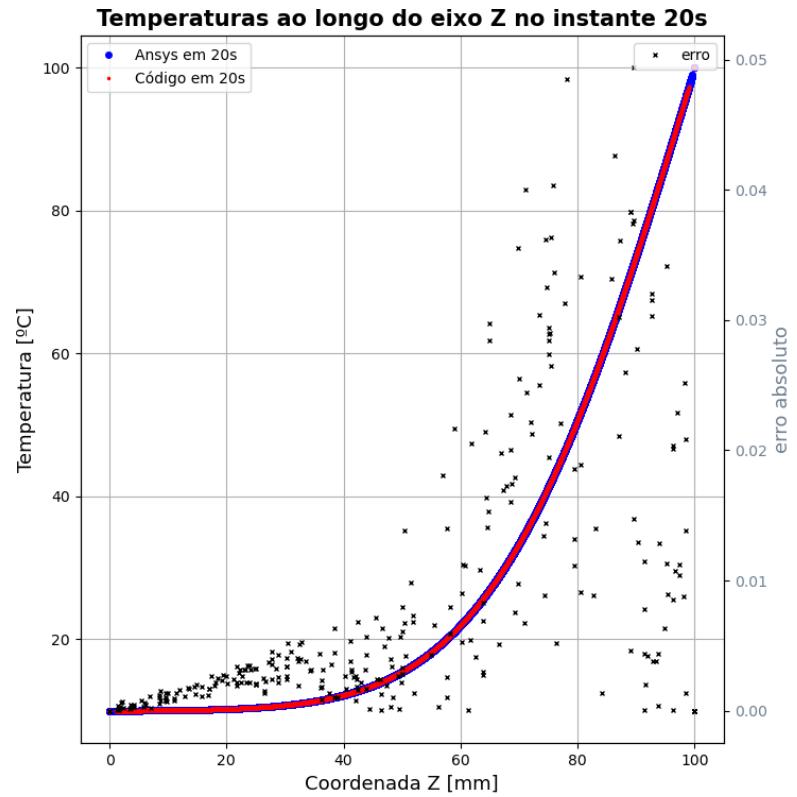


Figura 4.9: Comparação de resultados no instante de 20 segundos

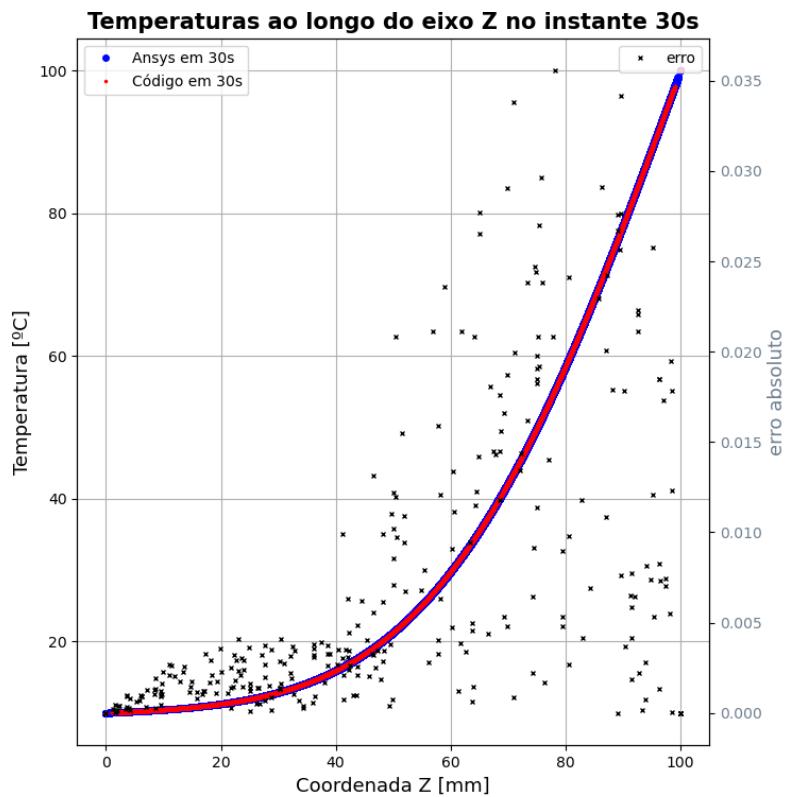


Figura 4.10: Comparação de resultados no instante de 30 segundos

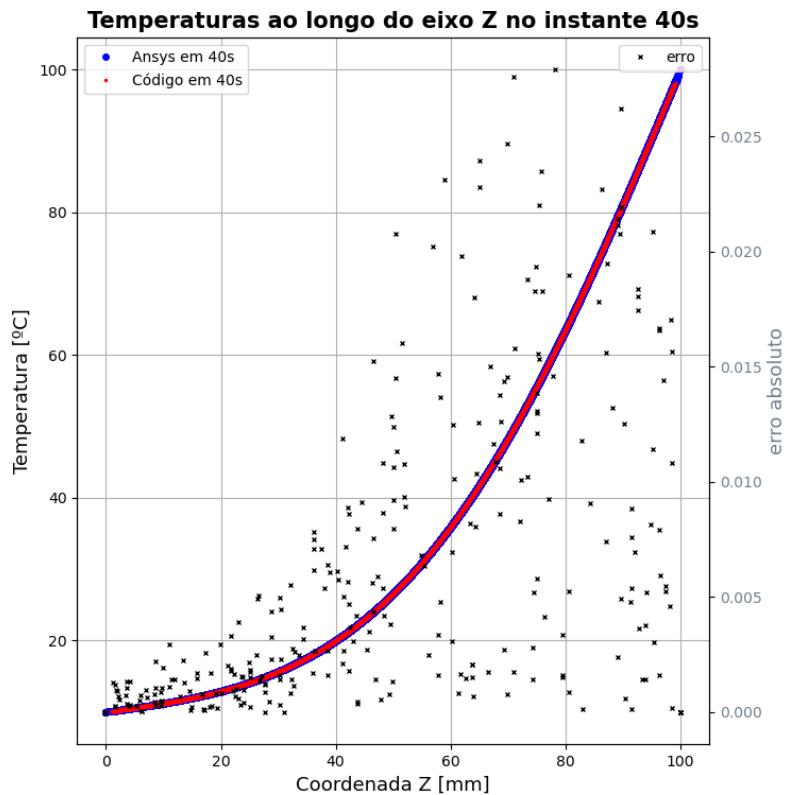


Figura 4.11: Comparação de resultados no instante de 40 segundos

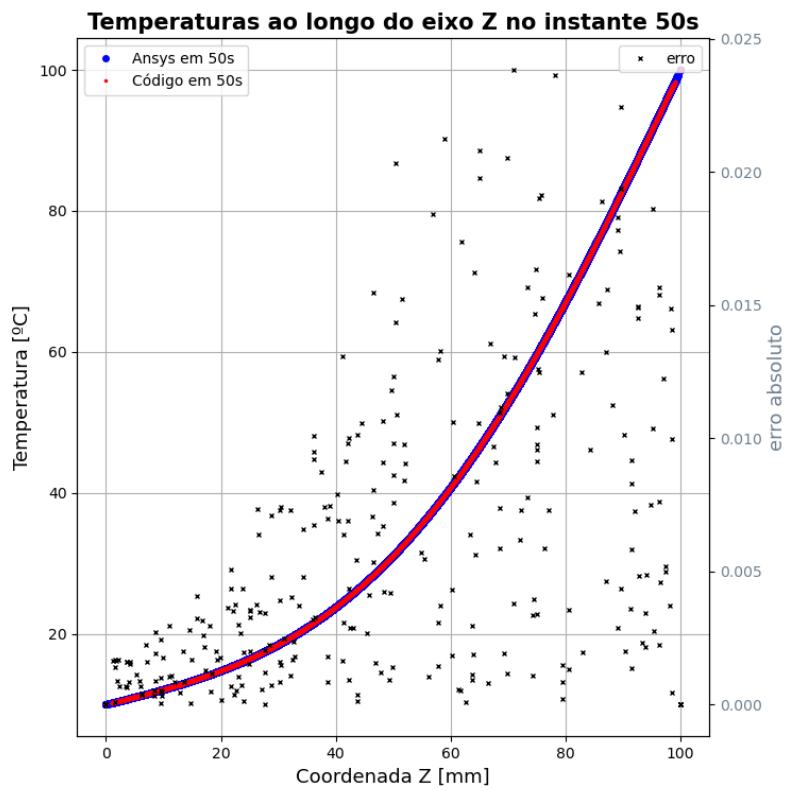


Figura 4.12: Comparação de resultados no instante de 50 segundos

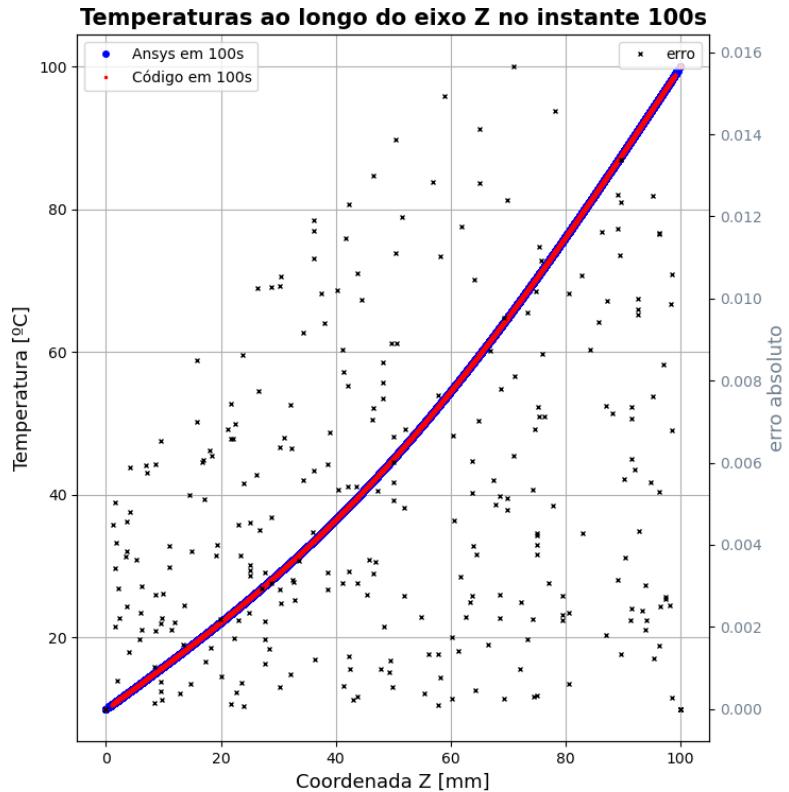


Figura 4.13: Comparação de resultados no instante de 100 segundos

Conforme as simulações se aproximam do regime permanente, o erro entre os resultados tende a zero.

4.2.2 Condição de contorno de Neumann e Robin

Para validar a capacidade do código de impôr um ganho de calor através de um fluxo (condição de contorno de Neumann) e a perda de calor por convecção (condição de contorno de Robin) uma frenagem única ($a = 1.5g$) foi simulada pelo código proposto e pelo *Ansys*. Os dados utilizados estão explicitados na Tabela 4.2.2.

Variável	Nome	Valor	Unid
dt	Time step	0.001	<i>s</i>
$nIter$	Número de iterações	1900	—
θ	Método do MDF	Implícito (1)	—
le	Tamanho médio do elemento	2.5	<i>mm</i>
e	Espessura do disco	5	<i>mm</i>
re	Raio externo do disco	109	<i>mm</i>
ri	Raio interno do disco	68.2	<i>mm</i>
rt	Raio interno da pastilha	80.8	<i>mm</i>
r	Raio dos furos	2	<i>mm</i>
k	Condutividade térmica do disco	49.8	<i>W/m.K</i>
c	Calor específico do disco	486	<i>J/kg.K</i>
ρ	Massa específica do disco	7850	<i>kg/m³</i>
Tin	Temperatura inicial do disco	35	°C
$Tinf$	Temperatura ambiente	35	°C

Tabela 4.3: Dados da simulação

Os coeficientes de convecção e o fluxo de calor utilizados estão mostrados na tabela 4.4.

Tabela 4.4: Fluxo de calor e coeficiente de convecção

tempo	Fluxo de calor	Coef. de convecção
[s]	[W]	[W/m ² K]
0	53178	482.12
0.1	50361	461.58
0.2	47544	440.8
0.3	44727	419.79
0.4	41911	398.5
0.5	39094	376.92
0.6	36277	355.04
0.7	33460	332.81
0.8	30643	310.2
0.9	27826	287.17
1.0	25009	263.67
1.1	22193	239.63
1.2	19376	214.97
1.3	16559	189.58
1.4	13742	163.31
1.5	10925	135.93
1.6	8108	107.09
1.7	5291	76.11
1.8	2475	41.44

Dois instantes foram comparados: o instante que o disco de freios atinge a temperatura máxima e o instante final ($t = 1.9s$). Os resultados do instante final foram comparados na Figura 4.14

Os erros relativos são mostrados na Tabela 4.5.

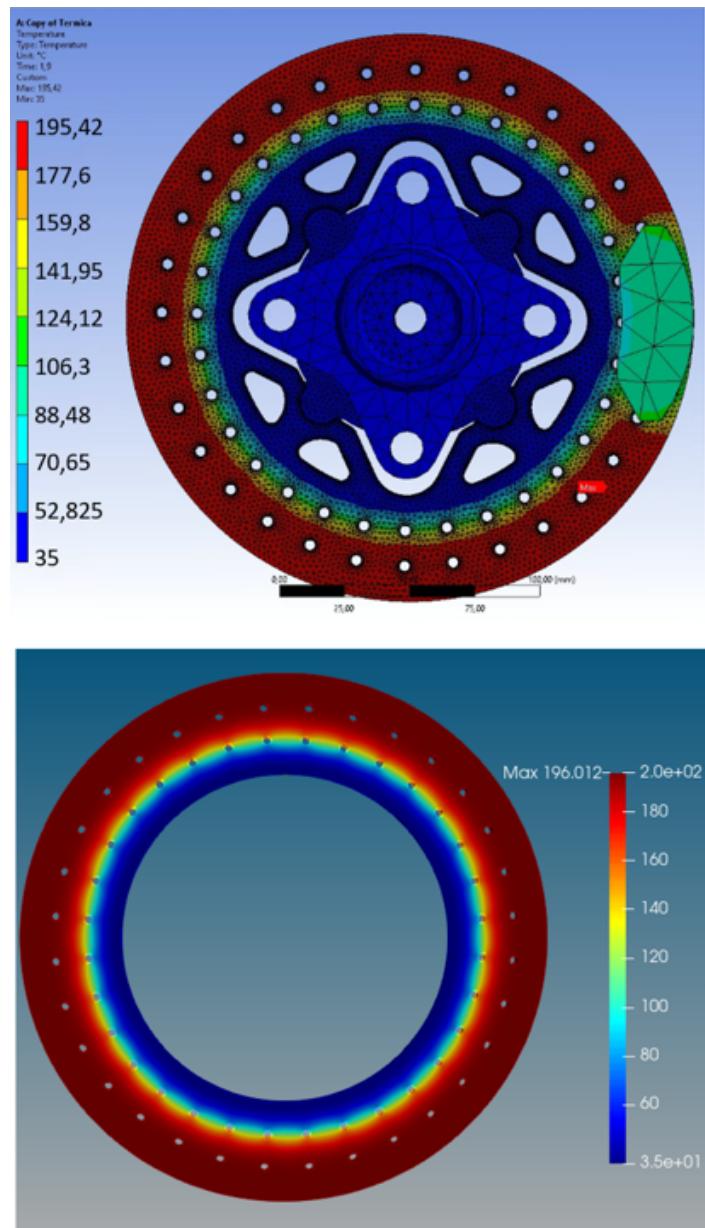


Figura 4.14: Temperatura final do código e do *Ansys*

Fonte: Elaborada pelo autor.

Tabela 4.5: Erros relativos na simulação da frenagem única

Temperatura	Ansys	Código	Erro relativo (%)
Máxima	196.95	196.798	0.077
Final	195.42	196.012	0.303

Capítulo 5

Resultados e Discussões

Neste capítulo, os resultados das simulações utilizando o código proposto serão expostos. Para visualização do gradiente de temperatura sobre a superfície, o software visualizador aberto *Paraview* foi utilizado.

5.1 Utilização de matrizes esparsas

Para trabalhar com códigos tridimensionais, a capacidade computacional é bastante exigida na construção matricial e o tempo de simulação pode ser indevidamente alto. Para contornar isto, as matrizes globais do problema podem ser transformadas para o tipo esparsa, que têm como vantagem a não utilização de memória para valores nulos.

As matrizes esparsas utilizadas são dos seguintes tipos:

- CSR - São matrizes mais eficientes nas operações aritméticas fundamentais (soma, multiplicação, etc);
- LIL - São matrizes eficientes na construção de matrizes esparsas, porém, inefficientes nas operações aritméticas;
- CSC - Assim como a csr, são mais eficientes na solução de operações aritméticas fundamentais.

A biblioteca *tqdm* foi utilizada para medir a velocidade de processamento do código. As etapas de *Assembling* (pricipal 3D e de contorno 2D) são medidas em

iterações por segundo (it/s) enquanto que a solução da equação transiente - que é mais lenta - é medida em segundos por iteração (s/it). Além disso a biblioteca *time* é utilizada para medir o tempo de simulação. O terminal mostra o status e os tempos de interesse da simulação, como mostra a Figura 5.1.

```

Info      : Done writing 'disc.msh'

Preparando simulacao:
100%|██████████| 33318/33318 [00:00<00:00, 680074.66it/s]

Main Assembling:
100%|██████████| 76244/76244 [00:38<00:00, 1969.21it/s]

Boundary Assembling:
100%|██████████| 21690/21690 [00:03<00:00, 6012.08it/s]
100%|██████████| 14462/14462 [00:02<00:00, 6057.65it/s]
100%|██████████| 14462/14462 [00:02<00:00, 6492.88it/s]

Equacao transiente:
100%|██████████| 19/19 [00:29<00:00, 1.56s/it]

Simulacao finalizada

Tempos da simulacao:
Tempo total -> 1.42 min
Construcao da malha -> 6.27 seg
Simulacao -> 1.32 min

```

Figura 5.1: Terminal mostrando a performance de simulação

A tabela 5.1 compara os resultados quando se utiliza as matrizes esparsas e quando não se utiliza em uma simulação com 190 iterações e malha com 55896 elementos.

Tabela 5.1: Tabela comparativa entre o uso de matrizes comuns e esparsas

Matrizes	Assembling [it/s]		[s/it]	Tempo [min]
	3D	2D		
Comuns	6117	5715	87.4	276.77 [*]
Esparsas	2085	6219	1.0	3.19

Nota [*] - Tempo estimado pela velocidade da Eq. transiente.

Para um número mais elevado de elementos não foi possível executar o código com matrizes comuns devido ao espaço em memória ocupado (em um computador com 8GB de memória RAM), já com as matrizes esparsas foi possível.

Na montagem principal (*main Assembling*) foi observada uma maior velocidade na utilização de matrizes comuns. Entretanto após a montagem as matrizes devem ser transformadas em esparsas e isso, consequentemente demanda um tempo indesejado. Com isso foi vantajoso montar a matriz principal já no formato de matriz esparsa.

Mostra-se, portanto, que as matrizes esparsas tornaram o trabalho viável quanto à exigência de memória computacional.

5.2 Frenagem crítica

Conforme Equação 2.34 a frenagem mais eficiente que o protótipo pode atingir tem o valor do coeficiente de atrito entre pista e pneu, medido em g's. conforme curvas do fabricante dos pneus, é dado que o valor deste coeficiente de atrito é de $\mu_P = 1.5 = a$. Portanto, este valor será considerado neste estudo. Os dados do protótipo a serem utilizados no estudo são mostrados na Tabela 5.2.

Variável	Nome	Valor	Unidade
W	Massa do protótipo com piloto	350	kg
L_F	Distância do eixo dianteiro ao CG	757.2	mm
L_R	Distância do eixo traseiro ao CG	782.8	mm
$L = (L_F + L_R)$	Entre-eixos	1540	mm
h	Altura do CG	350	mm
$\chi = \frac{h}{L}$	Relação geométrica do protótipo	0.23	—
$\Psi = \frac{L_F}{L}$	Relação geométrica do protótipo	0.492	—
I	Momento de Inércia do conjunto da roda	0.55	kg/m^2
R_P	Raio efetivo do pneu	254.8	mm
r_e / r_i	Raio externo / interno do disco dianteiro	109 / 68.2	mm
r_t	Raio interno da pastilha	80.8	mm
μ_F	Coeficiente de atrito entre disco e pastilha	0.4	—
A_F	Área do pistão da pinça dianteira	981.748	mm^2
e	Espessura do disco dianteiro	5	mm

Tabela 5.2: Dados do protótipo considerados

Para o estudo, propõe-se a parada do protótipo de uma velocidade de 100 km/h com a desaceleração máxima definida. Com isso a cinemática analisada é resumida na Tabela 5.3.

Velocidade inicial	Desaceleração	Tempo de parada
[m/s]	[m/s ²]	[s]
27.78	14.715 (1.5g)	1.9

Tabela 5.3: Cinemática da frenagem

O material selecionado foi o aço 1045, com as propriedades conforme Tabela 5.4.

Condutividade térmica	Calor específico	Massa específica
[W/m.K]	[J/kg.K]	[kg/m ³]
49.8	486.0	7850.0

Tabela 5.4: Propriedades físicas do material selecionado para o disco de freios

O coeficiente de partição de calor para o disco γ foi calculado em 87%. A temperatura inicial do disco e do ar convectivo foram consideradas de 35°C.

Os parâmetros da simulação são mostrados na Tabela 5.5.

Time step	Iterações	MDF
0.01	190	0.5 (crank nicolson)

Tabela 5.5: Parâmetros gerais da simulação

A pressão exercida no sistema dianteiro do protótipo para gerar a aceleração proposta pode ser calculada conforme demonstrado na seção 2.2.

$$P_F = \frac{N_F}{A_F} = \frac{F_{atF}}{\mu A_F} = \frac{T}{2r_m \mu A_F} \quad (5.1)$$

$$T = F_x R_P + I\alpha \quad (5.2)$$

A força de atrito máxima F_x , por sua vez é calculada conforme a força vertical dinâmica máxima F_z .

$$T = \frac{F_z \mu_P}{2} R_P + I\alpha = \frac{F_z a}{2} R_P + I\alpha \quad (5.3)$$

Com isso, a pressão exercida nas linhas dianteiras tem o valor de $P_F = 7.90$ MPa.

Conforme as Equações 3.33 e 3.34, o fluxo de calor e coeficiente de convecção foram calculados, resultando nos valores da Tabela 5.6. O fluxo de calor em *Watts* é dividido no código pela área de contato entre pastilha e disco de freios para se transformar em fluxo de calor por área [W/m^2].

Tabela 5.6: Fluxo de calor e coeficiente de convecção considerados

tempo [s]	Fluxo de calor [W]	Coeficiente de convecção [W/m ² K]
0	55593.92	103.69
0.1	55593.92	99.25
0.2	52648.89	94.75
0.3	47070.85	90.21
0.4	39590.25	85.60
0.5	31201.24	80.94
0.6	22936.97	76.21
0.7	15646.59	71.41
0.8	9844.55	66.53
0.9	5672.51	61.56
1.0	2968.05	56.49
1.1	1395.76	51.31
1.2	582.43	46.00
1.3	212.19	40.53
1.4	66.06	34.88
1.5	17.07	29.00
1.6	3.51	22.81
1.7	0.53	15.98
1.8	0.05	9.99

A maior temperatura encontrada foi de $144.82^\circ C$, enquanto que a temperatura final foi de $142.10^\circ C$. O resultado da temperatura ao longo do tempo pode ser vista nas Figuras 5.2 e 5.3.

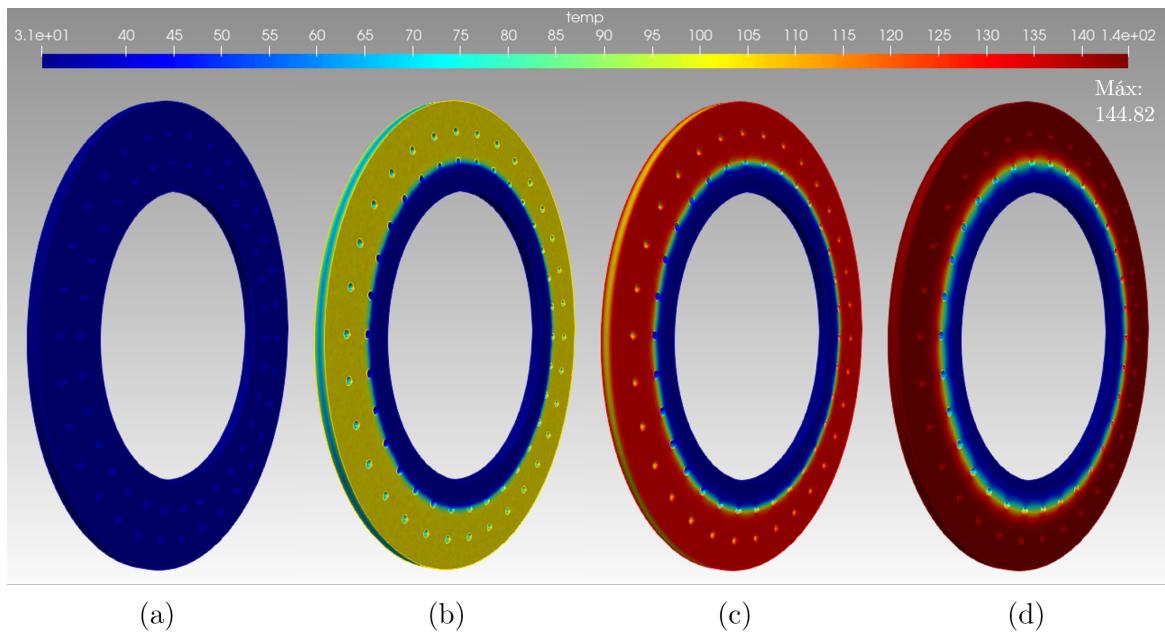


Figura 5.2: Gradiente de temperatura nos instantes (a) $t = 0s$, (b) $t = 0.25s$, (c) $t = 0.50s$, (d) $t = 1.0s$.

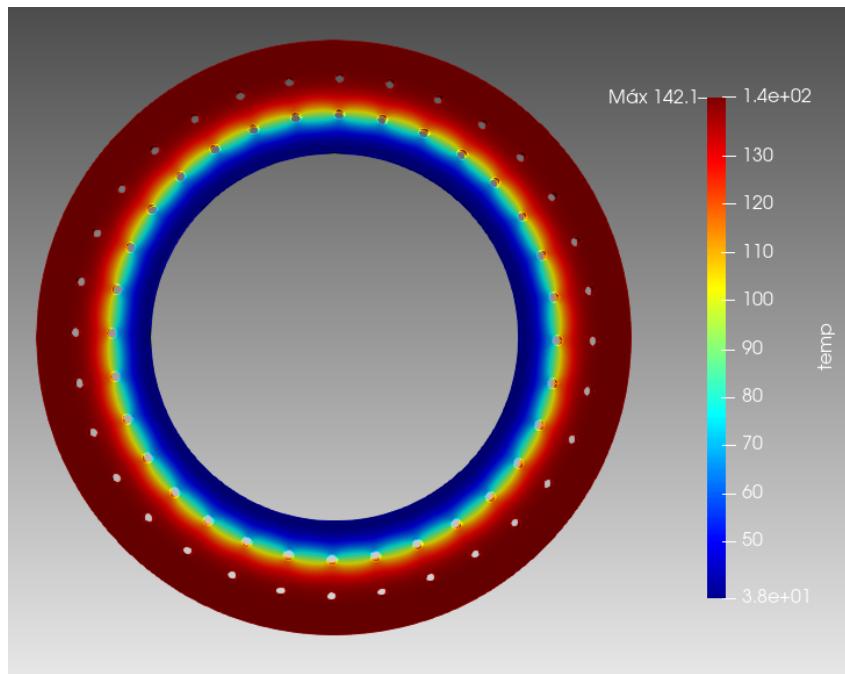


Figura 5.3: Gradiente de temperatura no instante final $t = 1.9s$

5.3 Variação da espessura do disco

Com o intuito de comparar o efeito da variação da espessura do disco de freios na transferência de calor do projeto, foram feitas simulações com os mesmos parâmetros

do caso anterior, variando a espessura entre os valores $4mm$, $5mm$, $10mm$ e $15mm$. Os resultados para as diferentes espessuras do disco de freios são mostrados na Tabela 5.8

Tabela 5.7: Resultados para diferentes espessuras

Simulação	Espessura do disco [mm]	Temperaturas [C]	
		Máxima	Final
#1	4	169.606	168.577
#2	5	144.822	142.096
#3	10	118.230	88.824
#4	15	118.485	73.235

Percebe-se que, com o aumento da espessura, a temperatura máxima atingida neste tipo de frenagem é reduzida pois o calor é conduzido para o interior do disco, como mostra a Figura 5.4. Entretanto, percebe-se que uma espessura maior do que $10mm$ não tem mais eficácia, uma vez que a condução do material tem velocidade limitada. Para múltiplas frenagens, entretanto, o aumento de espessura continua sendo eficaz, uma vez que a temperatura final é mais baixa devido a condução contínua do material.

O resultado pode ser visto na Figura 5.5

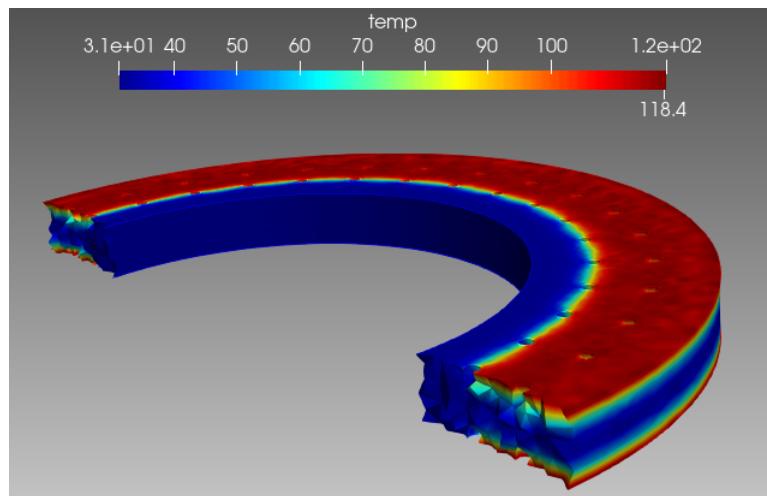


Figura 5.4: Disco mais espesso conduzindo calor

5.4 Variação no raio interno do disco

Na análise térmica, percebe-se que a parte interna do disco tem pouco efeito de condução em uma frenagem de emergência. Portanto uma simulação com os mesmos parâmetros foi feita, aumentando apenas o raio interno do disco para o mesmo valor do raio interno da pastilha $r_i = r_t = 80.8mm$.

Tabela 5.8: Comparaçõe considerando o aumento do raio interno

Simulação	Raio interno do disco [mm]	Temperaturas [C]	
		Máxima	Final
#2	68.2	144.822	142.096
#5	80.8	144.871	142.101

O resultado pode ser visto na Figura 5.5

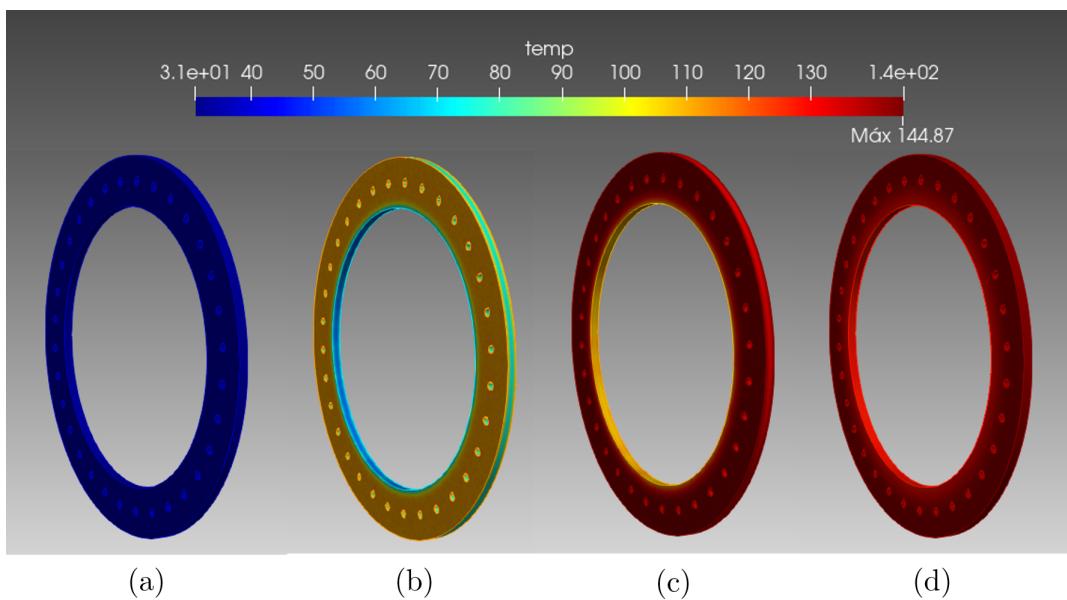


Figura 5.5: Gradiente de temperatura nos instantes (a) $t = 0s$, (b) $t = 0.30s$, (c) $t = 0.80s$, (d) $t = 1.90s$.

Com isso, fica evidente que, na questão térmica, o raio interno do disco não precisa ser menor que o raio interno da pastilha para uma frenagem de emergência.

5.5 Variação na distribuição de calor entre pastilhas

É importante notar que o código proporciona uma riqueza enorme de comparações. Uma delas é simular, em uma situação hipotética, a pastilha de um dos lados do disco fornecendo mais calor comparado à pastilha do lado oposto. Isso pode representar uma situação onde a pinça flutuante de freios esteja com defeito na sua flutuação ou a utilização de pastilhas de freio de diferentes materiais.

Para fazer este tipo de simulação, basta modificar os comandos responsáveis por construir uma lista a ser preenchida com os valores de fluxo de calor nos nós de contorno. Em um exemplo exagerado, se um dos lados vai aquecer 50% a mais que o outro, pode-se compensar nestas listas, como mostrado abaixo:

```
1 qi = np.zeros((npoints), dtype='float')
2 for b in bound1:
3     qi[b] = 1.0      # modificar para qi[b] = 1.5
4 for b in bound2:
5     qi[b] = -1.0    # modificar para qi[b] = -0.5
```

O sinal negativo se deve ao fato de que cada lista representa as superfícies de contorno opostas. Se ambos foram positivos, em um dos lados haverá ganho de calor e no outro perda. Os resultados podem ser vistos na Tabela 5.9 e na Figura 5.6

Tabela 5.9: Simulação de fluxo de calor desigual entre as pastilhas de freio

Simulação	Distribuição do calor nas pastilhas	Temperaturas [C]	
		Máxima	Final
#2	1.0/1.0	144.822	142.096
#6	1.5/0.5	169.216	142.158

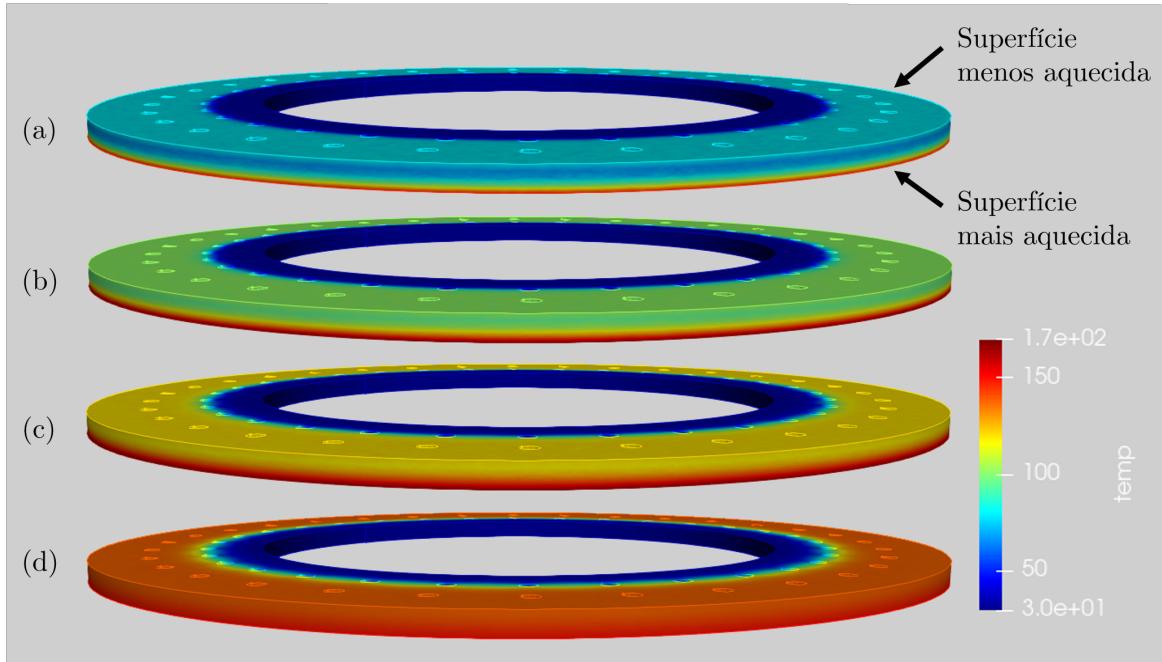


Figura 5.6: Temperatura desigual nos instantes (a) $t = 0.3s$, (b) $t = 0.5s$, (c) $t = 0.7s$, (d) $t = 1.1s$.

5.6 Múltiplas frenagens

Para estudar condições mais extremas de temperatura, múltiplas frenagens são simuladas. De acordo com as provas mais exigentes da competição de Formula SAE, a frenagem partindo de uma velocidade de $60km/h$ ocorre frequentemente. Para fazer um estudo mais conservador, propõe-se a frenagem até a velocidade nula. Os parâmetros físicos do protótipo são os mesmos da Tabela 5.2 e os parâmetros das múltiplas frenagens são apresentados na Tabela 5.10.

Time step	Iterações	MDF
0.5	760	0.5 (crank nicolson)

Tabela 5.10: Parâmetros gerais da simulação

Para simular de forma extrema, foram propostos 100 ciclos de frenagens, considerando 5 segundos de aceleração e 4 segundos de frenagem. O fluxo de calor e o coeficiente de convecção foram calculados de acordo com a velocidade do veículo, considerando uma pressão de 1.61 MPa no sistema. Os dados utilizados em 1 ciclo são apresentados na Tabela 5.11

Tabela 5.11: Dados propostos para 1 ciclo de frenagem

Tempo	Aceleração	Velocidade	Fluxo de calor	Coef. de convecção
[s]	[m/s ²]	[m/s]	[W]	[W/m ² K]
0	3.33	0.00	0.00	0.00
0.5	3.33	1.67	0.00	10.18
1	3.33	3.33	0.00	16.38
1.5	3.33	5.00	0.00	22.70
2	3.33	6.67	0.00	28.62
2.5	3.33	8.33	0.00	34.25
3	3.33	10.00	0.00	39.66
3.5	3.33	11.67	0.00	44.91
4	3.33	13.33	0.00	50.00
4.5	3.33	15.00	0.00	54.97
5	3.33	16.67	0.00	59.84
5.5	-4.12	14.61	6785.84	59.84
6	-4.12	12.55	6107.26	53.81
6.5	-4.12	10.49	4885.81	47.61
7	-4.12	8.43	3420.06	41.21
7.5	-4.12	6.37	2052.04	34.56
8	-4.12	4.31	1026.02	27.58
8.5	-4.12	2.25	410.41	20.13
9	-4.12	0.19	123.12	12.24

Os resultados do primeiro e segundo ciclos podem ser vistos nas Figuras 5.7 e 5.8 respectivamente.

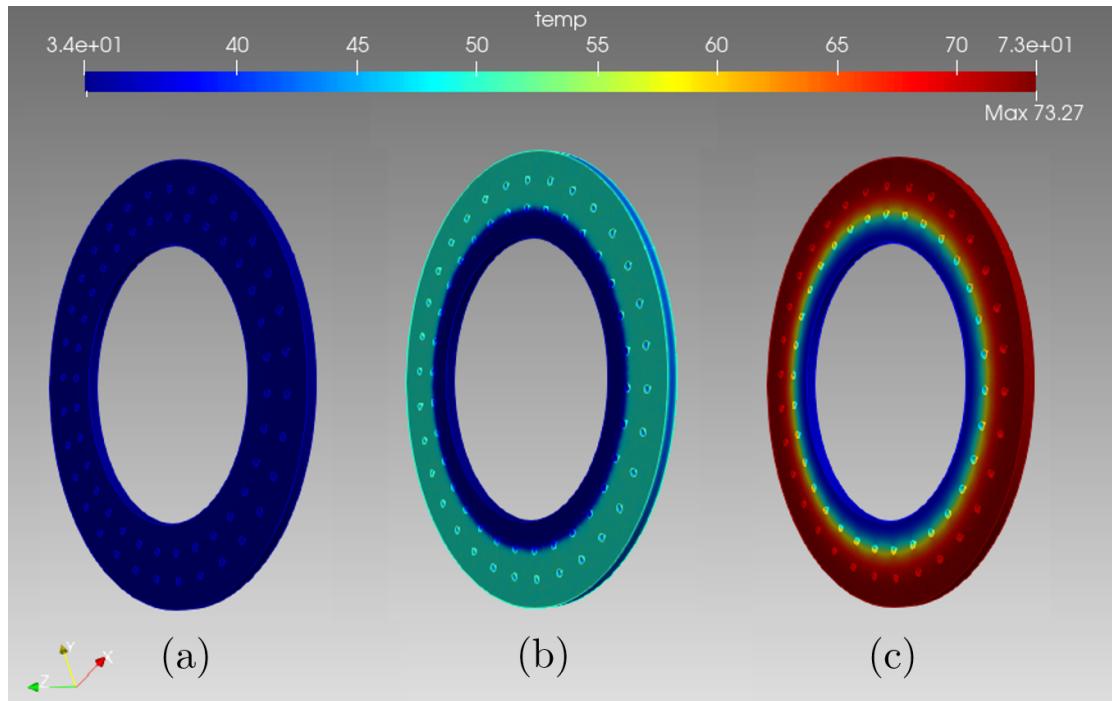


Figura 5.7: Temperatura nos instantes (a) $t = 3s$, (b) $t = 6s$, (c) $t = 9s$

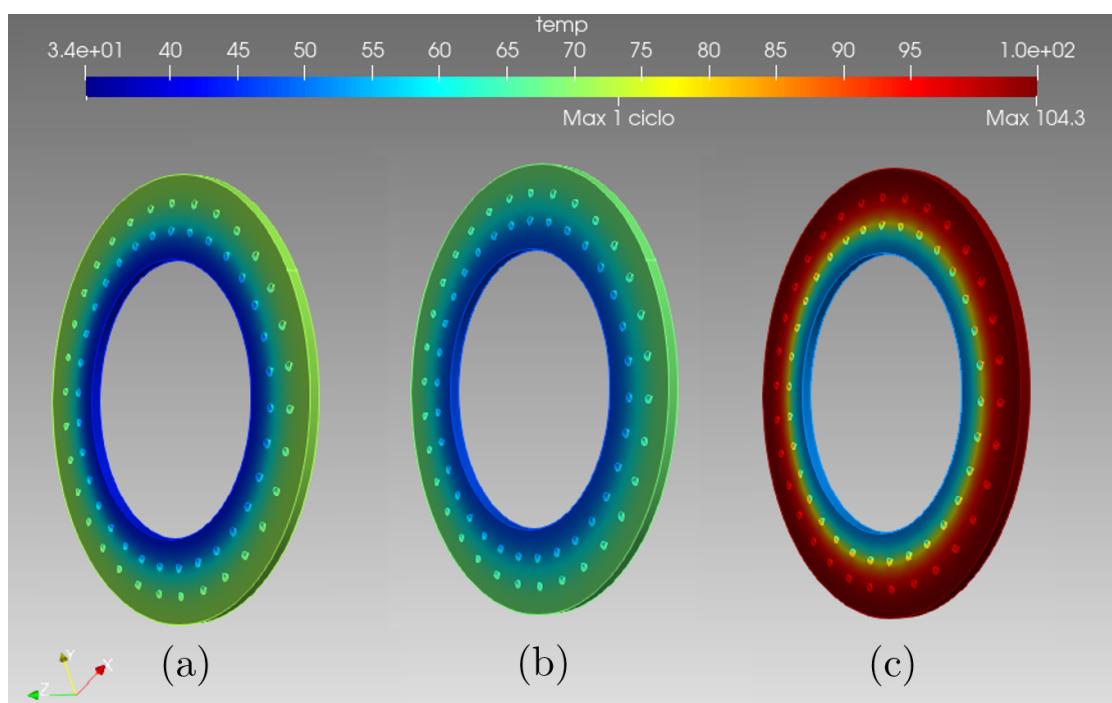


Figura 5.8: Temperatura nos instantes (a) $t = 12s$, (b) $t = 15s$, (c) $t = 18s$

Em cada ciclo, a temperatura inicialmente decresce devido a convecção e condução e depois aumenta devido à nova frenagem, como mostrado na Figura 5.9.

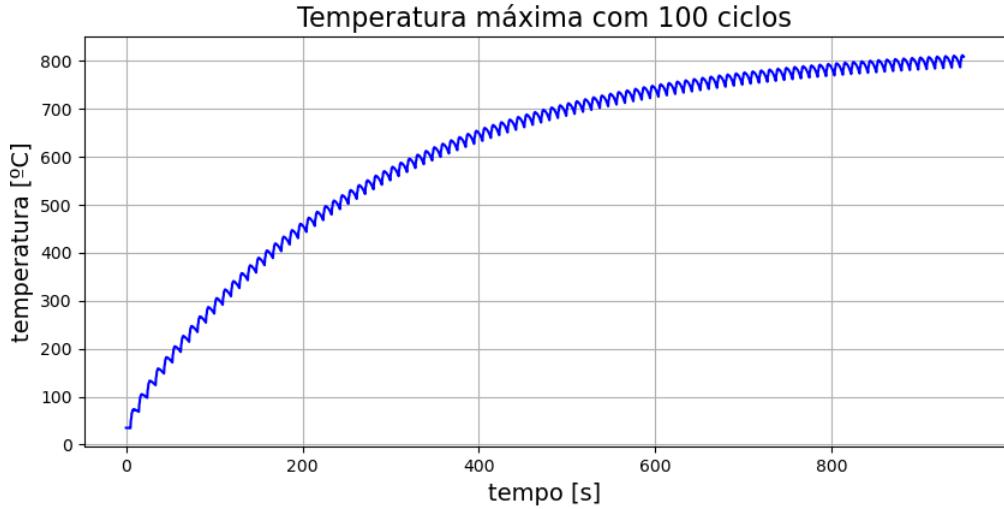


Figura 5.9: Temperatura em todos os ciclos

A temperatura média do disco tem um comportamento assintótico, até atingir uma faixa de temperatura máxima para a condição proposta. A temperatura máxima atingida está apresentada na Tabela 5.9.

Tabela 5.12: Temperaturas atingidas pelo disco de freio dianteiro em frenagens múltiplas

Simulação	Temperaturas [°C]	
	Máxima 1 ciclo	Máxima 100 ciclos
#7	73.27	811.85

Vale ressaltar que as condições simuladas na frenagem múltipla representa uma tratativa exagerada para estudo da temperatura do disco, uma vez que o protótipo não é submetido a um número tão elevado de ciclos de frenagem, e também não são frenagens até a velocidade nula.

Além das comparações propostas, o código pode ser utilizado para gerar diversos comparativos, como por exemplo:

- Variações na espessura do disco para múltiplas frenagens;

- Comparação de performance de diferentes materiais para o projeto;
- Variação no raio externo e interno do disco e da pastilha para ambos os tipos de frenagem;
- Variação no raio dos furos do disco;
- Nos cálculos, alterar o coeficiente de partição de calor entre disco a pastilha, o que vai resultar na mudanças do fluxo de calor para o disco;
- Simular maiores valores de velocidade do ar para o cálculo de coeficientes de convecção em projetos de túnel de vento direcionado ao disco (*brake cooling duct*).

Capítulo 6

Conclusões

O código proposto que utiliza o Método de Elementos Finitos em *Python* mostra que o método numérico é uma alternativa acessível a todas as pessoas interessadas, e fornece uma ferramenta poderosa de solução dos mais diversos problemas da engenharia. Além do *Python*, softwares abertos fornecem soluções completas para a construção dos mais diversos tipos de geometrias e malhas, permitindo pós processamento e visualização das soluções numéricas.

O código apresentou estabilidade na convergência dos modelos, entretanto é necessário ter cuidado na construção matricial dos problemas para que a exigência de memória dos computadores não seja muito acima da razoável. Matrizes esparsas se mostraram viáveis nesta função.

Os resultados encontrados a partir do método numérico foram satisfatórios quando comparados com solução analítica e simulação por software profissional, tendo erros relativos em torno de 1%. Os valores resultantes do fluxo de calor, convecção e condução do disco se mostraram coerentes com medidas reais de sensores de temperatura utilizados nos protótipos de Formula SAE.

A ferramenta desenvolvida permite a fácil variação geométrica dos projetos de discos de freio, proporcionando uma facilidade em estudos comparativos e rapidez em tomadas de decisão. A ferramenta ainda permite a alteração total da geometria, onde é possível adaptar o código para a utilização em outros projetos. Torna-se portanto possível a utilização da ferramenta criada para estudos diversos de transferência de calor em sólidos.

6.1 Sugestões para trabalhos futuros

Este trabalho se transforma em um projeto em constante desenvolvimento. Alguns pontos que podem ser implementados são:

- Utilização de um método mais abrangente no momento da definição da condição de contorno - como o código proposto calcula as matrizes de massa do contorno triangular baseado na premissa de que a superfície de contorno se encontra inteiramente na mesma coordenada z (Equação 2.67), o estudo de convecção no interior dos furos do disco de freios fica inviabilizado;
- Construir geometrias de disco de freio mais complexas, como discos de freios ventilados;
- Estudos de melhoria de performance deste tipo de código e implementação de tratativas de erros, como erro de leitura de input, erro de execução e divergência de resultados.

Além do escopo proposto, os passos utilizados na implementação do Método de Elementos Finitos podem ser aplicados em outras equações diferenciais, como por exemplo:

- Equações termomecânica, unindo a questão térmica com as tensões mecânicas;
- Equações de estudos de rigidez axial, flexional e torcional;
- Equações referentes a hidrodinâmica de barcos ou aerodinâmica de aerofólios;
- Equações de vibração.

Referências Bibliográficas

- [1] OZISIK, M., *Heat Transfer, a Basic Approach*. 1985.
- [2] INCROPERA, *Fundamentos de Transferência de Calor e de Massa*. v. 6^a EDIÇÃO.
- [3] LIMPERT, R., *Brake Desing and Safety*. v. third edition.
- [4] TALATI, F., “Analysis of heat conduction in a disk brake system”, 2009.
- [5] LOGAN, D., *A First Course in the Finite Element Method*. v. 4th edition. 2007.
- [6] PEREIRA, D., TORII, A., “Uma Introdução ao Método dos Resíduos Ponderados”. 11 2015.
- [7] ANJOS, G., MANGIAVACCHI, N., BORHANI, N., et al., “3D ALE Finite-Element Method for Two-Phase Flows With Phase Change”, *Heat Transfer Engineering*, v. 35, n. 5, 2014.
- [8] YEVTSHENKO, A. A., GRZES, P., “The FEM-Modeling of the Frictional Heating Phenomenon in the Pad/Disc Tribosystem (A Review)”, *Numerical Heat Transfer, Part A: Applications*, v. 58, n. 3, pp. 207–226, 2010.
- [9] MAJCHERCZAK D, DUFRÉNOY P, N.-A. M., “THE COMPARABLE ANALYSIS OF TEMPERATURE DISTRIBUTIONS ASSESSMENT IN DISC BRAKE OBTAINED USING ANALYTICAL METHOD AND FE MODEL”, 2011.
- [10] JIANYONG, Z., “Analysis of temperature field of axle disc brake equipment for high-speed locomotive”, 2012.

[11] GEUZAIN, C., REMACLE, J.-F., “Gmsh 4.8.3 - <https://gmsh.info/doc/texinfo/gmsh.html>”, .

Apêndice A

Código Fonte

A.1 Código principal

A.1.1 Importação de módulos

Os módulos estão organizados em uma pasta separada. Para ser possível chamar os módulos, basta criar um arquivo `__init__.py` no interior desta pasta. Com isso, o programa chama as bibliotecas e módulos da seguinte forma:

```
1 import time
2 import meshio
3 import numpy as np
4 from tqdm import tqdm
5 from scipy.sparse.linalg import spsolve
6 from scipy.sparse import lil_matrix, csr_matrix, issparse
7 from modulos.malha import disc, polar, boundf, contornoDisco
8 from modulos.montagem import assembling3D, assembling2D
9 from modulos.input import inputInfo
```

A.1.2 Módulo principal

```
#####
2 # 1) Leitura do input
#####
4 fileIpt = 'Input.xlsx'           # Nome do arquivo de input
```

```

1  ipt      = inputInfo(fileIpt)          # Chamada da funcao
2  param    = ipt.getParam()              # Parametros da simulacao
3  geom     = ipt.getGeom()              # Parametros da geometria do disco
4  prop     = ipt.getProperties()        # Propriedades termicas do disco
5  ambt     = ipt.getAmb()               # Temperatura inicial e do ar
6  qcoefs   = ipt.getCoefs('q')         # Fluxo de calor
7  hcoefs   = ipt.getCoefs('h')         # Coeficientes de conveccao
8
9
10
11
12 #####
13
14 # 1.1) Parametros da simulacao
15 #####
16 dt      = param[0]                   # Time step
17 nIter  = int(param[1])              # Numero de iteracoes
18 theta   = param[2]                   # Metodo dif. finitas - imp = 1.0;
19 # - exp = 0.0;
20 # - c.n = 0.5
21 #####
22
23 # 1.2) Parametros fisicos
24 #####
25 Tin    = ambt[0]                   # Temp inicial do solido [C]
26 Tinf   = ambt[1]                   # Temp do ar na conveccao [C]
27 #####
28
29 # 1.3) Dados do Disco de freio
30 #####
31 e      = geom[0]/1000              # Espessura do disco [m]
32 re    = geom[1]/1000              # Raio externo do disco [m]
33 ri    = geom[2]/1000              # Raio interno do disco [m]
34 rt    = geom[3]/1000              # Raio interno da pastilha [m]
35 r     = geom[4]/1000              # Raio dos furinhos [m]
36 k     = prop[0]                  # Condut termica disco [W/m.K]
37 cv    = prop[1]                  # Calor especifico do disco [J/kg.K]
38 rho   = prop[2]                  # Massa especifica do disco [kg/m^3]
39 Ad    = np.pi*((re**2)-(rt**2))  # Area do disco [m^2]
40 Afur  = 32*np.pi*(r**2)         # Area dos furinhos
41 Ad = Ad - Afur                  # Area disco descontando os furinhos
42

```

```

44 ######
# 2) Malha
45 #####
46 ######
# Gerar malha no API do GMSH
47 le_min = param[3]/1000           # Tamanho minimo dos elementos [m]
48 le_max = param[3]/1000           # Tamanho maximo dos elementos [m]
49 filename = 'disc.msh'
50 disc(re, rt, ri, e, le_min, le_max, filename, furos=[r,(11.25/180)*np.pi])
51
52 ######
# Salvando o tempo levado para construcao da malha
53 meshTime = time.time() - startTime
54
55 ######
56 ######
# 2.1) Leitura das matrizes da malha
57 ######
58 ######
59 msh = meshio.read(filename)
60 X = msh.points[:, 0]           # Coordenada x dos nohs
61 Y = msh.points[:, 1]           # Coordenada y dos nohs
62 Z = msh.points[:, 2]           # Coordenada z dos nohs
63 npoints = len(X)              # Numero de nos
64
65 pol = polar(X,Y,Z)
66 ang = pol[0]
67 raio = pol[1]
68
69 ######
70 # 2.2) Matriz de conectividade (IEN)
71 #####
72 boundParam = boundf(filename)
73 IENbound = boundParam[0]
74 IEN = boundParam[1]
75 ne = len(IEN)
76
77 ######
78 # 2.3) Nohs de contorno
79 #####
80 boundDisco = contornoDisco(IENbound, raio, Z, rt, e)
81 bound1 = boundDisco[0]

```

```

82 bound2 = boundDisco [1]
IENboundG1 = boundDisco [2]
84 IENboundG2 = boundDisco [3]
IENboundG = IENboundG1 + IENboundG2
86 IENconv1 = boundDisco [4]
IENconv2 = boundDisco [5]

88

90 """
#####
92 # 3) Vetores para condicao de contorno
#####
94 """
95 print( '\nPreparando simulacao: ')
96 Tc = np.zeros(( npoints ), dtype='float ')
97 qi = np.zeros(( npoints ), dtype='float ')
98 for b in tqdm(bound1):
99     qi [b] = 1.0      # Neumann - vetor para fluxo de calor
100 for b in bound2:
101     qi [b] = -1.0     # Neumann - vetor para fluxo de calor
102 for b in IENconv1:
103     Tc [b] = -Tinf    # Robin
104 for b in IENconv2:
105     Tc [b] = Tinf     # Robin

106

108 """
#####
110 # 4) Assembling ( matrizes K, M, MC e Mh)
#####
112 """
113 print( '\nMain Assembling: ')
114 main = assembling3D (IEN, npoints , ne ,X=X, Y=Y, Z=Z , k=k)
115 K = main [0]
116 M = main [1]

117 print( '\nBoundary Assembling: ')
118 MC = assembling2D (IENboundG , npoints ,X=X, Y=Y, Z=Z , c=1)

119
120

```

```

# Matrizes de convecção precisam ter sinal
122 Mh1 = assembling2D(IENconv1, npoints, X=X, Y=Y, Z=Z, c= 1)
123 Mh2 = assembling2D(IENconv2, npoints, X=X, Y=Y, Z=Z, c=-1)
124 Mhin = Mh1 + Mh2

126
127 ,,
128 #####
# 5) Sistema linear
130 #####
131 ,,
132 # change to csr: efficient arithmetic operations as CSR + CSR, CSR *
133     CSR, etc.
134 M = M.to csr()
135 K = K.to csr()
136 MC = MC.to csr()
137 Mhin = Mhin.to csr()

138 # Lado esquerdo do sistema
139 A = rho*cv*M + (theta)*dt*(K) # + (theta)*dt*Mh nas iteracoes
140 A = A.to lil()
141 A2 = A.copy()
142
143 # Criação das listas das variáveis
144 b = np.zeros((npoints), dtype='double') # Lado direito da eq
145 T = Tin*np.ones((npoints), dtype='double') # Temperaturas iniciais
146
147 # Salva resultados iniciais para visualização no Paraview
148 point_data = { 'temp' : T}
149 meshio.write_points_cells(f'sol-0.vtk', msh.points,
150                           msh.cells, point_data=point_data,)

152
153 #####
154 # 5.1) Preparando listas de coeficientes (q e h)
155 #####
156 ciclos = int(ipr.ciclos())
157 rel = nIter/(len(qcoefs))
158

```

```

160
161 qcoefs2 = []
162 hcoefs2 = []
163 if ciclos > 1:
164     print('Frenagem multipla')
165     rel = nIter/ciclos
166     qcoefs2 = []
167     hcoefs2 = []
168     for r in range(int(ciclos)):
169         qcoefs2.extend(qcoefs)
170         hcoefs2.extend(hcoefs)
171     else:
172         for i in range(len(qcoefs)):
173             for r in range(int(rel)):
174                 qcoefs2.append(qcoefs[i])
175                 hcoefs2.append(hcoefs[i])
176
177 ##### 5.2) Iteracoes no tempo #####
178 maxT = []
179 print('\nEquacao transiente:')
180 for n in tqdm(range(nIter)):
181     h = hcoefs2[n][0]
182     qm = qcoefs2[n][0]/(2*Ad)
183     q = qm*qi # Vetor de fluxo de energia
184
185     Mh = h*MC
186     Mh2 = h*Mh
187
188     # Lado esquerdo da equacao (incluindo a conveccao)
189     A = A2 + (theta)*dt*Mh2
190
191     # Lado direito da equacao
192     f = rho*cv*M - (1-theta)*dt*(K+Mh2)
193     b = f.dot(T) + dt*Mh.dot(Tc) + dt*MC.dot(q) # Q=0
194
195     # Solucao do sistema linear (AT=b)
196     T = spsolve(A.tocsc(), b)

```

```

198 maxT.append(max(T))
# Salva resultados para visualizacao no Paraview
200 point_data = { 'temp' : T}
meshio.write_points_cells(f'sol-{n+1}.vtk',msh.points,
202 msh.cells,point_data=point_data,)

204 print('\nSimulacao finalizada')

206 ,
207
208 ##### # 6) Tempos de simulacao e Resultados em .txt
209 #####
210 #####
211 ,
212 totalTime = time.time()/60 - startTime/60
print(f'\n\nTempos da simulacao:')
214 print(f'Tempo total -> {round(totalTime, 2)} min')
print(f'Construcao da malha -> {round(meshTime, 2)} seg')
216 print(f'Simulacao -> {round(totalTime - meshTime/60, 2)} min')

218 with open("Results.txt", "a") as f:
    f.write(f'Simulacao realizada por: {ipt.header[0]} em {ipt.header[2]}\n\n')
    f.write(f'Temperatura maxima = {max(maxT)}\n')
    f.write(f'Temperatura max final = {max(T)}\n\n')
    f.write(f'Tempo total -> {round(totalTime, 2)} min\n')
    f.write(f'Construcao da malha -> {round(meshTime, 2)} seg\n')
224 f.write(f'Simulacao -> {round(totalTime - meshTime/60, 2)} min\n')
    f.write('Para melhor visualizacao dos resultados, utilize o\nParaview')

```

A.2 Módulo da leitura do input

```

# Modulo para leitura da planilha de input.

2
class inputInfo():

```

```
4
# filename = string containing the excel input filename location
6
def __init__(self, filename):
    import pandas as pd
8
try:
    headerList = pd.read_excel(
        filename,
        sheet_name='Input',
        skiprows=1,
        usecols='P',
        nrows=3,
        header=None,
        # dtype=object
        ).T
10
12
14
16
18
20
projectInfo = pd.read_excel(
    filename,
    sheet_name='Input',
    skiprows=5,
    usecols='B,E,G,H,J,L,O',
    nrows=1,
    # dtype=object
    )
22
24
26
28
filledData = pd.read_excel(
    filename,
    sheet_name='Input',
    skiprows=10,
    usecols='B:P',
    # dtype=object
    )
30
32
34
36
heatFlux = pd.read_excel(
    filename,
    sheet_name='Coefs',
    skiprows=2,
    usecols='C',
    # dtype=object
    )
38
40
42
```

```

        )

44
    convCoef = pd.read_excel(
46
        filename ,
47
        sheet_name='Coefs' ,
48
        skiprows=2,
49
        usecols='D' ,
50
        # dtype=object
51
    )

52
# except FileNotFoundError:
53
#     raise InputReadError(filename)

56
except:
57
    # Exception to get possible modules not found
58
    raise

60
self.header = headerList.values.tolist()[0]
61
self.project = projectInfo.values.tolist()[0]
62
self.read = filledData.values.tolist()[0]
63
self.q = heatFlux.values.tolist()
64
self.h = convCoef.values.tolist()

66
# Retorna os parametros da simulacao
67
def getParam(self):
68
    return self.read[1:5]

69
# Retorna os inputs do disco
70
def getGeom(self):
71
    return self.read[5:10]

72
# Retorna as propriedades do disco
73
def getProperties(self):
74
    return self.read[10:13]

75
# Retorna as temperaturas inicial e ambiente
76
def getAmb(self):
77
    return self.read[13:15]

```

```

82
83     # Retorna as temperaturas inicial e ambiente
84     def getCoefs(self, coef):
85         if coef == 'q':
86             return self.q
87         elif coef == 'h':
88             return self.h
89
90     # Retorna as informacoes do projeto
91     def projectInfo(self):
92         return self.project
93
94     # Retorna o numero de ciclos
95     # Para a frenagem unica = 1
96     # Para multiplas frenagens > 1
97     def ciclos(self):
98         if self.project[3] == "Unica":
99             return 1
100        else:
101            return self.project[4]

```

A.3 Módulo da construção e leitura da malha

A.3.1 Malha de discos de freio

```

1 import numpy as np
2 import meshio
3 import gmsh
4
5 def disc(re, rt, ri, e, le_min, le_max, filename, furos):
6     # Before using any functions in the Python API, Gmsh must be
7     # initialized:
8     gmsh.initialize()
9     gmsh.model.add("minha_malha")
10
11    # in order to output messages on the terminal, just set the

```

```

# "General.Terminal" option to 1:
12 gmsh.option.setNumber("General.Terminal", 1)

14 # Adicionar cilindros para construir o disco
15 # addCylinder(x, y, z, dx, dy, dz, raio, tag)
16 cylinder = gmsh.model.occ.addCylinder(0, 0, 0, 0, 0, e, re, tag=1)
17 cylinder = gmsh.model.occ.addCylinder(0, 0, 0, 0, 0, e, ri, tag=2)

18
19 if furos != 'no':
20     # furos eh um lista com [raio do furinho, angulo entre furos]
21     r    = furos[0]
22     ang = furos[1]
23     d = re-(13.6/1000)
24     p = ri+(13.6/1000)
25     coordIn = [d,p]

26
27 coord = []
28 for i in range(1,9):
29     coord.append([d*np.cos(i*ang), d*np.sin(i*ang)])
30 for i in range(1,9):
31     coord.append([p*np.cos(i*ang), p*np.sin(i*ang)])

32
33 t = 4
34 for c in coordIn:
35     gmsh.model.occ.addCylinder(c, 0, 0, 0, 0, e, r, tag=t)
36     gmsh.model.occ.addCylinder(-c, 0, 0, 0, 0, e, r, tag=t+1)
37     gmsh.model.occ.addCylinder(0, c, 0, 0, 0, e, r, tag=t+2)
38     gmsh.model.occ.addCylinder(0,-c, 0, 0, 0, e, r, tag=t+3)
39     t = t+4

40
41 for c in coord:
42     gmsh.model.occ.addCylinder(c[0], c[1], 0, 0, 0, e, r, tag=t)
43     gmsh.model.occ.addCylinder(-c[0], c[1], 0, 0, 0, e, r, tag=t+1)
44     gmsh.model.occ.addCylinder(-c[0], -c[1], 0, 0, 0, e, r, tag=t+2)
45     gmsh.model.occ.addCylinder(c[0], -c[1], 0, 0, 0, e, r, tag=t+3)
46     t = t+4

47
48 gmsh.model.occ.cut([(3, 1)], [(3, i) for i in range(4,t)], 3)
49 # furinhos

```

```

        domain = gmsh.model.occ.cut([(3, 3)], [(3, 2)], 4) # miolo do
        disco

50
else:
    gmsh.model.occ.cut([(3, 1)], [(3, 2)], 4) # apenas miolo do
    disco

54
# sincronizar o modelo
56 gmsh.model.occ.synchronize()
volumes = gmsh.model.getEntities(dim=3)

58
# Definir os tamanhos dos elementos da malha
60 gmsh.option.setNumber("Mesh.CharacteristicLengthMin", le_min)
62 gmsh.option.setNumber("Mesh.CharacteristicLengthMax", le_max)

64
# Gerar malha
gmsh.model.mesh.generate(3) # 3D

66
# Salvar arquivo .msh
gmsh.write(filename)
# gmsh.option.setNumber("Mesh.SaveAll", 1)
gmsh.finalize()

70

72 def polar(X,Y,Z):
    npoints = len(X) # Numero de nos
74

    # listas com angulo e raio de cada noh
    ang = np.zeros((npoints), dtype='float')
    raio = np.zeros((npoints), dtype='float')
    for p in range(len(X)):
        raio[p] = np.sqrt(X[p]**2 + Y[p]**2)
        a = np.arctan2(Y[p], X[p])
        if a < 0:
            a = a + 2*np.pi
        ang[p] = a

82
84     return ang, raio

```

```

86

88 def boundf(filename):
  msh = meshio.read(filename)
  IENbound = [] # IEN de nohs do contorno
  for elem in msh.cells:
    if elem[0] == 'triangle': # Elementos triangulares
      IENbound.append(elem[1])
    elif elem[0] == 'tetra': # Elementos tetraedricos
      IEN = elem[1] # Matriz de conectivide IEN
  return IENbound, IEN

96

98

100 def contornoDisco(IENbound, raio, Z, rt, e):
  bound1 = [] # Lista com os nohs da 1a superficie das ccs
  bound2 = [] # Lista com os nohs da 2a superficie das ccs
  IENboundG1 = [] # IEN de fluxo de calor (contorno de neumann)
  IENboundG2 = [] # IEN de fluxo de calor (contorno de neumann)
  IENconv1= [] # IEN de conveccao (contorno de robin)
  IENconv2= [] # IEN de conveccao (contorno de robin)

  for elem in IENbound[1]:
    aux1 = []
    aux2 = []
    for noh in elem:
      if Z[noh] == e:
        aux2.append(noh)
      # apenas nohs no contato com pastilha
      if raio[noh] > rt:
        bound1.append(noh)
        aux1.append(noh)

  if len(aux1) == 3:
    IENboundG1.append(aux1)
  if len(aux2) == 3:
    IENconv1.append(aux2)

  for elem in IENbound[2]:

```

```

126     aux1 = []
127     aux2 = []
128     for noh in elem:
129         if Z[noh] == 0:
130             aux2.append(noh)
131             # apenas nohs no contato com pastilha
132             if raio[noh] > rt:
133                 bound2.append(noh)
134                 aux1.append(noh)
135             if len(aux1) == 3:
136                 IENboundG2.append(aux1)
137             if len(aux2) == 3:
138                 IENconv2.append(aux2)
139
140     return bound1, bound2, IENboundG1, IENboundG2, IENconv1, IENconv2

```

A.3.2 Malha de paralelepípedos

```

1 def cube(Lx, Ly, Lz, le, filename):
2     # Before using any functions, Gmsh must be initialized:
3     gmsh.initialize()
4
5     # in order to output messages on the terminal, just set the
6     # "General.Terminal" option to 1:
7     gmsh.option.setNumber("General.Terminal", 1)
8
9     # iniciando a malha
10    gmsh.model.add("minha_malha")
11
12    # pontos:
13    # gmsh.model.geo.addPoint(x, y, z, target mesh size, tag)
14    gmsh.model.geo.addPoint(0, 0, 0, le, 1)
15    gmsh.model.geo.addPoint(Lx, 0, 0, le, 2)
16    gmsh.model.geo.addPoint(Lx, Ly, 0, le, 3)
17    gmsh.model.geo.addPoint(0, Ly, 0, le, 4)
18
19    # linhas:

```

```

20 # gmsh.model.geo.addLine(ponto_inicial, ponto_final, tag)
21 gmsh.model.geo.addLine(1, 2, 1)
22 gmsh.model.geo.addLine(2, 3, 2)
23 gmsh.model.geo.addLine(3, 4, 3)
24 gmsh.model.geo.addLine(4, 1, 4)

26 # superficies:
27 gmsh.model.geo.addCurveLoop([1, 2, 3, 4], 1)
28 gmsh.model.geo.addPlaneSurface([1], 1)

30 # physical group
31 ps = gmsh.model.addPhysicalGroup(2, [1], 6)
32 gmsh.model.setPhysicalName(2, ps, "My surface")

34 # extrudar:
35 # e = gmsh.model.geo.extrude([(dim, tag)], 0, 0, h)
36 h = Lz # geometry height in the z-direction
37 e = gmsh.model.geo.extrude([(2, 1)], 0, 0, h)

38 # Physical groups are collections of model entities and are
39 identified
40 # by their dimension and by a tag.
41 # Whole domain:
42 domain_tag = e[1][1]
43 domain_physical_tag = 1001 # the volume
44 gmsh.model.addPhysicalGroup(dim=3, tags=[domain_tag], tag=
45 domain_physical_tag)
46 gmsh.model.setPhysicalName(dim=3, tag=domain_physical_tag, name="Whole domain")

48 # O volume tem planos de contorno. Os planos tem retas. As retas
49 # tem pontos
50 planos = gmsh.model.getBoundary(dimTags=[(3, domain_tag)], 
51 oriented=False)

52 # sincronizar o modelo
53 gmsh.model.geo.synchronize()

54 # Gerar malha
55 gmsh.model.mesh.generate(3) # 3D

```

```

56 # Salvar arquivo .msh
57 gmsh.write(filename)
58 # gmsh.option.setNumber("Mesh.SaveAll", 1)
59 gmsh.finalize()

```

A.4 Módulo da montagem

```

2 from tqdm import tqdm
3 from modulos.matrizes import matriz3D, matriz2D
4 from scipy.sparse import lil_matrix, csr_matrix, issparse
5
6 def assembling3D(IEN, npoints, ne, X, Y, Z, k):
7     # LIL is a convenient format for constructing sparse matrices
8     K = lil_matrix((npoints, npoints), dtype='double')
9     M = lil_matrix((npoints, npoints), dtype='double')
10
11     for e in tqdm(range(ne)):
12         # construir as matrizes do elemento
13         v1 = IEN[e, 0]           # vertice 1
14         v2 = IEN[e, 1]           # vertice 2
15         v3 = IEN[e, 2]           # vertice 3
16         v4 = IEN[e, 3]           # vertice 4
17
18         # importando matrizes do modulo matrizes
19         m = matriz3D(v1=v1, v2=v2, v3=v3, v4=v4, X=X, Y=Y, Z=Z)
20         melem = m.matrizm()
21         kelem = m.matrizk(k)
22
23         for ilocal in range(0, 4):
24             iglobal = IEN[e, ilocal]
25             for jlocal in range(0, 4):
26                 jglobal = IEN[e, jlocal]
27                 K[iglobal, jglobal] = K[iglobal, jglobal] + kelem[ilocal, jlocal]
28                 M[iglobal, jglobal] = M[iglobal, jglobal] + melem[ilocal, jlocal]

```

```

30     return K, M

32 def assembling2D(IENboundG, npoints, X, Y, Z, c):
33     # Matriz de massa do contorno (de Neumann e Robin)
34     MC = lil_matrix((npoints, npoints), dtype='double')
35     for e in tqdm(IENboundG):
36         # construir as matrizes do elemento
37         v1 = e[0]
38         v2 = e[1]
39         v3 = e[2]
40         v = [v1, v2, v3]

42         # importando matrizes do modulo matrizes
43         # obs: o codigo pode ser usado pois os elementos estao sempre na
44         # mesma
45         # coordenada Z. Caso nao estivesse, teria que fazer diferente.
46         sq = matriz2D(v1=v1, v2=v2, v3=v3, X=X, Y=Y)
47         melemg = sq.matrizm()

48         for ilocal in range(0, 3):
49             iglobal = v[ilocal]
50             for jlocal in range(0, 3):
51                 jglobal = v[jlocal]
52                 MC[iglobal, jglobal] = MC[iglobal, jglobal] + c * melemg[ilocal, jlocal]

54     return MC

```

A.5 Módulo das matrizes elementares

```

2 import numpy as np

4 class matriz3D():
5     def __init__(self, X, Y, Z, v1, v2, v3, v4):
6

```

```

8      self.b1=(Y[v2]-Y[v4])*(Z[v3]-Z[v4])-(Y[v3]-Y[v4])*(Z[v2]-Z[v4])
9      self.b2=(Y[v3]-Y[v4])*(Z[v1]-Z[v4])-(Y[v1]-Y[v4])*(Z[v3]-Z[v4])
10     self.b3=(Y[v1]-Y[v4])*(Z[v2]-Z[v4])-(Y[v2]-Y[v4])*(Z[v1]-Z[v4])
11     self.b4 = -(self.b1 + self.b2 + self.b3)

12     self.c1=(X[v3]-X[v4])*(Z[v2]-Z[v4])-(X[v2]-X[v4])*(Z[v3]-Z[v4])
13     self.c2=(X[v1]-X[v4])*(Z[v3]-Z[v4])-(X[v3]-X[v4])*(Z[v1]-Z[v4])
14     self.c3=(X[v2]-X[v4])*(Z[v1]-Z[v4])-(X[v1]-X[v4])*(Z[v2]-Z[v4])
15     self.c4 = -(self.c1 + self.c2 + self.c3)

16     self.d1=(X[v2]-X[v4])*(Y[v3]-Y[v4])-(X[v3]-X[v4])*(Y[v2]-Y[v4])
17     self.d2=(X[v3]-X[v4])*(Y[v1]-Y[v4])-(X[v1]-X[v4])*(Y[v3]-Y[v4])
18     self.d3=(X[v1]-X[v4])*(Y[v2]-Y[v4])-(X[v2]-X[v4])*(Y[v1]-Y[v4])
19     self.d4 = -(self.d1 + self.d2 + self.d3)

22
23     self.vol=(1/6)*np.linalg.det(np.array([[1,X[v1],Y[v1],Z[v1]],
24                                         [1,X[v2],Y[v2],Z[v2]],
25                                         [1,X[v3],Y[v3],Z[v3]],
26                                         [1,X[v4],Y[v4],Z[v4]]]))
27

28
29     def volCalc(self):
30         return self.vol

32
33     def matrizm(self):
34         melem = (self.vol/20.0)*np.array([[2.0, 1.0, 1.0, 1.0],
35                                         [1.0, 2.0, 1.0, 1.0],
36                                         [1.0, 1.0, 2.0, 1.0],
37                                         [1.0, 1.0, 1.0, 2.0]])
38
39         return melem

42
43     def matrizk(self, k):
44         # condutividade termica k
45         # Para material isotropico (kx=ky=kz=k)
46         B = (1/(6* self.vol))*np.array([[self.b1, self.b2, self.b3, self.b4],
47                                         [self.c1, self.c2, self.c3, self.c4],
48                                         [self.d1, self.d2, self.d3, self.d4],
49                                         []])

```

```

46     BT = np.transpose(B)

48     kelem = self.vol * k * np.dot(BT, B)
     return kelem

50

52 class matriz2D():
53
54     def __init__(self, X, Y, v1, v2, v3):
55         self.b1 = Y[v2] - Y[v3]
56         self.b2 = Y[v3] - Y[v1]
57         self.b3 = Y[v1] - Y[v2]
58         self.c1 = X[v3] - X[v2]
59         self.c2 = X[v1] - X[v3]
60         self.c3 = X[v2] - X[v1]
61
62         self.area=(1/2)*np.linalg.det(np.array([[1.0,X[v1],Y[v1]],
63                                             [1.0,X[v2],Y[v2]],
64                                             [1.0,X[v3],Y[v3]]]))
65
66     def areaCalc(self):
67         return self.area

68
69
70     def matrizm(self):
71
72         melem = (self.area/12.0)*np.array([[2.0, 1.0, 1.0],
73                                           [1.0, 2.0, 1.0],
74                                           [1.0, 1.0, 2.0]])
75
76         return melem

77
78
79     def matrizk(self):
80
81         B = (1/(2* self.area))*np.array([[self.b1, self.b2, self.b3],
82                                           [self.c1, self.c2, self.c3]])
83
84         BT = np.transpose(B)
85
86         kelem = self.area * np.dot(BT, B)
87
88         return kelem

```

A.6 Código principal da malha de paralelepípedo

```
2 import meshio
3
4 import numpy as np
5
6 from datetime import datetime
7 from tqdm import tqdm
8
9 from scipy.sparse.linalg import cg, spsolve
10 from scipy.sparse import lil_matrix, csr_matrix, issparse
11
12 from modulos.malha import cube
13 from modulos.montagem import assembling3D, assembling2D
14
15 from modulos.matrizes import matriz3D, matriz2D
16
17
18 startTime = datetime.now()
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
```

12 startTime = datetime.now()

14 , , ,

16 #####

16 # 1) Input – Parametros da simulacao

17 #####

18 , , ,

19 dt = 0.1 # Time step

20 nIter = 50 # Numero de iteracoes

21 theta = 1.0 # Metodo dif. finitas – imp = 1.0;

22 # – exp = 0.0;

23 # – cn = 0.5.

24 condCont = 1 # 1-dirichlet , 2-neumann e robin

25

26 T2 = 100

27 k = 49.8

28 rho = 7850

29 cv = 486

30 Tin = 35

31

32 , , ,

33 #####

34 # 2) Input – gerar malha no API do GMSH

35 #####

36 , , ,

```

38 # Inputs de um cubo
39 Lx = 0.05          # Tamanho da aresta no eixo x [m]
40 Ly = 0.05          # Tamanho da aresta no eixo y [m]
41 Lz = 0.1           # Tamanho da aresta no eixo z [m]
42 le = 0.025         # Tamanho medio do elemento [m]
43 filename = 'cube.msh'
44 cube(Lx, Ly, Lz, le, filename=filename)

46 # Salvando o tempo levado para construcao da malha
47 meshTime = datetime.now() - startTime
48

50 #####
51 # 2.1) Leitura das matrizes da malha
52 #####
53 msh = meshio.read(filename)
54 X = msh.points[:, 0]          # Coordenada x dos nohs
55 Y = msh.points[:, 1]          # Coordenada y dos nohs
56 Z = msh.points[:, 2]          # Coordenada z dos nohs
57 npoints = len(X)             # Numero de nos
58

60 #####
61 # 2.3) Matriz de conectividade (IEN)
62 #####
63 IENbound = []                 # Nohs do contorno
64 for elem in msh.cells:
65     if elem[0] == 'triangle':
66         IENbound.append(elem[1])
67     elif elem[0] == 'tetra':      # Elementos tetraedricos
68         IEN = elem[1]            # Matriz de conectivide IEN
69 ne = len(IEN)                 # Numero de elementos
70

72 #####
73 # 2.4) Nohs de contorno
74 #####
75 bound1 = []                  # Lista com os nohs da primeira superficie das ccs
76 bound2 = []                  # Lista com os nohs da segunda superficie das ccs

```

```

78 for elem in IENbound[0]:
    for noh in elem:
        bound1.append(noh)

82 for elem in IENbound[5]:
    for noh in elem:
        bound2.append(noh)

86 bound = bound1 + bound2

88 #####
90 # 3) Condicao de contorno
91 #####
92 print('\nPreparando simulacao:')
93 if condCont == 1: # Dirichlet
94     bval = np.zeros((npoints), dtype='float')
95     for b in tqdm(range(len(bval))):
96         if b in bound1:
97             bval[b] = T2
98         elif b in bound2:
99             bval[b] = T2
100    else:
101        qi = np.zeros((npoints), dtype='float')
102        Tc = np.zeros((npoints), dtype='float')
103        for b in tqdm(range(len(qi))):
104            if b in bound1:
105                qi[b] = 1.0      # Neumann
106                Tc[b] = -Tinf   # Robin
107            elif b in bound2:
108                qi[b] = -1.0    # Neumann
109                Tc[b] = Tinf    # Robin
110

112 #####
113 # 4) Assembling (matrizes K, M, MG e MC)
114 #####

```

```

116 print( '\nMain Assembling: ')
117 main = assembling3D (IEN, npoints , ne ,X=X, Y=Y, Z=Z , k=k )
118 K = main [0]
119 M = main [1]
120
121 #####
122 # 5) Montagem do sistema linear
123 #####
124 # change to csr: efficient arithmetic operations as CSR + CSR, CSR *
125 # CSR, etc.
126 M = M. tocsr ()
127 K = K. tocsr ()
128
129 # Lado esquerdo do sistema
130 A = rho*cv*M + (theta)*dt*(K) # + MC nas iteracoes
131 # # Salvar A em A2
132 # A2 = A.copy ()
133 # A2 = A2.todense ()
134
135 A = A. tolil ()
136
137 # Criacao das listas das variaveis
138 b = np.zeros ((npoints) , dtype='double ') # Lado direito da eq
139 T = Tin*np.ones ((npoints) , dtype='double ') # Temperaturas iniciais
140
141 #####
142 # 5.1) Imposicao das condicoes de contorno
143 #####
144 # Deixar a matriz H simetrica (passa os valores para o outro lado da
145 # equacao)
146 if condCont == 1:
147     for i in bound:
148         A[i , :] = 0.0 # zera a linha toda
149         # for j in range(npoints):
150         #     # passa os valores para o outro lado da equacao
151         #     b[j] = b[j] - A2[j , i]*bval [i]
152         # A[:, i] = 0.0 # zera a coluna toda

```

```

154     A[ i , i ] = 1.0                                # 1 na diagonal
155     T[ i ] = bval[ i ]                            # Temperatura de
156     contorno
157     # print( 'A eh esparsa?', issparse(H) )
158
159
160 # Salva resultados iniciais para visualizacao no Paraview
161 point_data = { 'temp' : T}
162 meshio.write_points_cells(f 'sol-0.vtk', msh.points,
163                           msh.cells, point_data=point_data, )
164 #####
165 # 6) Iteracoes no tempo
166 #####
167 print( '\nEquacao transiente: ')
168
169 if condCont == 1: # Dirichlet
170     for n in tqdm( range( nIter ) ):
171         # Lado direito da equacao
172         b = rho*cv*M. dot( T ) - dt*(1-theta)*K. dot( T ) # + M*dt*Q
173         for i in bound:
174             b[ i ] = bval[ i ] # mantem os nohs de contorno c/ a temp de
175             contorno
176
177             # Solucao do sistema linear (AT=b)
178             T = spsolve(A. tocsc(), b) # cg(H, f)[0] -> apenas zerando
179             coluna:
180
181             # simetrica positiva
182             definida.
183
184
185 # Salva resultados para visualizacao no Paraview
186 point_data = { 'temp' : T}
187 meshio.write_points_cells(f 'sol-{n+1}.vtk', msh.points,
188                           msh.cells, point_data=point_data, )
189
190
191 print( '\nSimulacao finalizada' )

```

```
188 #####
190 # 7) Tempos de simulacao
191 #####
192 totalTime = datetime.now() - startTime
193 print(f'\n\nTempos da simulacao: ')
194 print(f'Construcao da malha -> {meshTime} ')
195 print(f'Simulacao -> {totalTime - meshTime} ')
196 print(f'Tempo total -> {totalTime} ')
```