Politécnica UFRJ

SOLUÇÃO DA EQUAÇÃO DE LAPLACE TRIDIMENSIONAL PARA A TEMPERATURA APLICADA À UM BLOCO DE MOTOR ATRAVÉS DO MÉTODO DOS ELEMENTOS FINITOS

Mateus Mesquita Teixeira

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: Gustavo Rabello dos Anjos

Rio de Janeiro
Junho de 2023

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Escola Politécnica

Departamento de Engenharia Mecânica

SOLUÇÃO DA EQUAÇÃO DE LAPLACE TRIDIMENSIONAL PARA A TEMPERATURA APLICADA À UM BLOCO DE MOTOR ATRAVÉS DO MÉTODO DOS ELEMENTOS FINITOS

Mateus Mesquita Teixeira

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. Albino José Kalab Leiroz, Ph.D.

Prof. Fábio da Costa Figueiredo, D.Sc.

RIO DE JANEIRO, RJ BRASIL

JUNHO DE 2023

Mesquita Teixeira, Mateus

Solução da Equação de Laplace Tridimensional para a Temperatura Aplicada à um Bloco de Motor através do Método dos Elementos Finitos/ Mateus Mesquita Teixeira. – Rio de Janeiro: UFRJ/Escola Politécnica, 2023.

XII, 76 p.: il.; 29, 7cm.

Orientador: Gustavo Rabello dos Anjos

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Mecânica, 2023.

Referências Bibliográficas: p. 68 – 70.

1. Equação de Laplace. 2. Método dos elementos finitos. 3. Bloco de motor. I. Rabello dos Anjos, Gustavo. II. Universidade Federal do Rio de Janeiro, UFRJ, Curso de Engenharia Mecânica. III. Solução da Equação de Laplace Tridimensional para a Temperatura Aplicada à um Bloco de Motor através do Método dos Elementos Finitos.

"Quantas são as tuas obras, Senhor! Fizeste todas elas com sabedoria! A terra está cheia de seres que criaste."(Salmo 104:24)

Agradecimentos

Agradeço primeiramente à Deus, criador de todas as coisas, meu Senhor e salvador, que em sua misericórdia me concedeu cada oportunidade e a capacidade para aprender e poder concluir essa jornada.

Agradeço à minha esposa, Fernanda, pelo seu apoio incondicional, durante nosso namoro e agora no início do nosso casamento, sempre me encorajando e ficando ao meu lado nos momento mais difíceis.

Agradeço aos meus pais, Juscelino e Rosangela, e à minha famlia, por sempre ter me incentivado e dado apoio nos meus estudos, me dando condições para que eu pudesse atingir meus objetivos.

Agradeço à toda equipe do L'CADAME, que foi essencial na minha formação acadêmica.

Agradeço ao corpo docente do Departamento de Engenharia Mecânica pelo ensino e pela formação acadêmica que me foi proporcionada, e agradeço ao meu orientador, Gustavo Rabello dos Anjos pelo auxílio, apoio, paciência, incentivo e pelos ensinamentos que tornaram este trabalho possível.

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

SOLUÇÃO DA EQUAÇÃO DE LAPLACE TRIDIMENSIONAL PARA A TEMPERATURA APLICADA À UM BLOCO DE MOTOR ATRAVÉS DO MÉTODO DOS ELEMENTOS FINITOS

Mateus Mesquita Teixeira

Junho/2023

Orientador: Gustavo Rabello dos Anjos

Departamento: Engenharia Mecânica

Este trabalho propõe a solução do problema térmico em regime permanente utilizando-se do método de elementos finitos implementado utilizando linguagens de alto nível, como Python e Julia, e recursos acessíveis e de código aberto, como FreeCAD e Gmsh, solucionando a equação de Laplace 3D em um contexto que mostra a aplicabilidade do código em um ramo da indústria e da pesquisa, possibilitando a tomada de decisões de projeto por meio de ferramentas acessíveis e baixo custo. É feito uma análise comparativa entre diferentes fluidos usados para o resfriamento dos cilindros de um motor, mostrando como afetam a temperatura no material do bloco.

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

SOLUTION OF THE THREE-DIMENSIONAL LAPLACE'S EQUATION FOR TEMPERATURE APPLIED TO AN ENGINE BLOCK THROUGH THE FINITE ELEMENT METHOD

Mateus Mesquita Teixeira

June/2023

Advisor: Gustavo Rabello dos Anjos

Department: Mechanical Engineering

This work proposes the solution of the thermal problem in permanent regime using the finite element method implemented using high level languages, such as Python and Julia, and accessible and open-source resources, such as FreeCAD and Gmsh, solving 3D Laplace's equation in a context that shows the applicability of the code in a branch of industry and research, enabling design decisions through accessible and low cost tools. A comparative analysis is made between different fluids used for cooling the cylinders of an engine, showing how they affect the temperature in the block material.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Motores à combustão interna são a principal fonte de potência para meios de transporte rodoviários, e mesmo com a recente busca e preocupação pelas reduções de emissão dos gases de efeito estufa e a aprovação de legislações que buscam banir a venda de automóveis movidos à combustíveis fósseis como na União Europeia, a tecnologia tem a tendência de continuar dominando as frotas veiculares à curto prazo.

Os motores, que tem mais de um século de aprimoramento, atingiram um patamar no qual aparenta ser difícil e custoso o desenvolvimento de novas tecnologias que permitam a redução de emissões de poluentes e aumento de eficiência, além de que, devido às legislações recentes relativas à veículos movidos à combustíveis fósseis, muitas empresas não estão priorizando o desenvolvimento de novos motores do zero, apenas modificando projetos já existentes.

Ao longo da história, para o desenvolvimento e pesquisa para novos projetos de motores e as modificações pontuais mais recentes, envolvem estudos acerca dos motores de combustão interna, englobando diversos aspectos da engenharia mecânica. Um deles é a análise da transferência de calor do motor, que surge devido à necessidade de se quantificar as temperaturas e o fluxo de calor presentes no sistema, que permite a determinação mais completa das condições de operação do equipamento, que afeta diretamente outras partes do projeto, como condições de lubrificação dos pistões, formação de poluentes, integridade estrutural do bloco, o quanto da energia gerada pela combustão deixa de ser convertida em potência, para citar alguns exemplos.

Foram desenvolvidos diversos modelos para tal estudo, muitos deles empíricos ou semi-empíricos, porém a partir da década de 90(FONSECA *et al.* (2020)) as simulações computacionais se tornaram uma ferramenta importante no campo dos automóveis, ajudando na redução de custos e resolução de modelos diversos e mais complexos para o estudo do desempenho e integridade do motor e os processos envolvidos de forma geral.

Nesse contexto há a possibilidade realizar uma análise térmica global ou em partes específicas da estrutura do motor, ou do fluido de trabalho, em função do tempo, levando em conta a variação geométrica da câmara de combustão, em combinação com modelos empíricos, com mais ou menos hipóteses simplificadoras sendo adotadas. Isso permite que as grandezas físicas desse problema sejam determinadas com mais precisão e em um maior nível de detalhe que modelos puramente empíricos ou analíticos muito simples.

1.1 Objetivo

O objetivo deste trabalho é realizar uma análise térmica em um bloco de motor genérico, num modelo físico simplificado, a partir do método de elementos finitos(FEM), demonstrando a aplicabilidade do método em problemas de engenharia e a acessibilidade de sua implementação computacional a partir de linguagens de alto nível, estudando o caso da transferência de calor dos gases da combustão, para o material do bloco, e para o refrigerante, num regime de operação estacionário, comparando-se dois tipos de fluido refrigerantes e observando sua influência nas temperaturas do motor.

1.2 Metodologia

Para a implementação do método de elementos finitos, é necessário antes realizar a modelagem matemática do problema, deduzindo suas equações diferenciais governantes e condições de contorno a partir de princípios físicos e adoção de hipóteses simplificadoras, estimadas a partir da literatura disponível sobre a transferência de calor em motores de combustão interna.

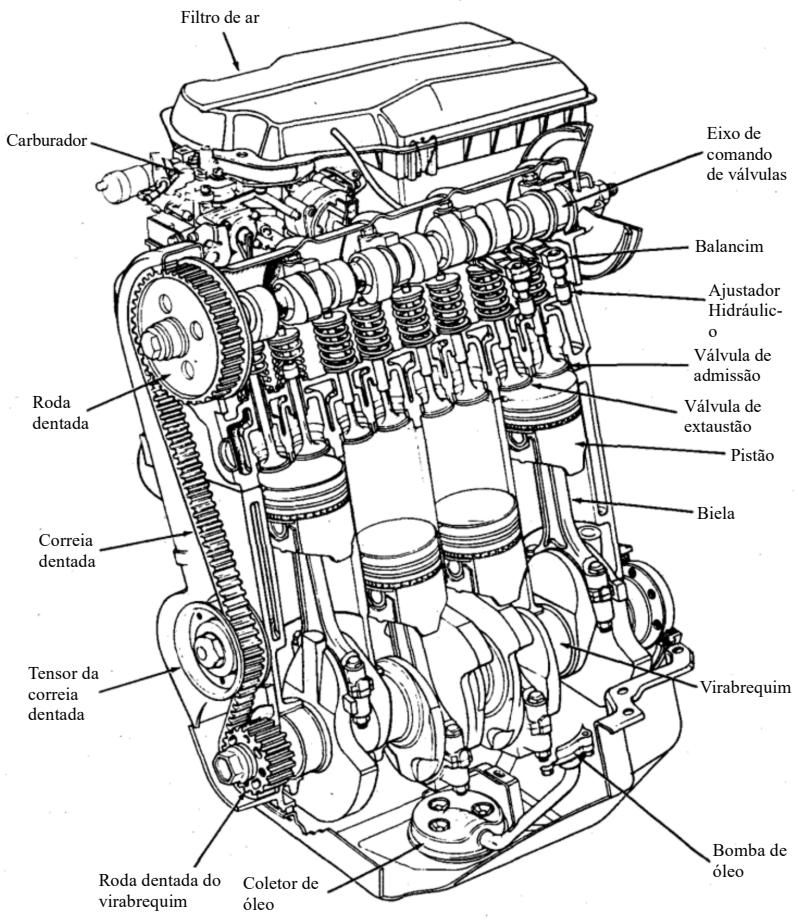


Figura 1.1: Exemplo de um motor a combustão interna. Adaptado de HEYWOOD (1988).

Com o problema equacionado, é possível realizar a aplicação do método de elementos finitos para a discretização das equações e do domínio físico, para possibilitar sua implementação computacional. A discretização do domínio será realizada a partir da geração de uma malha utilizando o *Gmsh* sobre um modelo CAD criado no software *FreeCAD*.

A implementação computacional das equações discretizadas, por sua vez, será realizada em duas linguagens: Python, para a leitura de malhas e gravação de resultados, e Julia, para a montagem das matrizes e resolução do sistema linear resultantes da aplicação método de elementos finitos.

1.3 Organização do texto

1. No 2 será mostrada toda a fundamentação teórica que permitirá o desenvolvimento do trabalho, desde o fenômeno da condução do calor, passando pela equação governante do problema, apresentando a teoria básica do método de elementos finitos e da transferência de calor em um motor.

2. No 3 será apresentado, de forma específica para o problema proposto, como serão obtidos os parâmetros necessários para solucionar a equação governante do problema, a implementação tridimensional do método de elementos finitos e como o problema será implementado computacionalmente.

3. No 4 será feita a validação e verificação do código, mostrando como ele de fato é capaz de resolver o problema matemático proposto de forma confiável ao realizar a comparação com uma solução analítica, e como o código também é confiável em um aspecto numérico, demonstrando a convergência de malha para a implementação computacional proposta.

4. No 5 serão mostrados as simulações que foram realizadas e os seus resultados obtidos, comparando as soluções para diferentes fluidos refrigerantes e salientando alguns aspectos da solução.

5. No 6 será apresentada a conclusão do trabalho, e sugestões para trabalhos subsequentes na mesma linha.

Capítulo 2

Fundamentação Teórica

2.1 Transferência de Calor

O calor, principal objeto de estudo em uma análise térmica, é, de acordo com ÇENGEL e GASHAR (2015), a forma de energia que pode ser transferida de um sistema para outro devido à diferença de temperatura, e a ciência que lida com as taxas dessa transferência de energia. De forma que, através desse estudo detalhado, não apenas avalia-se a quantidade de energia transferida na forma de calor em determinados processos, mas também é possível determinar o tempo necessário para tal processo. O calor pode ser transferido através de três mecanismos: condução, convecção e radiação.

2.1.1 Condução

A condução é um mecanismo de transferência de calor que ocorre devido à um fenômeno de difusão molecular, ou seja, a energia é transferida de uma partícula para outra através de colisões entre as mesmas devido à sua vibração aleatória. A lei de Fourier afirma que o fluxo de calor é proporcional ao gradiente de temperatura, ou seja:

$$\mathbf{q} = -k\nabla T \tag{2.1}$$

onde $\mathbf{q}$ é o vetor de fluxo de calor e k é a condutividade térmica do material.

2.1.2 Convecção

Na convecção, a transferência de energia se dá entre um sólido e um fluido em movimento, de forma que primeiro, através da condução, o calor é transferido para a camada de fluido diretamente em contato com a superfície do sólido, e então a energia é transportada pelo próprio movimento do fluido.

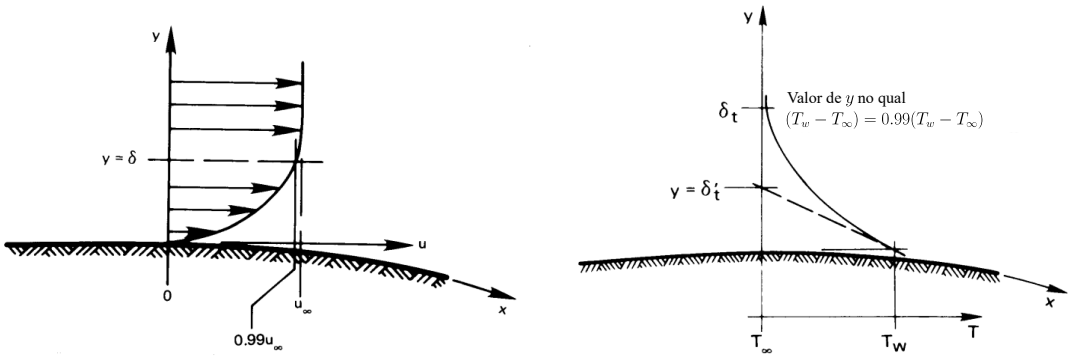
Existem duas formas em que a convecção pode ocorrer: através do escoamento forçado do fluido sobre o sólido em questão, sendo chamado de convecção forçada, e a convecção livre, que ocorre quando no interior do fluido há uma diferença de temperatura, que promove um gradiente de densidade, que sob efeito campo gravitacional provoca um desequilíbrio de forças, gerando movimento no fluido. Por mais complexo que este fenômeno seja, é observado que o fluxo de calor é proporcional a uma diferença de temperatura

$$q = h(T_w - T_\infty) \tag{2.2}$$

onde q é o fluxo de calor que sai da superfície do corpo, h é o coeficiente de transferência de calor por convecção, T_w é a temperatura na superfície e T_∞ é a temperatura no fluido numa região suficientemente afastada do sólido em questão. O coeficiente de transferência de calor por convecção, diferentemente da condutividade térmica, não é uma propriedade física, mas é um parâmetro definido experimentalmente que depende dos fatores de influenciam nessa forma de transferência de energia, como a geometria do corpo, velocidade do escoamento e sua natureza, e as propriedades do fluido.

Quando o fluido escoa próximo a uma superfície, há a formação da chamada camada limite, como estudado por Ludwig Prandtl no início do século XX. Esta é uma região onde os efeitos viscosos são relevantes, havendo a presença de um grande gradiente de velocidade devido à condição de não escorregamento, que é a origem de fenômenos como o arrasto. É também devido à condição de não escorregamento que a tranmissão de energia entre o fluido e a superfície do sólido se dá apenas por condução, de forma que, conhecendo-se o campo de temperaturas, pode-se determinar h.

$$-k_f \left.\frac{\partial T}{\partial y}\right|_{y=0} \equiv h(T_w - T_\infty)$$



(a) Camada limite.

(b) Camada limite térmica.

Figura 2.1: Adaptado de LIENHARD e LIENHARD (2020).

rearranjando a equação, tem-se:

$$\left.\frac{\partial\left(\frac{T_w-T}{T_w-T_\infty}\right)}{\partial(y/L)}\right|_{y/L=0} = \frac{hL}{k_f} \equiv Nu_L \tag{2.3}$$

onde Nu_L é o número de Nusselt.

2.1.3 Radiação

O terceiro modo de transmissão de energia é a radiação, que é a energia emitida por um corpo na forma de ondas eletromagnéticas e não precisa da presença de um meio, ou seja, ela pode ocorrer no vácuo, e neste caso não sofre atenuação. Neste caso, há a radiação térmica, que é a energia emitida por um corpo devido à sua temperatura. Em sólidos normalmente a radiação térmica é considerada um fenômeno de superfície, ou seja, apenas a parte mais externa destes corpos é relevante na emissão e absorção de radiação. A lei de Stefan-Boltzmann fornece o fluxo de de energia máximo emitido por um corpo, chamado aqui de poder emissivo de um corpo negro, E_b, objeto teórico que emite a maior taxa possível de energia radiante.

$$E_b(T) = \sigma T_s^4 \tag{2.4}$$

onde T_s é a temperatura da superfície do corpo e $\sigma = 5,670 \times 10^{-8}$ W/m^2 é a constante de Stefan-Boltzamann. Corpos reais emitem uma fração do poder emissivo de um corpo negro, ou seja,

$$E(T) = \epsilon\sigma T_s^4 \tag{2.5}$$

onde $0 \leq \epsilon \leq 1$ é a emissividade do corpo real.

A troca de calor radiante entre uma superfície e sua vizinhança pode ser calculada como

$$Q_{rad} = \epsilon \sigma A_s \left(T_s^4 - T_{viz}^4\right) \tag{2.6}$$

Para simplificar os cálculos, é possível definir um coeficiente de transferência de calor por radiação, $h_{rad} = 4T_s^3 \epsilon \sigma$, de forma que a transferência de calor radiante torna-se

$$Q_{rad} = h_{rad} A_s \left(T_s - T_{viz}\right) \tag{2.7}$$

2.1.4 Equação da Condução de Calor

A equação da condução de calor pode ser deduzida a partir da primeira lei da termodinâmica, ou balanço de energia, aplicado sobre um corpo Ω.

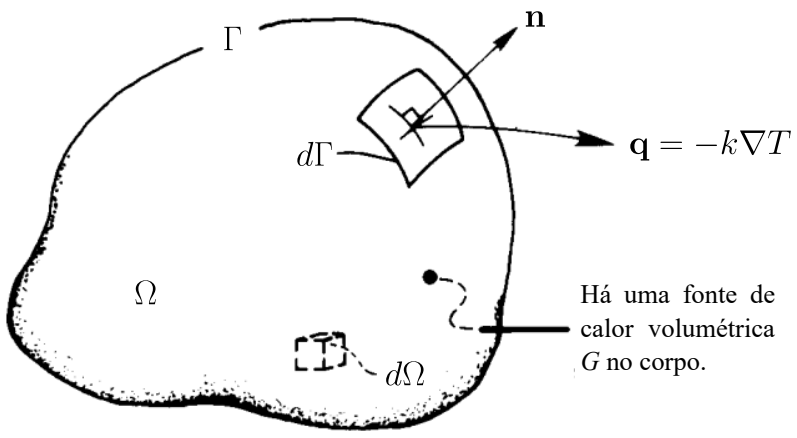


Figura 2.2: Imagem adaptada de LIENHARD e LIENHARD (2020).

$$\dot{Q} = \frac{dU}{dt} \tag{2.8}$$

Nele não há trabalho sendo realizado pelo sistema. O termo $\dot{Q}$ é o calor sendo transferido para o sistema, que pode ser fornecido na forma do fluxo que calor $\mathbf{q} = -k\nabla T$ que atravessa a superfície Γ, que possui normal $\mathbf{n}$, do corpo e também por uma geração de calor volumétrica G, de forma que

$$\dot{Q} = -\int_\Gamma (-k\nabla T) \cdot (\mathbf{n} dS) + \int_\Omega G d\Omega \tag{2.9}$$

O incremento de energia armazenada no corpo pode ser dada por

$$\frac{dU}{dt} = \int_{\Omega} \rho c_v \frac{\partial T}{\partial t} d\Omega \tag{2.10}$$

onde $\rho c_v \frac{\partial T}{\partial t} d\Omega$ é a taxa de variação da energia interna do sistema em um processo à volume constante. Assim, a equação (2.8) torna-se

$$\int_{\Gamma} (k\nabla T) \cdot (\mathbf{n} d\Gamma) + \int_{\Omega} G d\Omega = \int_{\Omega} \rho c_v \frac{\partial T}{\partial t} d\Omega \tag{2.11}$$

Pelo teorema de Gauss a integral do fluxo de calor que atravessa a superfície Γ pode ser reescrita como a integral do divergente do fluxo de calor no corpo Ω.

$$\int_{\Omega} \nabla \cdot (k\nabla T)\, d\Omega + \int_{\Omega} G d\Omega = \int_{\Omega} \rho c_v \frac{\partial T}{\partial t} d\Omega \tag{2.12}$$

e simplificando

$$\int_{\Omega} \left[\nabla \cdot (k\nabla T) + G - \rho c_v \frac{\partial T}{\partial t} \right] d\Omega = 0 \tag{2.13}$$

Como o corpo R é arbitrário, essa expressão vale para qualquer corpo, de forma que pode-se eliminar a integral e, por fim, obtém-se a equação da condução de calor, uma equação diferencial parcial em $T = T(\mathbf{r}, t)$.

$$\nabla \cdot (k\nabla T) + G - \rho c_v \frac{\partial T}{\partial t} = 0 \tag{2.14}$$

Caso não haja variação temporal no sistema, ou seja, quando ele está em regime permanente, e também não haja variação espacial da condutividade térmica, então a equação acima pode ser simplificada para

$$k\nabla^2 T + G = 0 \tag{2.15}$$

que é chamada equação de Poisson. Caso não haja um termo fonte, ela se reduz à equação de Laplace

$$\nabla^2 T = 0 \tag{2.16}$$

Há três variedades de condições de contorno linear para este problema: condições do primeiro tipo(*Dirichlet*), onde prescreve-se a temperatura no contorno, segundo tipo(*Neumann*), onde prescreve-se o fluxo de calor, e terceiro tipo(*Robin*), onde há convecção na fronteira. Essas condições de contorno podem ser homogêneas ou não, e estão apresentadas a seguir:

$$T = \overline{T} \text{ em } \Gamma_T \tag{2.17}$$

$$k\nabla T \cdot \hat{\mathbf{n}} = \overline{q} \text{ ou } k\frac{\partial T}{\partial n} = \overline{q} \text{ em } \Gamma_q \tag{2.18}$$

$$k\nabla T \cdot \hat{\mathbf{n}} = h(T_\infty - T) \text{ ou } k\frac{\partial T}{\partial n} + hT = hT_\infty \text{ em } \Gamma_h \tag{2.19}$$

onde a normal $\hat{\mathbf{n}}$ aponta para fora da superfície.

2.2 Método dos Elementos Finitos

Os fenômenos que ocorrem na natureza, sejam eles de origem biológica, mecânica ou geológica, podem ser descritos como modelos matemáticos a partir das leis da física, de forma que é possível obter equações diferenciais, integrais ou algébricas que relacionam as quantidades de interesse do problema em questão. Na maioria das vezes, a criação de um modelo e a dedução das chamadas equações governantes de um problema não é uma tarefa excessivamente difícil, encontrar uma solução analítica para estas equações pode ser quase impossível, demandando que fossem desenvolvidos métodos alternativos para encontrar uma solução.

Uma dessas alternativas é o Método dos Elementos Finitos, um método numérico para solução de equações diferenciais, cuja ideia básica é dividir o domínio em *elementos finitos*, normalmente chamados apenas de *elementos*, conectados por *nós* e obter uma solução aproximada(FISH e BELYTSCHKO (2007)).

O Método dos Elementos Finitos tem sua origem nos métodos variacionais, que foram propostos também como uma forma aproximada para solucionar problemas de engenharia. De forma geral, um método variacional consiste em escrever a equação diferencial numa forma integral-ponderada(formulação fraca) e então a solução aproximada sobre o domínio é assumida como sendo uma combinação linear das funções de aproximação e de coeficientes a serem determinados, que devem ser avaliados de forma que a formulação fraca seja equivalente à equação diferencial original seja satisfeita(REDDY (1993)).

Baseado neste princípio, o MEF consiste em formular sistematicamente funções de aproximação para os ditos elementos do domínio. REDDY (1993) afima que o método dos elementos finitos tem vantagens sobre os tradicionais métodos variacionais ou métodos de diferenças finitas por três razões: um domínio complexo é representado como o conjunto de subdomínios simples, as funções de aproximação em cada elemento são deduzidas a partir da ideia básica de que qualquer função

contínua pode ser representada por uma combinação linear de polinômios algébricos, e, por último, as relações algébricas entre os coeficientes à determinar são obtidos satisfazendo as equações governantes, normalmente em sua formulação fraca, sobre cada elementos. De forma que o método dos elementos finitos pode ser entendido como uma aplicação de um método variacional ou método do resíduo ponderado no subdomínio de um elemento. Nesse caso, polinômios são escolhidos para as funções de aproximação e os parâmetros indeterminados representam os valores da solução nos nós sobre na fronteira e no interior do domínio.

Historicamente, características chave do método de elementos finitos podem ser observados em trabalhos de Hrenikoff (1941) e Courant (1943), entretanto uma apresentação formal do método de elementos finitos a Argyris e Kelsey (1960), e Turner, Clough, Martin e Topp (1956), sendo que o termo "elementos finitos"foi usado primeiramente por Clough em 1960(REDDY (1993)). Após seu desenvolvimento inicial, no contexto da indústria aeroespacial dos anos 50, estes trabalhos geraram esforços intensos na academia, principalmente na universidade de Berkeley, que foi um centro de pesquisa do método por anos(FISH e BELYTSCHKO (2007)). Na década de 60, o método chamou atenção de matemáticos, que demonstraram que para problemas lineares, o método de elementos finitos converge para a solução correta das equações diferenciais parciais.

A partir da década de 60, com o desenvolvimento dos primeiros softwares de elementos finitos, combinado com o desenvolvimento de computadores cada vez mais baratos e potentes o método de elementos finitos tornou-se cada vez mais popular, primeiramente na área de análise estrutural, e depois aplicados a problemas de diferentes naturezas físicas, no contexto de fluidodinâmica computacional por exemplo, resultando num método numérico consolidado na indústria e na pesquisa atualmente.

A análise de um problema por meio do método de elementos finitos consiste nos seguintes passos, de acordo com REDDY (1993):

1. Discretização do domínio em elementos finitos previamente escolhidos, gerando uma malha.(Esta etapa pode ser realizada após a formulação de elementos finitos da equação.)

2. Dedução das equações elementares para os elementos típicos da malha.

3. A montagem, ou *assembly*, das equações do problema completo.

4. Imposição das condições de contorno.

5. Solução do sistema linear montado.

6. Pós-processamento dos resultados.

Neste capítulo, será enfatizado a dedução das equações elementares a partir do método do resíduo ponderado, um dos mais importantes na aplicação do método de elementos finitos, reservando as outras etapas da análise para o capítulo dedicado à metodologia, já que são procedimentos de ordem mais prática e específica para cada problema.

2.2.1 Elementos e Funções de Forma

Antes de deduzir as equações gerais para um elemento, é importante apresentar de forma mais profunda a discretização do domínio e da equação governante. A ideia principal no Método de Elementos Finitos é representar as variáveis do problema de uma maneira particionada sobre a região discretizada. Ao dividir a região do problema em elementos e aproximar a solução em cada um deles por uma função conhecida e adequada, é estabelecida uma relação entre a equação diferencial e os elementos, e as funções utilizadas para representar a natureza da solução em cada elemento são chamadas de funções de interpolação ou funções de forma, sendo utilizadas para determinar os valores da variável do problema dentro de um elemento interpolando os valores nodais. Normalmente são utilizadas funções polinomiais, devido à facilidade com que são diferenciadas ou integradas, e porque a precisão do resultado pode ser facilmente aprimorada aumentando a ordem do polinômio(NITHIARASU *et al.* (2016)). O tipo de elemento e de função de forma a ser escolhido depende do problema a ser solucionado, há elementos unidimensionais, bidimensionais ou tridimensionais, tendo formas triangulares, de quadriláteros, tetraédricas ou hexaédricas, e as funções de forma podem ser lineares, quadráticas ou mesmo cúbicas, e esta escolha resulta diretamente no número de nós por elemento.

As funções e forma, normalmente denominadas, $N_i^{(e)}$, ou seja, a função de forma do nó i e do elemento e, precisam obedecer os seguintes requisitos:

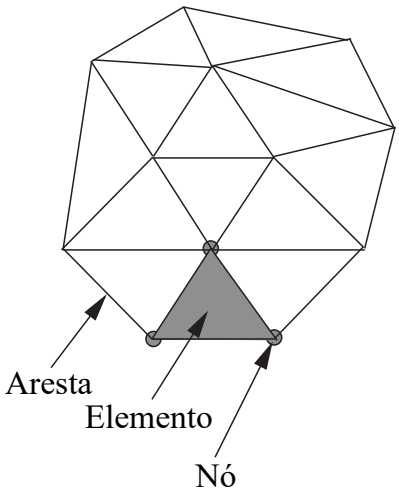


Figura 2.3: Uma malha típica de elementos finito, com nós, arestas e elementos. Adaptada de NITHIARASU *et al.* (2016).

- Valem 1 no nó i e se anulam nos outros nós do elemento e.
- São contínuas dentro do domínio do elemento.

de forma que a variável do problema, neste caso a temperatura T, possa ser aproximada no domínio do elemento por

$$T^{(e)} = \sum_i N_i^{(e)} T_i^{(e)} \tag{2.20}$$

onde $T_i^{(e)}$ é a temperatura do nó i dentro do elemento.

2.2.2 Método do Resíduo Ponderado - Galerkin

Um dos principais métodos de formulação para as equações dos elementos finitos é o Método do Resíduo Ponderado, ele proporciona um procedimento de para a aproximação muito poderoso, aplicável a uma variedade de problemas, de forma que não é necessário procurar uma formulação variacional para a aplicação do método de elementos finitos(NITHIARASU *et al.* (2016)).

Para dar inicio à aplicação do método, como descrito em NITHIARASU *et al.* (2016) e HUEBNER *et al.* (2001), assume-se que a temperatura no domínio em questão pode ser aproximada como

$$T \approx \widetilde{T} = \sum_{i=1}^{n} N_i T_i = \mathbf{NT} \tag{2.21}$$

onde $\mathbf{N}$ é o vetor linha de funções de forma e $\mathbf{T}$ é o vetor coluna de temperaturas nodais.

Será utilizada a equação da condução em regime permanente (2.16) para a dedução de uma equação elementar genérica. Substituindo a temperatura pela aproximação proposta, a tem-se que

$$k\nabla^2\widetilde{T} + G = R \tag{2.22}$$

onde R é o resíduo ou o erro da aproximação proposta. O método requer que os parâmetros desconhecidos a_i sejam satisfeitos de forma que R seja pequeno no domínio do problema. Isso é alcançado calculando uma média ponderada do resíduo e determinando que ela seja nula sobre o domínio, e para tal escolhe-se funções de ponderação w_i e insiste-se que

$$\int_\Omega \mathbf{w} R d\Omega = \int_\Omega \mathbf{w} \left[k\nabla^2\widetilde{T} + G\right] d\Omega = 0 \tag{2.23}$$

onde $\mathbf{w} = \begin{bmatrix} w_1 & w_2 & ... & w_n \end{bmatrix}^{\mathbf{T}}$ é o vetor coluna das n funções peso.

Especificando-se as funções peso, então a equação anterior, também conhecida como a formulação fraca do problema, passa a representar um conjunto de equações algébricas. Existem algumas escolhas comuns para as funções de peso, porém uma das mais comuns é definir que

$$\mathbf{w} = \mathbf{N^T} = \begin{bmatrix} N_1 & N_2 & ... & N_n \end{bmatrix}^{\mathbf{T}}$$

ou seja, as mesmas funções escolhidas para aproximar T, com essa escolha consistindo no chamado Método de Galerkin, e assim, tem-se que

$$\int_\Omega \mathbf{N^T} \left[k\nabla^2\widetilde{T} + G\right] d\Omega = 0 \tag{2.24}$$

A equação acima também é válida para partições do domínio, neste caso, para os elementos, escrevendo

$$\int_{\Omega^{(e)}} \mathbf{N^{(e)}}^{\mathbf{T}} \left[k\nabla^2 T^{(e)} + G\right] d\Omega = 0 \tag{2.25}$$

onde $\mathbf{N}^{(e)} = \begin{bmatrix} N_1 & N_2 & ... & N_{n_{el}} \end{bmatrix}$ e n_{el} é o número de nós por elemento. Fazendo a integração por partes:

$$\int_{\Omega^{(e)}} \mathbf{N^{(e)}}^{\mathbf{T}} k\nabla^2 T^{(e)} d\Omega = \int_{\Gamma^{(e)}} \mathbf{N^{(e)}}^{\mathbf{T}} k\nabla T^{(e)} \cdot \hat{\mathbf{n}} d\Gamma - \int_{\Omega^{(e)}} \nabla \mathbf{N^{(e)}}^{\mathbf{T}} \cdot k\nabla T^{(e)} d\Omega$$

diminuindo a ordem da equação diferencial e incluindo os efeitos da fronteira na equação, respectivos à (2.18) e (2.19). Assim, (2.25) torna-se

$$-\int_{\Omega^{(e)}} \nabla \mathbf{N}^{(e)^{\mathbf{T}}} \cdot k \nabla T^{(e)} d\Omega + \int_{\Omega^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} G d\Omega + \int_{\Gamma_q^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} \overline{q} d\Gamma - \int_{\Gamma_h^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h T^{(e)} d\Gamma + \int_{\Gamma_h^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h T_\infty d\Gamma = 0 \quad (2.26)$$

Substituindo a temperatura no elemento pela aproximação proposta, $T^{(e)} = \mathbf{N}^{(e)}\mathbf{T}^{(e)}$, onde $\mathbf{T}^{(e)}$ é o vetor coluna de temperaturas nodais do elemento, tem-se por fim

$$-\int_{\Omega^{(e)}} \nabla \mathbf{N}^{(e)^{\mathbf{T}}} k \nabla \mathbf{N}^{(e)} d\Omega \mathbf{T}^{(e)} + \int_{\Omega^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} G d\Omega + \int_{\Gamma_q^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} \overline{q} d\Gamma - \int_{\Gamma_h^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h \mathbf{N}^{(e)} d\Gamma \mathbf{T}^{(e)} + \int_{\Gamma_h^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h T_\infty d\Gamma = 0 \quad (2.27)$$

que são as equações elementares para o problema de condução de calor em regime permanente. Usualmente é definida a matriz de gradiente $\mathbf{B}^{(e)} = \nabla \mathbf{N}^{(e)}$. Escrevendo na forma de um sistema de equações lineares:

$$\mathbf{K}^{(e)}\mathbf{T}^{(e)} = \mathbf{f}^{(e)} \quad (2.28)$$

onde $\mathbf{K}^{(e)}$ é a chamada matriz de rigidez ou condutividade térmica do elemento, $\mathbf{d}^{(e)}$ são as temperaturas nodais do elemento, e $\mathbf{f}^{(e)}$ é o vetor de carregamento elementar, e eles são dados por:

$$\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} \mathbf{B}^{(e)^{\mathbf{T}}} k \mathbf{B}^{(e)} d\Omega + \int_{\Gamma_h^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h \mathbf{N}^{(e)} d\Gamma \quad (2.29)$$

$$\mathbf{f}^{(e)} = \int_{\Omega^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} G d\Omega + \int_{\Gamma_q^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} \overline{q} d\Gamma + \int_{\Gamma_h^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h T_\infty d\Gamma \quad (2.30)$$

A partir da escolha das funções de forma, é possível a obtenção das equações elementares para o problema. Na literatura, há para tipos de elementos e funções de forma mais comuns maneiras já descritas de como calcular facilmente as matrizes das equações elementares, através de códigos computacionais. Neste trabalho, esta escolha será descrita posteriormente, no Capítulo 3, apresentando o elemento e a função de forma escolhida, assim como são calculadas as matrizes elementares.

Após a determinação das matrizes elementares, a partir do processo de *Assembling* ou Montagem, que será detalhado no Capítulo 3, é possível reunir as equações elementares no sistema linear global, que envolve todos os nós do domínio:

$$\mathbf{K}\mathbf{T} = \mathbf{f} \quad (2.31)$$

no qual a solução é o vetor de temperaturas nodais **T**.

2.3 Motores de Combustão Interna

Os motores de combustão interna tem o propósito de produzir energia mecânica a partir da energia química contida no combustível. Em motores de combustão interna, essa energia é gerada através da queima ou oxidação do combustível *dentro* do motor, onde os fluidos de trabalho são a mistura ar-combustível e os produtos da queima após a combustão, onde também o trabalho é transferido diretamente dos fluidos para os componentes mecânicos do motor(HEYWOOD (1988)).

O motor de combustão interna de ignição por centelha como é conhecido foi desenvolvido primeiramente por Nicolaus A. Otto no século XIX, e seu ciclo de funcionamento proposto consiste de 4 etapas: admissão, combustão, expansão, onde o trabalho é transferido para o virabrequim, e exaustão. Ao mesmo tempo a exploração comercial de poços de petróleo permitiu que combustíveis líquidos fossem utilizados e este fator contribuiu para uma aceleração no desenvolvimento de motores a combustão interna, diversificando também os tipos disponíveis, para além daquele proposto por Otto. Imediatamente após sua invenção, foram desenvolvidos motores de dois tempos e variações do ciclo Otto, como proposto por Atkinson(STONE (1999)). Já ao longo do século XX o desenvolvimento dos motores foi afetado pelos estudos e utilização de diferentes combustíveis, pelas crises do petróleo e início de preocupações ambientais com relação às formas de poluição geradas pelo motores, e mesmo depois de mais de 100 anos de desenvolvimento, ainda há aprimoramentos significativos em eficiência, potência e controle de emissões, seja por meio de novos materiais de fabricação, e tipos alternativos de ciclos(HEYWOOD (1988)).

O estudo da transferência de calor em motores a combustão interna possui importância significativa para o estudo da performance, eficiência e emissões de um motor, e além disso, para o próprio design de um sistema de refrigeração e entender como se dá o fluxo de calor através das suas partes constituintes. Dessa forma é necessária análise térmica para que o motor seja projetado adequadamente, de forma a ter uma performance otimizada e integridade estrutural(HEYWOOD (1988); TAYLOR (1985)).

Num motor de combustão interna, o processo de transmissão de calor é complexo, desde a combustão que transfere calor para a parede dos cilindros, à geometria envolvida, fenômenos de condução e radiação que ocorrem de forma simultânea, fluxos de calor e temperaturas de magnitude muito diferentes e que variam significativamente com o tempo em diferentes partes de todo o sistema que compõe um motor. Neste caso, há a necessidade de criar modelos simplificados e de dividir o motor em diversos subsistemas de interesse aos quais cabe a análise térmica, vale destacar o sistema de admissão, de exaustão, a câmara de combustão e o sistema de refrigeração, e através do estudo da transferência de calor por meio dos modelos disponíveis, é possível conhecer como se dá este fenômeno entre esses subsistemas e também com os componentes estruturais de um motor(TAYLOR (1985)).

No caso específico da transferência de calor no bloco de um motor há o fluxo de energia dos gases da combustão para as paredes do cilindro, e das paredes do cilindro para o fluido refrigerante. Há transferência de calor por radiação e convecção do lado dos gases, por condução ao longo da estrutura do bloco, e por convecção para o fluido de resfriamento. Em cada ciclo de operação há grandes variações da temperatura e pressão dos gases, do fluxo de calor, da geometria do cilindro, além do fluxo de calor não ser uniforme ao longo das paredes do cilindro. Por vezes, é possível adotar uma abordagem de regime "quasi-estacionário", entretanto dependendo do tipo e objetivo da análise a ser feita, o engenheiro pode modelar o problema como sendo em regime estacionário ou não, com fluxo uniforme na parede ou não(HEYWOOD (1988)).

Para um fluxo de calor unidimensional e estacionário, representada no gráfico anterior, tem-se o seguinte:

Lado da câmara de combustão: $\dot{q} = \dot{q}_{CV} + \dot{q}_R = h_{c,g}\left(\overline{T}_g - T_{w,g}\right) + \sigma\epsilon\left(\overline{T}_g^4 - T_{w,g}^4\right)$

normalmente o termo por radiação é desprezível em motores por ignição por centelha.

$$\text{Parede: } \dot{q} = \dot{q}_{CN} = \frac{k(T_{w,g} - T_{w,c})}{t_w}$$

onde t_w é a espessura da parede.

$$\text{Lado do refrigerante: } \dot{q} = \dot{q}_{CV} = h_{c,c}\left(T_{w,c} - \overline{T}_c\right)$$

Se os coeficientes da transferência de calor por convecção são conhecidos, as temperaturas podem ser relacionadas entre si(HEYWOOD (1988)).

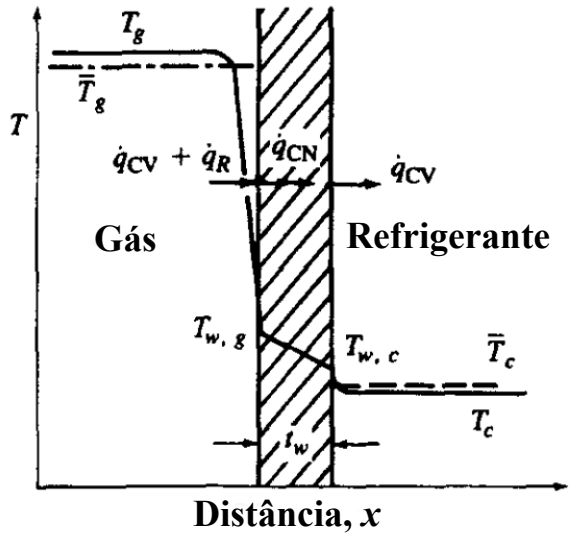


Figura 2.4: Processo de transferência de calor geral em um motor à combustão interna. Adaptado de HEYWOOD (1988).

Desprezando-se a radiação, o que é aceitável em motores de ignição por centelha, e seguindo o molde dado pelas três equações anteriores, a partir da análise dimensional, muitas fórmulas para o cálculo do coeficiente de transmissão de calor no lado da câmara de combustão $h_{c,g}$(HEYWOOD (1988); FONSECA *et al.* (2020); FERGUSON e KIRKPATRICK (2016))

$$Nu = CRe^a Pr^b \tag{2.32}$$

As correlações propostas podem utilizadas para predizer o fluxo de calor médio no tempo para as paredes da câmara de combustão, para o fluxo instantâneo médio para as paredes ou mesmo para o fluxo instantâneo local, dependendo da análise a ser feita(HEYWOOD (1988)).

Capítulo 3

Metodologia

3.1 Modelo Físico

Neste trabalho, como foi dito na introdução, será feita a análise térmica de um bloco de um motor. Para este propósito, é necessário realizar, em primeiro lugar, a modelagem física do problema. Como o objetivo é fazer uma análise da transferência de calor de forma geral para o refrigerante durante um regime de operação estacionário, como um carro em uma rodovia à velocidade constante, é necessário definir algumas hipóteses para representar o problema matematicamente:

1. O material do bloco é isotrópico e homogêneo e suas propriedades não variam com a temperatura.

2. O processo de transferência de calor é realizado em regime permanente, desprezando-se as variações do fluxo de calor na câmara de combustão ao longo de um ciclo, já que sua penetração no metal é em um nível muito superficial, e o interior do material permanece aproximadamente sem oscilações(FERGUSON e KIRKPATRICK (2016)). Além disso ele é homogêneo sobre toda a parede interna do cilindro.

3. O fluxo de calor advindo dos gases da combustão pode ser dado pela lei de resfriamento de Newton(2.2) e o coeficiente de transferência de calor pode ser determinado através de correlações disponíveis na literatura, da forma sugerida por (2.32).

4. A transferência de calor por convecção para o refrigerante que passa nas galerias ao redor dos cilindros pode ser modelada, segundo BOHAC *et al.* (1996), como o escoamento através de uma bancada de cilindros.

5. A radiação é desprezível em motores de combustão interna de ignição por centelha(HEYWOOD (1988)).

6. Assume-se que o bloco é isolado termicamente do ambiente externo.

7. A transferência de calor para o óleo, e a geração de calor serão desprezadas no presente modelo, já que o interesse principal é analisar como se dá a transferência de calor entre os gases da combustão e o refrigerante, que é o principal fenômeno que ocorre dentro do bloco do motor.

Foi criado então um modelo idealizado de um bloco de motor de 4 cilindros, omitindo propositalmente muitos detalhes geométricos e que seriam necessários para fabricação, mas considerado suficiente para a análise proposta.

Neste ponto, faz-se necessário realizar algumas ressalvas quanto às hipóteses adotadas, que distanciam o modelo da realidade. A hipótese (2) é discutível, porque em muitos casos existem oscilações internas de temperatura no interior do bloco, fora o fato de que é muitíssimo difícil na vida real um motor atingir regime permanente. Isso também está relacionado ao fato de que a combustão em cada cilindro ocorre em momentos diferentes, e uma aproximação mais realista seria a adoção de um regime cíclico e que leve em conta temperaturas diferentes em cada cilindro. Entretanto para a análise aqui proposta, é suficiente adotar uma média temporal.

No caso da hipótese (6), por mais que o bloco perca uma quantidade significativa de calor para o ambiente externo, adotou-se um modelo mais conservativo na análise, de forma que todo o calor rejeitado pelo bloco deve ser direcionado ao fluido de refrigeração.

3.2 Modelo Matemático

Dada as hipóteses listadas na seção anterior, a equação governante do problema é dada pela equação de Laplace(2.16) com condições de contorno de terceiro tipo

nas superfícies internas do cilindro, nas superficíes externas do bloco e nas galerias por onde escoa o refrigerante. Matematicamente, tem-se que:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = 0 \qquad \text{em } \Omega \tag{3.1}$$

$$k\frac{\partial T}{\partial n} + h_c T = h_c T_c \qquad \text{em } \Gamma_c \tag{3.2}$$

$$k\frac{\partial T}{\partial n} + \overline{h}_g T = \overline{h}_g \overline{T}_g \qquad \text{em } \Gamma_g \tag{3.3}$$

$$\frac{\partial T}{\partial n} = 0 \qquad \text{em } \Gamma_\infty \tag{3.4}$$

3.3 Modelo Numérico

A aplicação do método dos elementos finitos sobre o problema acima nos dá a seguinte forma para as equações (2.29) e (2.30):

$$\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} \mathbf{B}^{(e)^{\mathbf{T}}} k \mathbf{B}^{(e)} d\Omega + \int_{\Gamma_c^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h_c \mathbf{N}^{(e)} d\Gamma + \int_{\Gamma_g^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} \overline{h}_g \mathbf{N}^{(e)} d\Gamma \tag{3.5}$$

$$\mathbf{f}^{(e)} = \int_{\Gamma_c^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h_c T_c d\Gamma + \int_{\Gamma_g^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} \overline{h}_g \overline{T}_g d\Gamma \tag{3.6}$$

onde k é a condutividade térmica e é conhecida. Pode-se definir

$$\mathbf{K_{cond}}^{(e)} = \int_{\Omega^{(e)}} \mathbf{B}^{(e)^{\mathbf{T}}} k \mathbf{B}^{(e)} d\Omega \tag{3.7}$$

$$\mathbf{K_c}^{(e)} = \int_{\Gamma_c^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h_c \mathbf{N}^{(e)} d\Gamma \tag{3.8}$$

$$\mathbf{K_g}^{(e)} = \int_{\Gamma_g^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} \overline{h}_g \mathbf{N}^{(e)} d\Gamma \tag{3.9}$$

$$\mathbf{f_c}^{(e)} = \int_{\Gamma_c^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} h_c T_c d\Gamma \tag{3.10}$$

$$\mathbf{f_g}^{(e)} = \int_{\Gamma_g^{(e)}} \mathbf{N}^{(e)^{\mathbf{T}}} \overline{h}_g \overline{T}_g d\Gamma \tag{3.11}$$

que são, respectivamente, a matriz de condutividade térmica, a matriz de convecção do refrigerante, a matriz de convecção dos gases de combustão, o vetor de carregamento de convecção do refrigerante e o vetor de carregamento de convecção dos gases de combustão.

Torna-se necessário a determinação dos coeficientes de transferência de calor e as temperaturas das condições de contorno. Como foi dito anteriormente, h_c pode ser

determinada através da correlação para convecção forçada através de uma bancada de cilindros, dada por

$$\frac{\overline{h}D}{k_c} = 1,13 c_0 Re^n Pr^{1/3} \tag{3.12}$$

para $2000 \leq Re \leq 40000$, $Pr > 0,7$ e uma bancada de mais de 10 cilindros, que pode ser corrigido, caso tenha-se um número menor de cilindros.

$$h_N = c_1 h_{N \geq 10} \tag{3.13}$$

e também os números adimensionais são dados por

$$Re = \frac{G_{max} D}{\mu_c} \text{ e } Pr = \frac{\mu_c c_{p,c}}{k_c}$$

D é o diâmetro do cilindro e G_{max} é a vazão mássica máxima, onde ocorre a menor área transversal por onde passa o fluido refrigerante. Os parâmetros c_0, c_1 e n são tabelados, advindos da tabela 8-3 e 8-4 de ÖZIŞIK (1990).

Segundo FERGUSON e KIRKPATRICK (2016) o coeficiente de transferência de calor e temperatura dos gases pode ser calculado um valor médio:

$$\overline{T}_g = \frac{1}{4\pi \overline{h}_g} \int_0^{4\pi} h_g(\theta) T_g(\theta) d\theta \tag{3.14}$$

$$\overline{h}_g = \frac{1}{4\pi} \int_0^{4\pi} h_g(\theta) d\theta \tag{3.15}$$

onde $T_g(\theta)$ é a temperatura do gás pode ser calculada por uma simulação de um ciclo para prever temperaturas instantâneas e $h_g(\theta)$ é dado pela correlação de Woschni

$$h_g = 3,26 p^{0,8} U^{0,8} b^{-0,2} T^{-0,55} \tag{3.16}$$

onde p, U, b e T são a pressão do gás[kPa], velocidade característica do gás[m/s], diâmetro do cilindro[m] e temperatura do gás[K], respectivamente. A velocidade característica é dada por

$$U = 6,18 \overline{U}_p \quad \text{para a admissão e exaustão.} \tag{3.17}$$

$$U = 2,28 \overline{U}_p \quad \text{para a compressão.} \tag{3.18}$$

$$U = 6,18 \overline{U}_p + 0,00324 T_r \frac{V_d}{V_r} \frac{p - p_m}{p_r} \quad \text{para a combustão e expansão.} \tag{3.19}$$

onde $\overline{U}_p$, T_r, V_r, p_r, p_m e V_d são, respectivamente, a velocidade média do pistão[m/s], a temperatura[K], volume[m^3], a pressão[kPa] de quando a válvula de admissão se

fecha, e a pressão que aconteceria sem combustão[kPa], e o volume deslocado(m^3). p_m poder ser obtida a partir da relação isentrópica com um estado de referência e a velocidade média do pistão é dada por $2 \times curso[\text{m}] \times \frac{RPM}{60}$.

A simulação para a obtenção de T_g e h_g foi feita a partir de um código disponível no apêndice F.17 de FERGUSON e KIRKPATRICK (2016). A partir da hipótese de que durante as etapas de exaustão e admissão a transferência de calor para as paredes dos cilindros é desprezível com relação às etapas de compressão, combustão e expansão, foi utilizado o código *woschniheattransfer.m*, soluciona um modelo de emissão de energia finito, para calcular $h_g(\theta)$ e $T_g(\theta)$ para $0 \leq \theta \leq 2\pi$, correspondente à meio ciclo Otto, ou uma volta de virabrequim. A partir dos dados gerados, foi feita a integração numérica deles de acordo com (3.14) e (3.15), obtendo assim $\overline{h}_g$ e $\overline{T}_g$.

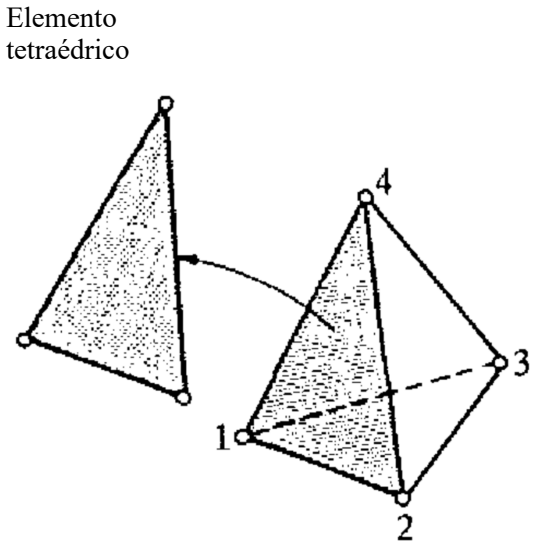


Figura 3.1: Elemento de malha tetraédrico. Adaptado de REDDY (1993).

Também é necessário a definição do tipo de elemento e função de forma que será utilizada no método de elementos finitos. Neste caso, como trata-se de um problema tridimensional, serão escolhidos os elementos tetraédricos lineares, ou seja, a função de forma utilizada para aproximar $T(x, y, z)$ em cada elemento é um polinômio do primeiro grau, que é o mais simples para este tipo de problema, e cada elemento terá $n_{el} = 4$ nós. Por consequência, as superfícies onde são aplicadas as condições de contornos serão formadas por elementos triangulares lineares, de 3 nós. Assim, segundo NITHIARASU *et al.* (2016):

$$N_i = \frac{1}{6V}(a_i + b_i x + c_i y + d_i z) \text{ para } i = 1, 2, 3, 4 \tag{3.20}$$

e para os triângulos dos contornos

$$N_i = \frac{1}{2A}(a_i + b_i x + c_i y) \text{ para } i = 1, 2, 3 \tag{3.21}$$

O volume do tetraédro V é

$$V = \frac{1}{6} \det \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \tag{3.22}$$

onde (x_i, y_i, z_i) é a coordenada de um dos vértices do tedraédro e a área de um triângulo, para os elementos da superfície, é

$$A = \frac{1}{2} \| [(x_3, y_3, z_3) - (x_1, y_1, z_1)] \times [(x_2, y_2, z_2) - (x_1, y_1, z_1)] \| \tag{3.23}$$

e (x_i, y_i, z_i), se referem às coordenadas dos vértices dos triângulos, com $i = 1, 2, 3$.

Escrevendo $T_c^{(e)} = \mathbf{N}^{(e)}\mathbf{T_c}$ e $\overline{T}_g^{(e)} = \mathbf{N}^{(e)}\overline{\mathbf{T}}_\mathbf{g}$, onde $\mathbf{T_c} = \begin{bmatrix} T_c & T_c & T_c \end{bmatrix}^\mathbf{T}$ e $\overline{\mathbf{T}}_\mathbf{g} = \begin{bmatrix} \overline{T}_g & \overline{T}_g & \overline{T}_g \end{bmatrix}^\mathbf{T}$, e substituindo as funções de forma definidas anteriormente em (3.7), (3.8), (3.9), (3.10) e (3.11), obtém-se

$$\mathbf{K_{cond}}^{(e)} = k(\mathbf{K_x}^{(e)} + \mathbf{K_y}^{(e)} + \mathbf{K_z}^{(e)}) \tag{3.24}$$

$$\mathbf{K_x}^{(e)} = \frac{1}{36V} \begin{bmatrix} b_1b_1 & b_1b_2 & b_1b_3 & b_1b_4 \\ b_2b_1 & b_2b_2 & b_2b_3 & b_2b_4 \\ b_3b_1 & b_3b_2 & b_3b_3 & b_3b_4 \\ b_4b_1 & b_4b_2 & b_4b_3 & b_4b_4 \end{bmatrix} \tag{3.25}$$

$$\mathbf{K_y}^{(e)} = \frac{1}{36V} \begin{bmatrix} c_1c_1 & c_1c_2 & c_1c_3 & c_1c_4 \\ c_2c_1 & c_2c_2 & c_2c_3 & c_2c_4 \\ c_3c_1 & c_3c_2 & c_3c_3 & c_3c_4 \\ c_4c_1 & c_4c_2 & c_4c_3 & c_4c_4 \end{bmatrix} \tag{3.26}$$

$$\mathbf{K_z}^{(e)} = \frac{1}{36V}\begin{bmatrix} d_1d_1 & d_1d_2 & d_1d_3 & d_1d_4 \\ d_2d_1 & d_2d_2 & d_2d_3 & d_2d_4 \\ d_3d_1 & d_3d_2 & d_3d_3 & d_3d_4 \\ d_4d_1 & d_4d_2 & d_4d_3 & d_4d_4 \end{bmatrix} \tag{3.27}$$

$$\mathbf{K_c}^{(e)} = \frac{h_cA}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.28}$$

$$\mathbf{K_g}^{(e)} = \frac{\overline{h}_gA}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.29}$$

$$\mathbf{f_c}^{(e)} = \frac{h_cA}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} T_c \\ T_c \\ T_c \end{bmatrix} \tag{3.30}$$

$$\mathbf{f_g}^{(e)} = \frac{\overline{h}_gA}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} \overline{T}_g \\ \overline{T}_g \\ \overline{T}_g \end{bmatrix} \tag{3.31}$$

onde

$$b_1 = -(y_2 - y_4)(z_3 - z_4) + -(y_3 - y_4)(z_2 - z_4)$$
$$b_2 = -(y_3 - y_4)(z_1 - z_4) + -(y_1 - y_4)(z_3 - z_4)$$
$$b_3 = -(y_1 - y_4)(z_2 - z_4) + -(y_2 - y_4)(z_1 - z_4)$$
$$b_4 = -(b_1 + b_2 + b_3)$$

$$c_1 = -(x_3 - x_4)(z_2 - z_4) + -(x_2 - x_4)(z_3 - z_4)$$
$$c_2 = -(x_1 - x_4)(z_3 - z_4) + -(x_3 - x_4)(z_1 - z_4)$$
$$c_3 = -(x_2 - x_4)(z_1 - z_4) + -(x_1 - x_4)(z_2 - z_4)$$
$$c_4 = -(c_1 + c_2 + c_3)$$

$$d_1 = -(x_2 - x_4)(y_3 - y_4) + -(x_3 - x_4)(y_2 - y_4)$$
$$d_2 = -(x_3 - x_4)(y_1 - y_4) + -(x_1 - x_4)(y_3 - y_4)$$
$$d_3 = -(x_1 - x_4)(y_2 - y_4) + -(x_2 - x_4)(y_1 - y_4)$$
$$d_4 = -(d_1 + d_2 + d_3)$$

Através do processo de *assembling*, ou montagem, da matriz global, reúne-se as informações das matrizes e vetores elementares nas matrizes globais do problema, através do mapeamento entre os nós do domínio e dos contornos e os nós dos elementos, de forma que, ao final da montagem, haverá um sistema linear a ser resolvido

$$(\mathbf{K_{cond}} + \mathbf{K_c} + \mathbf{K_g})\mathbf{T} = \mathbf{K_c}\mathbf{T_c} + \mathbf{K_g}\overline{\mathbf{T}}_\mathbf{g} \tag{3.32}$$

onde $\mathbf{T}$ é o vetor de temperaturas nodais a ser determinado, $\mathbf{T_c}$ e $\overline{\mathbf{T}}_\mathbf{g}$ são vetores em que cada elemento é o valor da temperatura do refrigerante ou do gás, que são conhecidos. O sistema linear é $n \times n$, onde n é o número de nós total do domínio.

3.4 Implementação Computacional

A implementação computacional para solução do problema proposto consiste em:

- Criação de um modelo tridimensional a partir do software de modelagem 3D FreeCAD, exportando o modelo para um formato compatível com o gerador de malhas Gmsh, que é utilizado para gerar a malha propriamente dita.
- Um módulo para leitura de malha em Python, *meshreader.py*, um para o cálculo de matrizes elementares e montagem(*mefmatrices.jl*), e um para implementação de condições de contorno(*boundary_conditions.jl*, estes dois últimos em Julia.
- Um Jupyter Notebook, escrito em Julia, onde são fornecidos os dados de entrada para os problemas e que chama as funções contidas nos outros scripts, e onde é solucionado o sistema linear e gerado os resultados.

3.4.1 Modelo 3D e Malha

O FreeCAD é um software de modelagem de código aberto, nele é possível construir modelos tridimensionais de objetos e realizar a exportação desse modelo para diferentes formatos. Tendo criado o modelo do bloco e feito a exportação para o formato *BRep*, o arquivo pode ser aberto pelo Gmsh, onde então é possível fazer através da interface gráfica a determinação dos *physical groups*, ou grupos físicos em tradução livre, que são grupos de entidades geométricas com significado físico,

de forma que é possível definir as superfícies onde atua as condições de contorno convectivas, além de definir o sólido correspondente ao metal do bloco do motor. Após este passo já é possível realizar a geração de malha, onde é possível definir o fator de tamanho do elemento e salvar a malha gerada.

O arquivo da malha é do formato *msh*, nele estão registrados os nós, os elementos, suas coordenadas e à que grupo físico pertencem, e também é possível extrair a matriz de mapeamento de nós elementares entre globais, utilizada para a montagem das matrizes globais do sistema linear. Vale ressaltar que ao exportar o modelo do FreeCAD para o Gmsh e gerar a malha, as unidades das coordenadas dos pontos são em milímetros.

A leitura de malha é feita através do script *meshreader.py*, disponível em A, de autoria do professor Gustavo Rabello dos Anjos e adaptado pelo autor. Esse código é responsável por extrair do arquivo de malha as informações necessárias para a realização da simulação do problema, e posteriormente gerar o arquivo de resultados em *VTK*, para execução em um software de visualização, e isso é feito através da biblioteca *meshio* para Python, que facilita tanto a leitura de malhas geradas quanto a exportação de resultados para a visualização no Paraview.

3.4.2 Matrizes Elementares, Montagem e Condições de Contorno

O módulo *mefmatrices.jl* contém o cálculo das matrizes elementares e a montagem das matrizes globais conforme o apresentado anteriormente, recebendo como entrada dados sobre a malha. O código faz uso de matrizes esparsas, no formato COO(Cordinate Format), que recebe uma coordenada (i, j) correspondente à linha i e coluna j da matriz, e o valor do elemento. Este formato-se destaca-se pela rapidez no acesso à informação dentro da matriz, e a utilização de matrizes esparsas reduz consideravelmente o uso de memória RAM, ao armazenar apenas os valores diferentes de zero, o que é vantajoso pois no método de elementos finitos é comum matrizes com a maior parte dos elementos nulos. A linguagem de programação Julia tem implementado em sua biblioteca padrão matrizes esparsas e operações envolvendo estruturas deste tipo, a partir dos pacotes *SparseArrays* e *LinearAlgebra*, que é o pacote padrão de álgebra linear da linguagem.

As matrizes de mapeamento para as condições do contorno convectiva são montadas separadamente no módulo *boundary_conditions.jl*, utilizando-se de dados vindos do leitor de malhas e de inputs do usuário, como o nome da condição de contorno em questão, que deve coincidir com o nome dado ao seu respectivo grupo físico. Também foi implementada a imposição de condições de contorno do tipo Dirichlet.

3.4.3 Notebook e Simulação

A configuração da simulação é feita num Jupyter Notebook, uma plataforma interativa de programação, de forma que é possível separar as diferentes etapas em células de código, observar o tempo de execução, saídas e eventuais erros em cada etapa. No notebook são chamados os módulos descritos anteriormente, são informados dados necessários, como propriedades físicas e outros cálculos, como aqueles necessários para determinação dos coeficientes de transferência de calor. Além de que é possível modificar parâmetros em determinadas células para observar os diferentes resultados, sem necessariamente rodar todo o script novamente.

A visualização dos resultados e pós-processamento é feita no Paraview, software de visualização gráfica de código aberto, onde também é possível gerar gráficos a partir dos dados, cortes na geometria, e outras formas de se analisar os resultados obtidos.

Capítulo 4

Validação e Verificação

4.1 Validação com uma solução analítica

Nesta seção será realizada a validação do código em questão, ou seja, se de fato ele fornece uma solução confiável da equação governante que se propõe a solucionar, neste caso, a equação de Laplace para temperatura com uma condição de contorno convectiva. A validação será realizada comparando-se a solução analítica para um problema bidimensional com a solução de elementos finitos para um problema tridimensional, construído de forma que cada um de seus planos transversais seja equivalente ao problema 2D.

O problema bidimensional é dado na seção III do capítulo 5 em CARSLAW e JAEGER (1959). Ele é definido matematicamente como

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \qquad 0 < x < a,\ 0 < y < b \tag{4.1}$$

$$T(x, y) = T_1 \qquad y = 0 \tag{4.2}$$

$$k\frac{\partial T}{\partial y} + hT = 0 \qquad y = b \tag{4.3}$$

$$\frac{\partial T}{\partial x} = 0 \qquad x = 0 \tag{4.4}$$

$$k\frac{\partial T}{\partial x} + hT = 0 \qquad x = a \tag{4.5}$$

e tem a solução dada pela equação (5.20) é

$$T(x, y) = 2HT_1 \sum_{n=1}^{\infty} \frac{\cos \alpha_n x(\alpha_n \cosh \alpha_n(b - y) + H \sinh \alpha_n(b - y))}{[(\alpha_n^2 + H^2)a + H](\alpha_n \cosh \alpha_n b + H \sinh \alpha_n b) \cos \alpha_n a} \tag{4.6}$$

onde $H = h/k$ e α_n são raízes da equação transcendental $\alpha \tan \alpha a = H$, que podem ser calculadas numericamente. Por simplicidade, é definido $T_1 = 10$ °C, $a = b = 1$ m, $h = 100$ W/m^2 e $k = 100$ W/m·°C. O problema tridimensional é similar ao descrito acima, definido num cubo, na mesma região em xy, com $0 < z < c$, $c = 1$, e ele está isolado em $z = 0$ e $z = c$. Isso deve produzir uma solução, tal que, um plano em qualquer z seja reduzido ao problema bidimensional, pela natureza matemática da equação de Laplace.

Foram criadas uma placa e um cubo no FreeCAD e as malhas foram geradas através do Gmsh, com o mesmo fator de tamanho do elemento de 0.1, a malha 2D apenas para plotar os pontos da solução bidimensional, e a malha 3D para solução por meio do MEF.

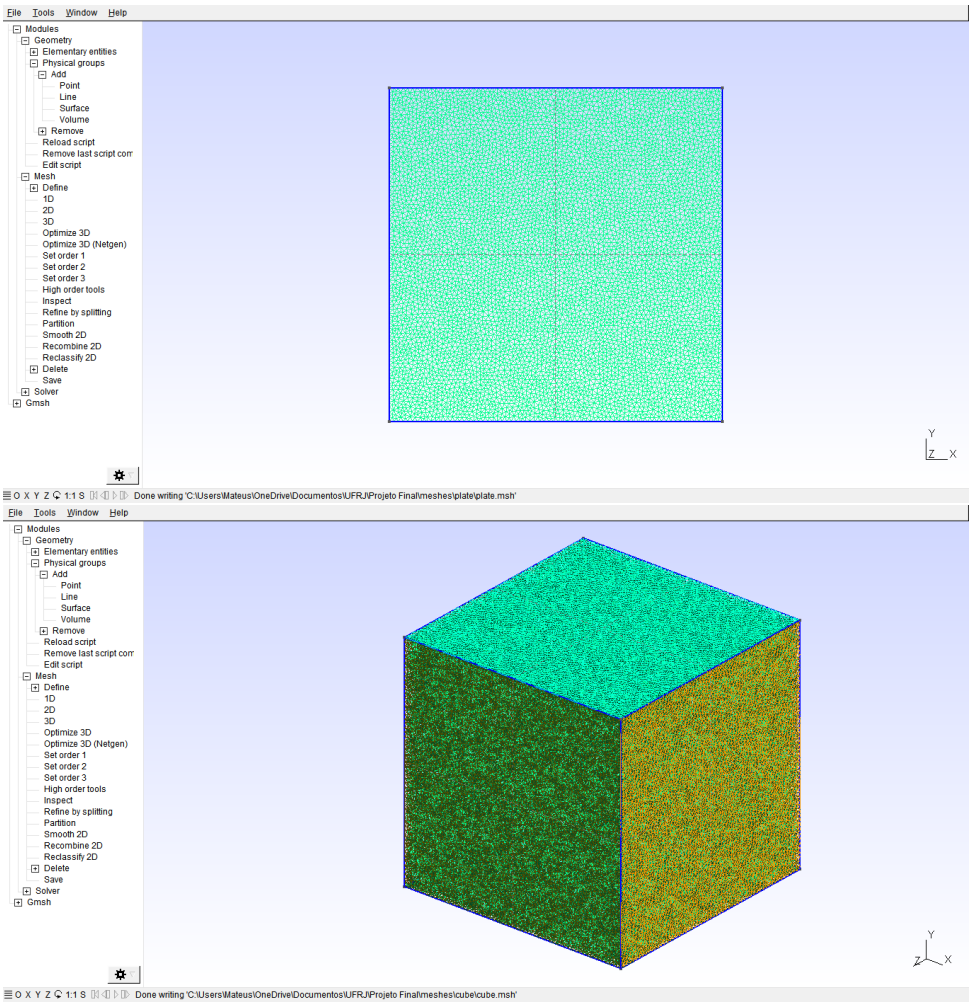


Figura 4.1: Malhas 2D e 3D. Elaborada pelo autor.

A seguir estão apresentados os resultados no Paraview:

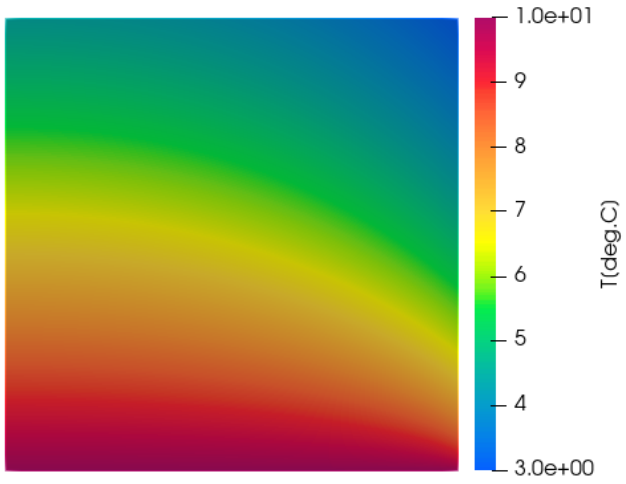


Figura 4.2: Solução analítica.

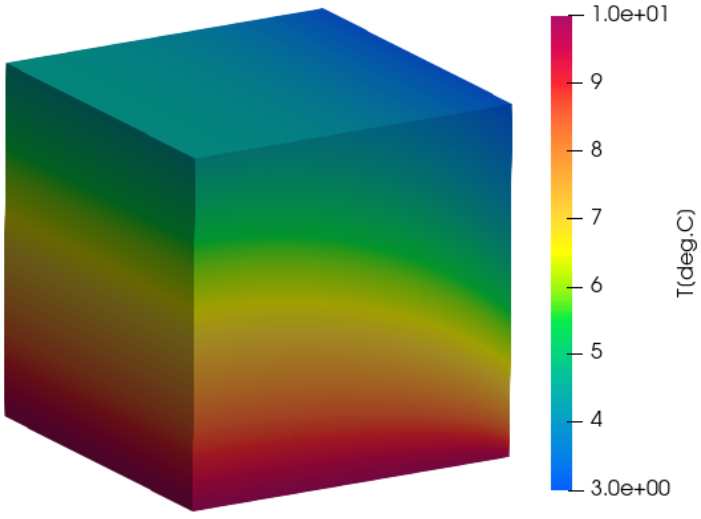


Figura 4.3: Solução por meio do método de elementos finitos.

Comparando-se o perfil de temperatura em $x = 0,5$ m entre o caso 2D com o caso 3D, escolhendo-se um plano em $z = 0,5$ m, com uma amostra de 1000 pontos:

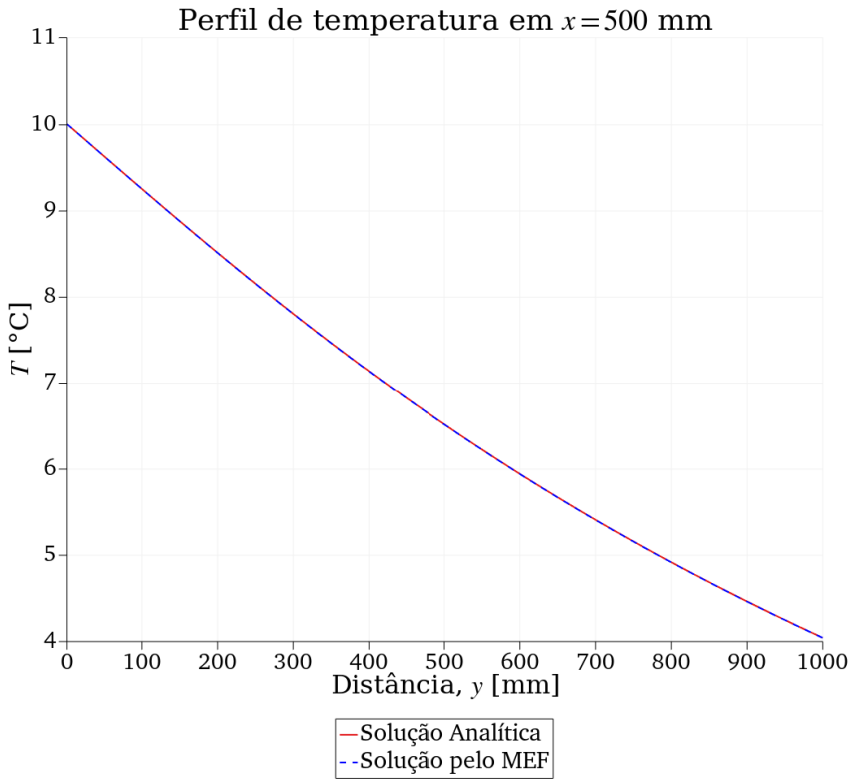


Figura 4.4: Comparação entre os perfis de temperatura.

Escolhendo outro perfil, agora em $x = 0,75$ m e um plano em $z = 0,2$ m:

Para ambos os casos, temos os erros relativos:

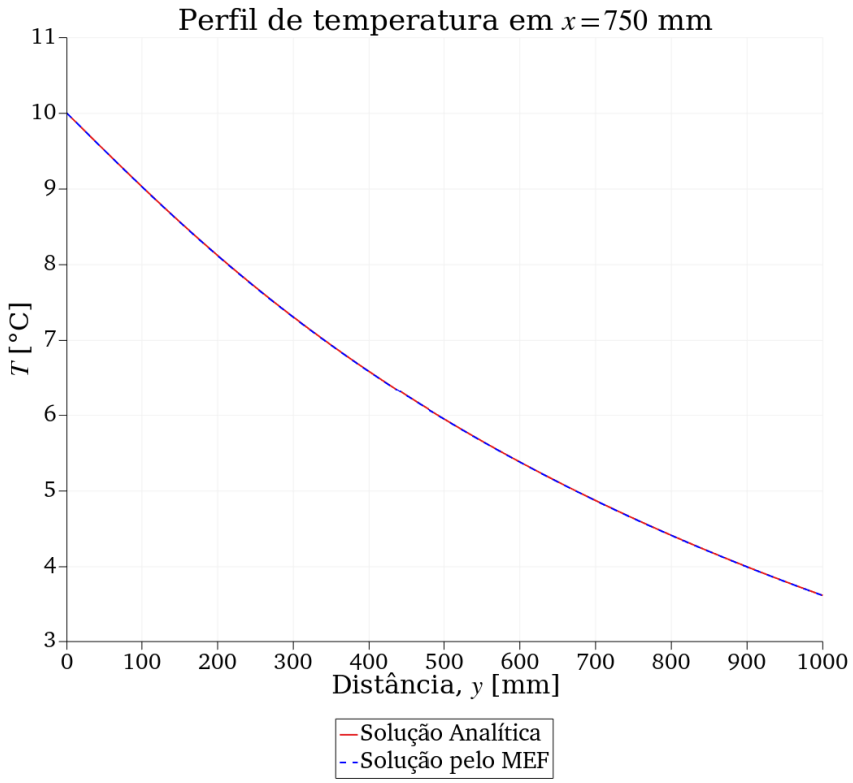


Figura 4.5: Comparação entre os perfis de temperatura.

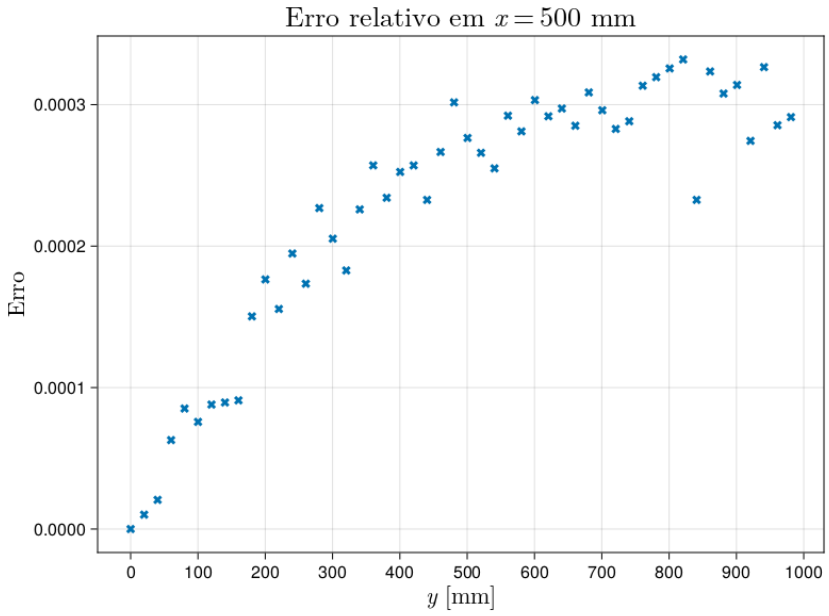


Figura 4.6: Erro relativo para $x = 0,5$ m.

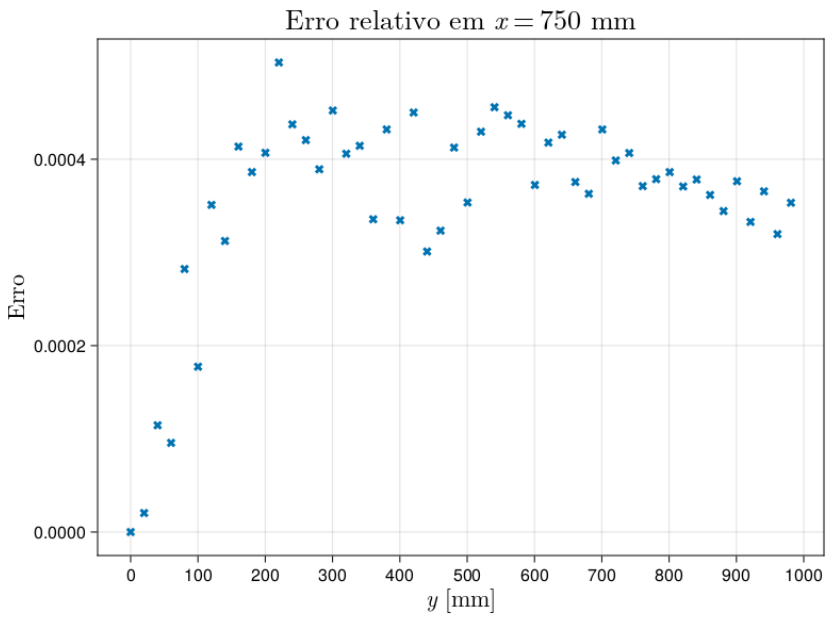


Figura 4.7: Erro relativo para $x = 0,75$ m.

É possível observar que os perfis de temperatura observados, são semelhantes entre si, e a análise do erro relativo, de ordem 10^{-4}, de onde é possível concluir que de fato o código de elementos finitos proposto consegue solucionar a equação governante do problema de forma satisfatória. Outro aspecto é se destacar, é como o código satisfez a condição de contorno de primeiro tipo em $y = 0$.

4.2 Verificação por convergência de malha

Para a verificação do código, foi feita uma análise de convergência de malha, que é observar o comportamento do erro em função do número de elementos da malha. A tendência é que, com uma malha cada vez mais refinada, o erro do código irá reduzir cada vez menos, até chegar num ponto que não haverá melhora significativa, de forma que aumentar o número de elementos acarretaria em um custo computacional maior, sem ganho em acurácia. Para o último perfil de temperatura apresentado anteriormente, foi calculado a norma euclidiana do erro relativo entre todos os pontos da solução analítica e numérica, e apresentado em função do número de elementos da malha tridimensional.

$$\text{norma do erro relativo} = \frac{||T_{analitico} - T_{aprox}||}{||T_{analitico}||}$$

Para os testes, foram utilizados os parâmetros descritos na tabela 4.1.

Fator de tamanho de elemento	Número de elementos	Número de pontos
1	1110	357
0,75	2362	681
0,5	6786	1714
0,3	30985	6626
0,2	99228	19402
0,1	717801	129025
0,075	1614440	283872
0,07	2057814	358216
0,065	2500867	433396
0,06	3348619	575392

Tabela 4.1: Parâmetros de malha dos testes.

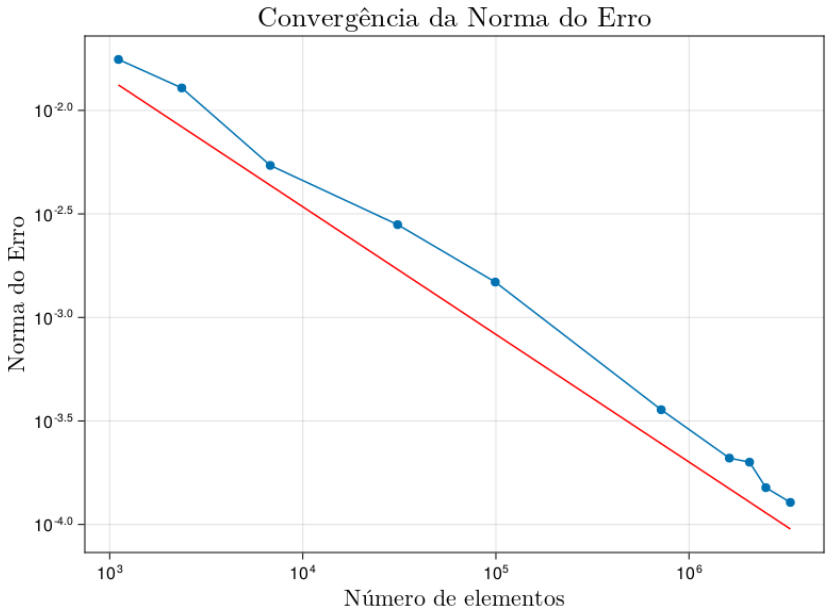


Figura 4.8: Análise de convergência de malha.

Para os últimos 4 valores do fator de tamanho do elemento, o comportamento assintótico fica bastante evidente, pois não há mais mudanças de ordem de grandeza do erro, e para valores menores ocorreram erros de falta de memória RAM para o método tradicional de solução de sistemas lineares em Julia. A partir desta análise, fica evidente que o fator de tamanho de elemento de $0,1$ é satisfatório, gerando ainda um número de elementos da ordem de 10^5, e resultando em um erro relativo de ordem de 10^{-4}. No gráfico 4.8 também é mostrada uma reta de tendência de inclinação $-0,6$, na escala logarítmica, da convergência do erro, cuja inclinação foi calculada ajustando-se uma reta aos dados obtidos, por meio do método dos mínimos quadrados.

Capítulo 5

Resultados e Discussões

Neste capítulo serão apresentados os resultados obtidos para a simulação seguindo a metodologia exposta no 3. Foram realizadas simulações considerando dois tipos de fluido refrigerantes: água e etilenoglicol, que já foi proposto na utilização como refrigerante mas foi descartado, de forma que normalmente é usado numa mistura com água(FERGUSON e KIRKPATRICK (2016)) para aumentar o ponto de ebulição e diminuir o ponto de solidificação, e nunca é utilizado sozinho na prática, mas é considerado aqui num contexto idealizado para comparação. Assume-se que o sistema esteja a 1 bar, que a temperatura do refrigerante entra nas galerias do refrigerante à 40 °C e sai à 80 °C, de forma que as propriedades dos fluidos são calculados na média aritmética, de 60 °C, para serem usados na correlação (3.12), e obtidos a partir da tabela B-2 de ÖZIŞIK (1990). A geometria das galerias do refrigerante era tal que os coeficientes retirados das tabelas de ÖZIŞIK (1990) são

- $c_0 = 0,348$
- $c_1 = 0,90$
- $n = 0,592$

com os cilindros, neste contexto, tendo 95 mm de diâmetro e 120 mm de espaçamento entre os seus centros. Na tabela 5.1 estão resumidas os resultados obtidos para os parâmetros dos fluidos refrigerantes necessários para a simulação.

Adaptando o código do apêndice F.17 de FERGUSON e KIRKPATRICK (2016) para a geometria do modelo, com 85 mm de diâmetro do cilindro e 90 mm de curso, e para uma velocidade de aproximadamente 4000 rpm, foi possível obter $\overline{T}_g$ e $\overline{h}_g$.

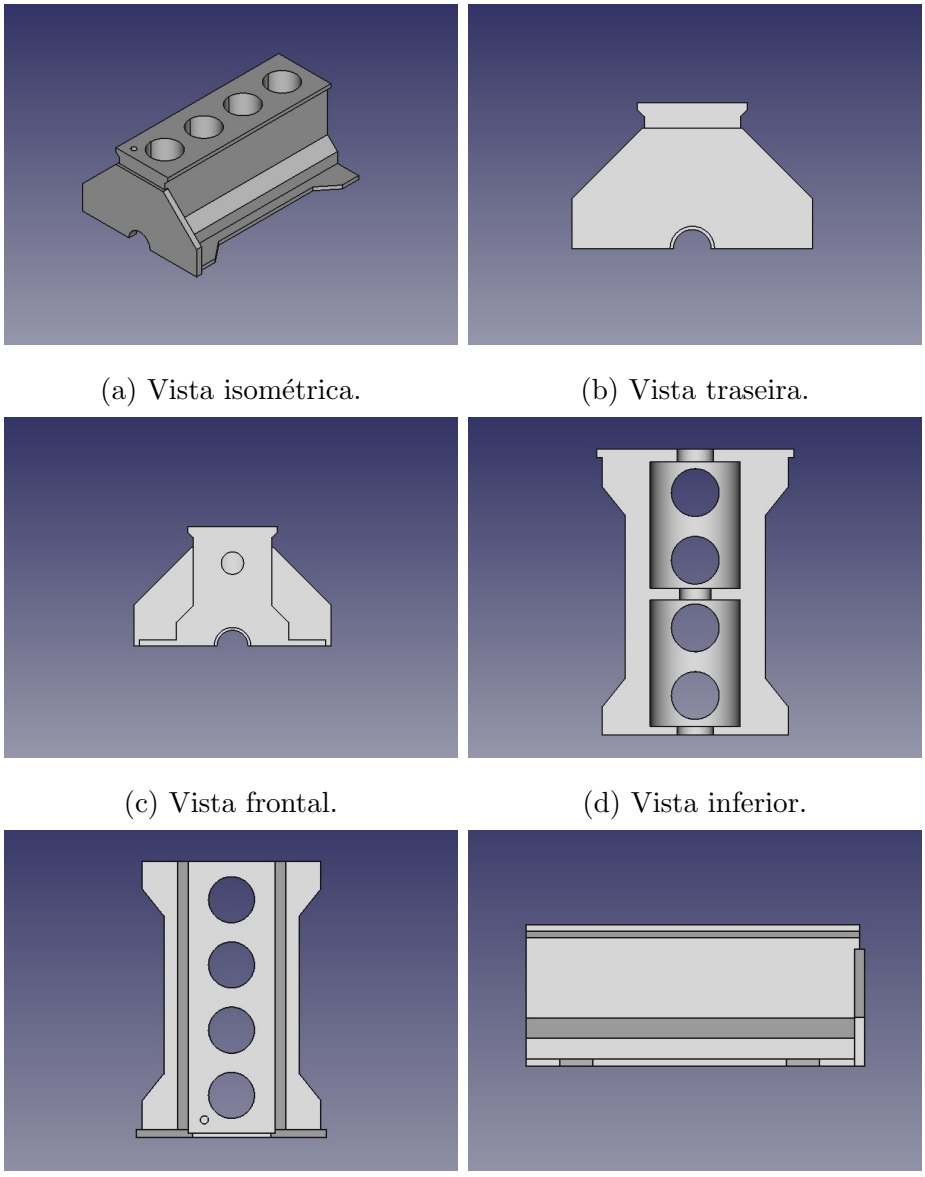
Parâmetro	Água	Etilenoglicol
ρ [kg/m^3]	985,46	1087,66
μ [N·s/m^2]	4,71 $\times$ 10^{-4}	5,17 $\times$ 10^{-3}
k [W/m·°C]	0,651	0,260
Pr	3,02	51
h [W/m^2]	1654,10	435,27

Tabela 5.1: Parâmetros para simulação.

- $\overline{T}_g = 1617,30$ °C
- $\overline{h}_g = 275,158$ W/m^2

O material do bloco suposto como alumínio, que numa temperatura intermediária entre a do fluido refrigerante e a temperatura média do gás, aproximadamente 600 K, da tabela A.1 de BERGMAN *et al.* (2011), possui $k = 231$ W/m·°C. O bloco tem comprimento de 0,5 m, altura de 0,2115 m e largura de 0,35 m na seção mais larga. Na figura 5.1 está representado o bloco do motor. No B estão diversos desenhos mostrando em maior detalhamento as medidas do bloco.

A malha gerada é com fator de tamanho do elemento de $0,1$, tal qual foi concluído na análise de convergencia no capítulo 4,representados na figura 5.2. Isso totaliza para simulação um total de 149442 elementos e 41922 pontos na malha.



(a) Vista isométrica. (b) Vista traseira.

(c) Vista frontal. (d) Vista inferior.

(e) Vista superior. (f) Vista lateral.

Figura 5.1: Bloco do motor no FreeCAD.

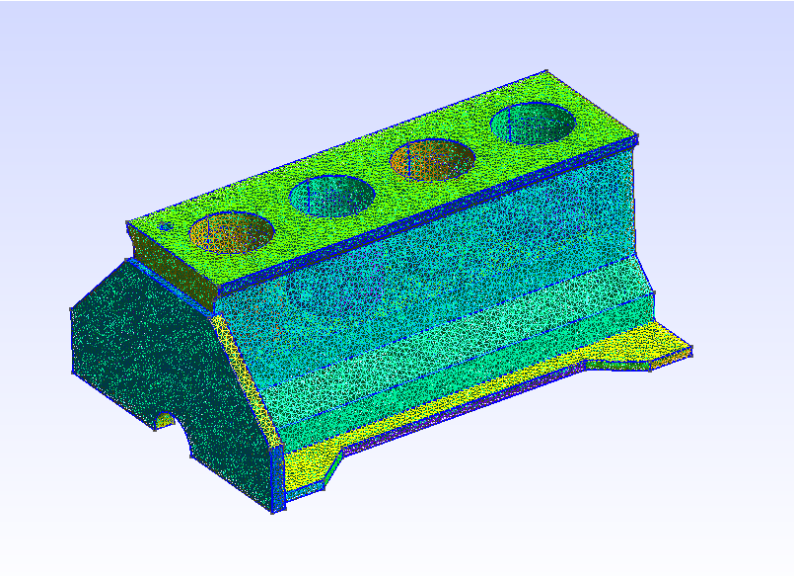
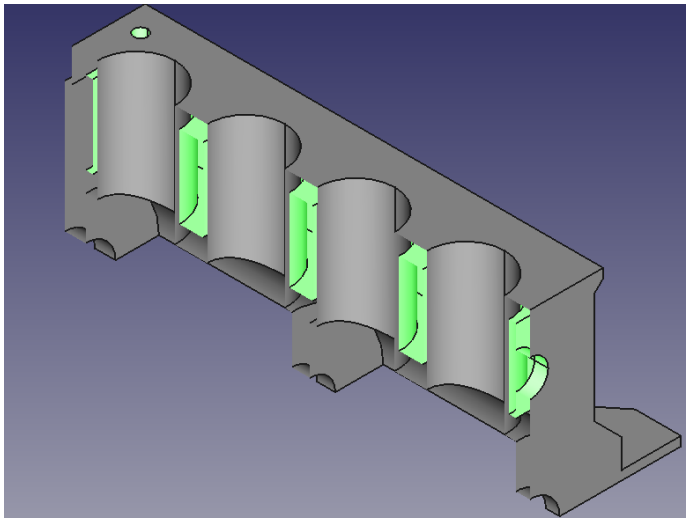
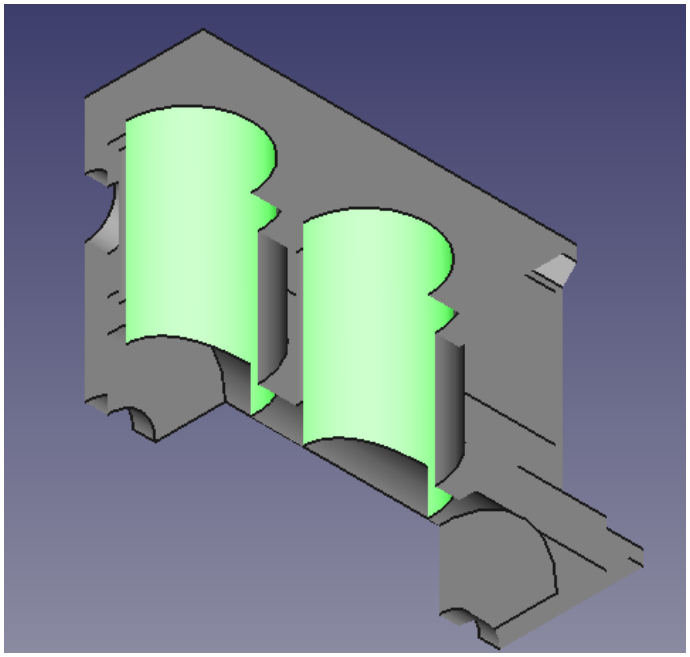


Figura 5.2: Malha gerada no Gmsh.

Em cortes feitos no FreeCAD é possível observar as galerias do fluido refrigerante, além das superfícies externas. Em toda superfície interior das galerias é aplicada a condição de contorno convectiva, correspondente à equação (3.2). Na superfícies internas correspondentes à dos quatro cilindros foi imposta a condição de contorno do terceiro tipo devida aos gases da combustão(3.3). Nas superfícies restantes, do exterior e parte inferior do bloco, são adotadas as condições de contorno de fluxo nulo(3.4).



(a) Condição de contorno de convecção do fluido de resfriamento.



(b) Condição de contorno de convecção dos gases da combustão.

Figura 5.3: Regiões de condição de contorno convectiva.

Nas duas figuras a seguir estão apresentados os resultados de forma geral, através da visualização do campo de temperaturas no bloco do motor sendo resfriado por água e por etilenoglicol.

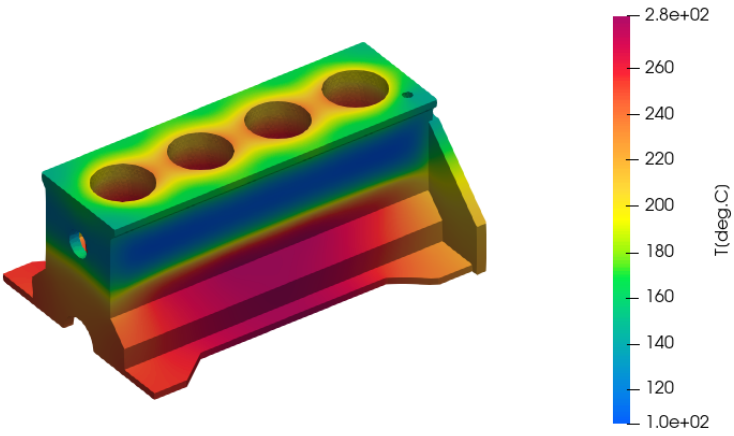


Figura 5.4: Bloco sendo resfriado por água.

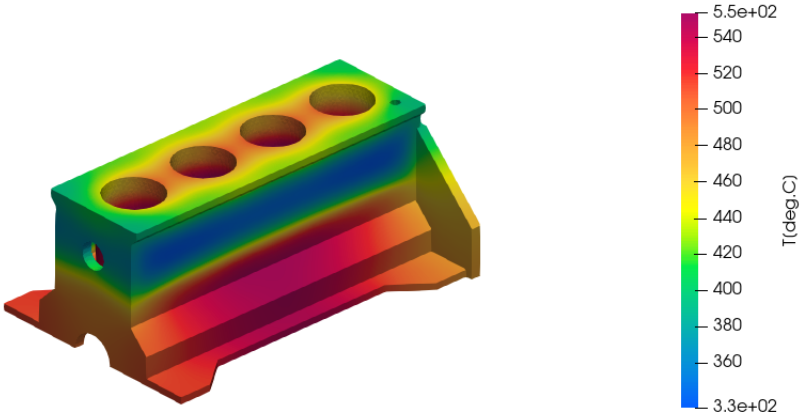


Figura 5.5: Bloco sendo resfriado por etilenoglicol.

Agora serão apresentados os resultados obtidos das simulações de forma comparativa. O Jupyter Notebook utilizado para simular os problemas está no A. Nas figuras à seguir, os modelos do lado esquerdo são os simulados com água, e do lado direito aqueles simulados com etilenoglicol.

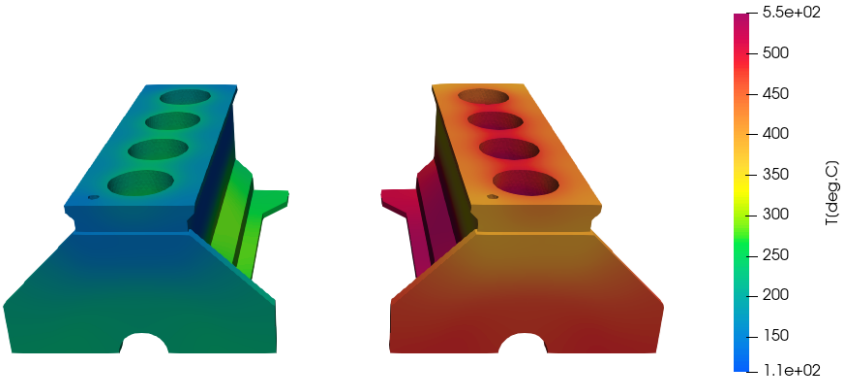


Figura 5.6: Vista traseira.

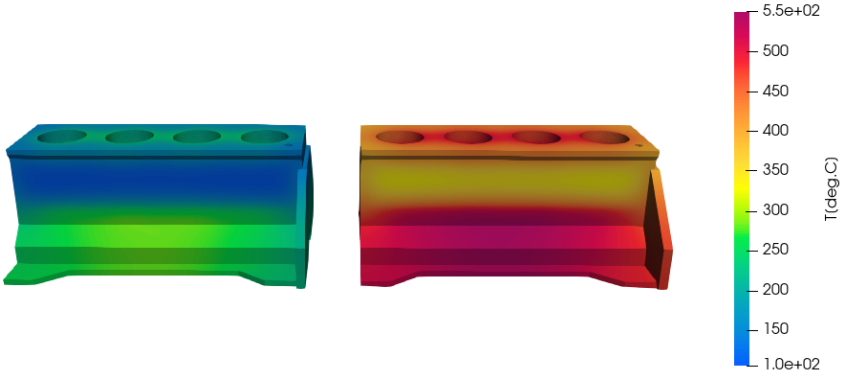


Figura 5.7: Vista lateral.

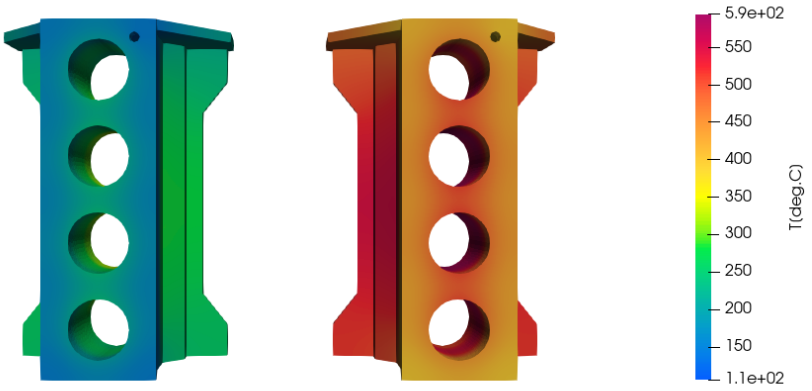


Figura 5.8: Vista superior.

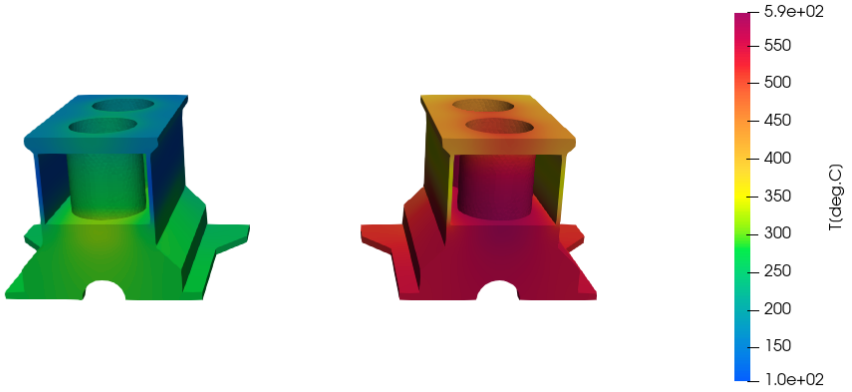


Figura 5.9: Corte na vista traseira.

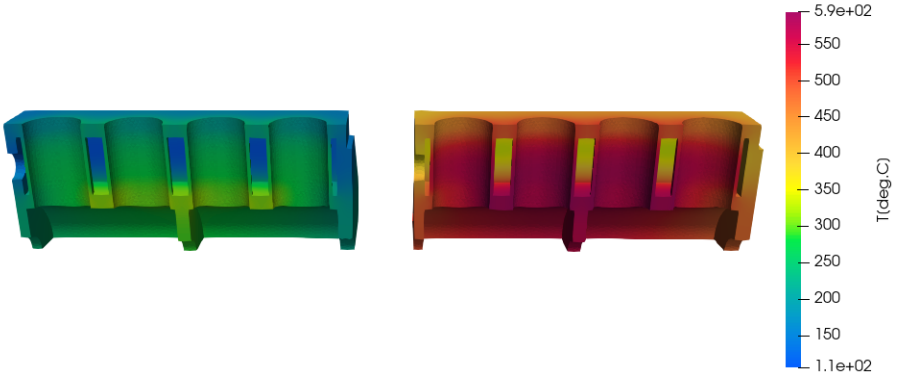


Figura 5.10: Corte na vista lateral.

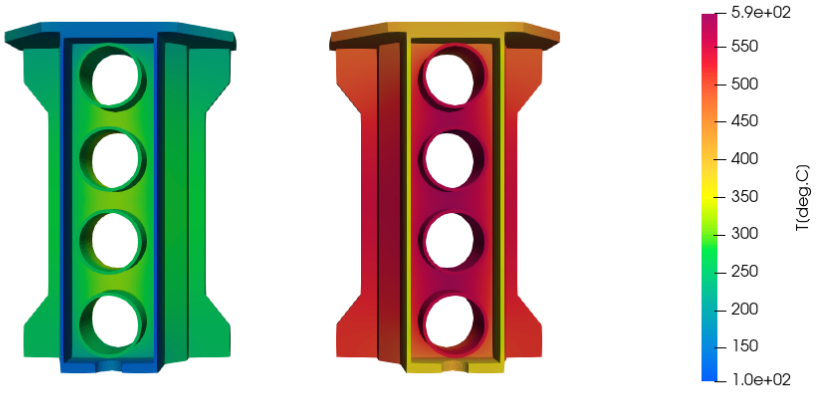


Figura 5.11: Corte na vista superior.

Observando as figuras 5.4 à 5.9, é evidente como a água é mais eficiente como um fluido de resfriamento dos cilindros. A escala do gráfico de temperatura é entre 100 °C e 600 °C, e em todas as figuras o bloco de mais alta temperatura é aquele onde o resfriamento ocorre por etilenoglicol. Observando a equação (3.12) nota-se que o coeficiente de transferência de calor para o fluido é proporcional à condutividade térmica do material, à uma potência do número de Reynolds e à uma potência do número de Prandtl. Da tabela 5.1 pode-se verificar que as massas específicas das substâncias são próximas uma da outra, a condutividade térmica da água é maior, enquanto Pr do etilenoglicol é maior. O fator que provavelmente é determinante na diferença entre os coeficientes de transferência de calor entre cada fluido é a viscosidade da água que é uma ordem de grandeza menor do que a do etilenoglicol, resultando em Re muito maior para a água. De fato, o Reynolds calculado para água foi da ordem de 10^4, enquanto para a outra substância foi da ordem de 10^3.

$T_{\text{max, água}}$	$T_{\text{max, etilenoglicol}}$
324,96 °C	594,18 °C

Tabela 5.2: Comparação entre as temperaturas máximas atingidas.

Em nenhum dos casos, de acordo com a tabela 5.2, em nenhum dos casos a temperatura ultrapassa o ponto de fusão do alumínio, de 660°C, de forma que em ambas as situações a integridade estrutural do bloco, a princípio estaria garantida. Entretanto, é importante levantar a questão de que temperaturas mais altas pode levar

à degradação de outros componentes que não estão sendo levados em consideração neste trabalho, como da vedação do pistão e do óleo lubrificante, por exemplo. Neste contexto, onde a combustão gera um certo fluxo de calor nas paredes do cilindro e o mesmo material para dois refrigerantes diferentes, é desejável que a rejeição do calor para o refrigerante seja a maior possível, resultando na menor temperatura no metal, de forma a evitar esse tipo de perda de integridade de outros componentes que compõem todo o sistema do motor, além do bloco, resultando numa menor temperatura da superfície externa e interna, que está em contato com outras estruturas.

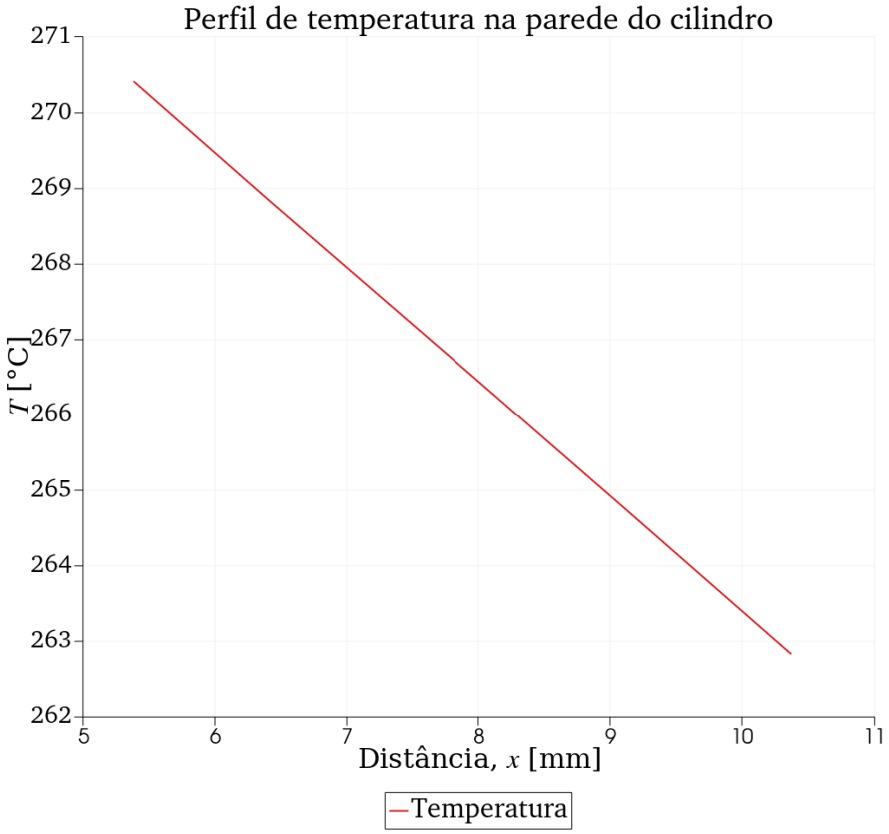


Figura 5.12: Perfil de temperatura na parede em volta do cilindro.

Traçando-se uma reta paralela ao eixo x, que sai da parede interna do cilindro até o refrigerante, a partir da função *Plot over the line* do Paraview, nas coordenadas z = 120 mm e y = -188,36 mm, é possível obsevar na figura 5.12 o mesmo perfil na parede da figura 2.4, que é o de um perfil linear, demonstrando como a simulação representa de forma confiável a física do problema, desde que sejam feitas hipóteses adequadas.

Capítulo 6

Conclusões

De acordo com os resultados mostrados no capítulo anterior, a solução obtida pelo método dos elementos finitos possui o comportamento físico esperado, lidando corretamente com as condições de contorno do tipo convectiva, produzindo um perfil de temperaturas na parede do cilindro de acordo com o esperando e, de forma geral, oferecendo resultados que condizem com a física do problema, de forma que todas as temperaturas estão entre aquelas presentes nas condições de contorno convectivas, ou seja, a do gás e do refrigerante. É possível também observar a difusão do calor das regiões de mais alta temperatura para regiões de temperatura inferiores nos cortes apresentados, mostrando a natureza de um processo difusivo independente do tempo, característico da equação de Laplace.

Os resultados obtidos mostram também como ferramentas acessíveis, como linguagens de alto-nível e programas de código aberto, podem ser utilizados na engenharia para solucionar problemas relevantes na indústria e na pesquisa. De fato, o problema proposto neste trabalho foi simplificado em vários níveis com relação à uma situação mais realista, uma geometria mais detalhada ou uma análise transiente do problema, mas de forma alguma deixa de ser suficientemente complexo, possuindo uma ordem de 10^5 elementos, com condições de contorno convectivas, foi implementado com linguagens cujo aprendizado é relativamente fácil, pois é muito próximo da sintaxe humana, e destacando a linguagem Julia, que através de suas bibliotecas padrão, tanto a montagem das matrizes quanto a solução do sistema linear foram feitas rapidamente em um computador pessoal. Mesmo para simulações com número de elementos de ordem de 10^6, como as realizadas para análise de

convergência de malha durante a verificação, não houve um longo tempo de espera.

Os resultados obtidos na etapa de validação e verificação demonstram a validade de um método como o de elementos finitos para solução de problemas matemáticos, porque é isso que ele se propõe a ser: um método de solução de equações diferenciais parciais complexas, que se originam a partir da modelagem de problemas reais. E para um problema com solução analítica conhecida, ele obteve uma solução precisa e acurada.

Considerando trabalhos posteriores, é dado como sugestão para aprimoramento do modelo físico, pode-se considerar a transferência de calor para o óleo, que é um aspecto que pode ser relevante à depender da análise à ser feita. Em segundo lugar, é possível analisar o problema em regime transiente, quando o interesse passa a ser de decisões de projeto, como a geometria ou materiais, mas quando é desejável realizar um estudo da eficiência e desempenho do motor ao longo de um número desejado de ciclos, desde à partida, ou em outros contextos. Também há a possibilidade uma análise de materiais heterogêneos, como no uso de cerâmicos na câmara de combustão para que ocorra menos rejeição de calor dos gases para o metal e depois para o refrigerante. Outra aspecto que pode ser modificado é com relação à geometria, onde podem ser representados sistemas de refrigeração mais complexos e inclusão de outras partes do motor, como do cabeçote.

Em último lugar, este trabalho levou ao desenvolvimento de um código confiável e funcional para o método de elementos finitos, que pode ser aprimorado para maior generalização, com algoritmos para discretização temporal e envolvendo condições de contorno do segundo tipo, que não foram implementadas neste problema, e também é um código que pode ser incorporado e adaptado em um programa maior, de um projeto acadêmico ou industrial mais complexo, podendo ser utilizados em estudos posteriores.

Referências Bibliográficas

HEYWOOD, J. *Internal Combustion Engine Fundamentals*. New York, NY, McGraw-Hill Education, 1988.

LIENHARD, IV, J. H., LIENHARD, V, J. H. *A Heat Transfer Textbook*. 5th ed. Cambridge, MA, Phlogiston Press, ago. 2020. Disponível em: <`http://ahtt.mit.edu`>. Version 5.10.

NITHIARASU, P., LEWIS, R. W., SEETHARAMU, K. N. *Fundamentals of the Finite Element Method for Heat and Mass Transfer*. 2 ed. Chichester, West Sussex, John Wiley & Sons, 2016.

REDDY, J. N. *An Introduction to the Finite Element Method*. 2 ed. New York, NY, McGraw-Hill Education, 1993.

FONSECA, L., OLMEDA, P., NOVELLA, R., et al. "Internal Combustion Engine Heat Transfer and Wall Temperature Modeling: An Overview", *Archives of Computational Methods in Engineering*, v. 27, pp. 1661–1679, 2020.

ÇENGEL, Y. A., GASHAR, A. J. *Heat and Mass Transfer: Fundamentals and Applications*. 5 ed. New York, NY, McGraw-Hill Education, 2015.

FISH, J., BELYTSCHKO, T. *A First Course in Finite Elements*. Chichester, West Sussex, John Wiley & Sons, 2007.

HUEBNER, K. H., DEWHIRST, D. L., SMITH, D. E., et al. *The Finite Element Method for Engineers*. 4 ed. New York, NY, John Wiley & Sons, 2001.

STONE, R. *Introduction to Internal Combustion Engines*. 3 ed. Basingstoke, Hampshire, Macmillan Press, 1999.

TAYLOR, C. F. *Internal Combustion Engine in Theory and Practice, Volume 1: Thermodynamics, Fluid Flow, Pèrformance*. Cambridge, MA, The MIT Press, 1985.

FERGUSON, C. R., KIRKPATRICK, A. T. *Internal Combustion Engines: Applied Thermosciences*. 3 ed. Chichester, West Sussex, John Wiley & Sons, 2016.

BOHAC, S. V., BAKER, D. M., ASSANI, D. N. "A Global Model for Steady State and Transient S.I. Engine Heat Transfer Studies", *SAE Technical Paper Series*, 1996.

ÖZIŞIK, M. N. *Transferência de Calor: Um Texto Básico*. Rio de Janeiro, RJ, Guanabara Koogan, 1990. ISBN: 9788527701600.

CARSLAW, H., JAEGER, J. *Conduction of Heat in Solids*. Oxford science publications. Oxford, UK, Clarendon Press, 1959. ISBN: 9780198533689.

BERGMAN, T. L., LAVINE, A. S., INCROPERA, F. P., et al. *Fundamentals of Heat and Mass Transfer*. 7 ed. New York, NY, John Wiley & Sons, 2011.

DOS ANJOS, G. R. *Computação Científica para Engenheiros*. Rio de Janeiro, RJ, 2022. Versão de 1 de novembro de 2022.

ÖZIŞIK, M. N. *Heat Conduction*. 2 ed. New York, NY, John Wiley & Sons, 1993. ISBN: 9780471532569.

ZIENKIEWICZ, O., TAYLOR, R., ZHU, J. *The Finite Element Method: Its Basis and Fundamentals*. 6 ed. Oxford, UK, Butterworth-Heinemann, 2005. ISBN: 978-0-7506-6431-8.

FINLAY, I. C., BOYLE, R. J., PIRAULL, J. P., et al. "Nucleate and Film Boiling of Engine Coolants Flowing in a Uniformly Heated Duct of Small Cross Section", *SAE Technical Paper Series*, 1987.

Apêndice A

Código Fonte

A.1 Leitor de Malhas - meshreader.py

```
1  import numpy as np
2  import meshio
3  # import matplotlib.pyplot as plt
4
5  # leitura de malha
6  # algoritmos por Gustavo R. Anjos(gustavo.rabello@coppe.ufrj
       .br)
7  def tri_meshreader(mshfile):
8      msh = meshio.read(mshfile)
9      X = msh.points[:,0]
10     Y = msh.points[:,1]
11     IEN = msh.cells[1].data
12     IENbound = msh.cells[0].data
13     IENboundTypeElem = list(msh.cell_data['gmsh:physical'
       ][0] - 1)
14     boundNames = list(msh.field_data.keys())
15     IENboundElem = [boundNames[elem] for elem in
       IENboundTypeElem]
16     npoints = len(X)
17     ne = IEN.shape[0]
```

```
18
19      # cria lista de nos do contorno
20      cc = np.unique(IENbound.reshape(IENbound.size))
21      ccName = [[] for i in range( len(cc) )]
22      for elem in range(0,len(IENbound)):
23          ccName[ IENbound[elem][0] ] = IENboundElem[elem]
24          ccName[ IENbound[elem][1] ] = IENboundElem[elem]
25
26      return X, Y, IEN.T, IENbound.T, IENboundElem, npoints,
    ne, cc, ccName, msh
27
28  def quad_meshreader(mshfile):
29      msh = meshio.read(mshfile)
30      msh = meshio.read('malha-quad.msh')
31      X = msh.points[:,0]
32      Y = msh.points[:,1]
33      IEN = msh.cells[1].data
34      IENbound = msh.cells[0].data
35      IENboundTypeElem = list(msh.cell_data['gmsh:physical'
    ][0] - 1)
36      boundNames = list(msh.field_data.keys())
37      IENboundElem = [boundNames[elem] for elem in
    IENboundTypeElem]
38      npoints = len(X)
39      ne = IEN.shape[0]
40      # cria lista de nos do contorno
41      cc = np.unique(IENbound.reshape(IENbound.size))
42      ccName = [[] for i in range( len(cc) )]
43      for elem in range(0,len(IENbound)):
44          ccName[ IENbound[elem][0] ] = IENboundElem[elem]
45          ccName[ IENbound[elem][1] ] = IENboundElem[elem]
46
```

```
47      return X, Y, IEN.T, IENbound.T, IENboundElem, npoints,
        ne, cc, ccName, msh
48
49  def tetra_meshreader(mshfile):
50      msh = meshio.read(mshfile)
51      X = msh.points[:,0]
52      Y = msh.points[:,1]
53      Z = msh.points[:,2]
54      IEN = msh.cells[1].data
55      IENbound = msh.cells[0].data
56      IENboundTypeElem = list(msh.cell_data['gmsh:physical'
        ][0] - 1)
57      boundNames = list(msh.field_data.keys())
58      IENboundElem = [boundNames[elem] for elem in
        IENboundTypeElem]
59      npoints = len(X)
60      ne = IEN.shape[0]
61      # cria lista de nos do contorno
62      cc = np.unique(IENbound.reshape(IENbound.size))
63      ccName = [[] for i in range( len(X) )]
64      for elem in range(0,len(IENbound)):
65          ccName[ IENbound[elem][0] ] = IENboundElem[elem]
66          ccName[ IENbound[elem][1] ] = IENboundElem[elem]
67          ccName[ IENbound[elem][2] ] = IENboundElem[elem]
68
69      return X, Y, Z, IEN.T, IENbound.T, IENboundElem, npoints
        , ne, cc, ccName, msh
70
71  def results_2D(X, Y, sol, name, msh, n):
72      point_data = {name : sol}
73      meshio.write_points_cells('sol-'+str(n)+'.vtk',
74      msh.points,
```

```
75      msh.cells,
76      point_data=point_data,)
77      return None
78
79  def results_3D(X, Y, Z, sol, name, msh, n):
80      point_data = {name : sol}
81      meshio.write_points_cells('sol-'+str(n)+'.vtk',
82      msh.points,
83      msh.cells,
84      point_data=point_data,)
85      return None
```

A.2 Matrizes elementares e montagem - mefmatrices.jl

```
1  module AssemblyMef
2
3      export tri_assemble_K, tri_assemble_M, line_convec_M,
       tetra_assemble_K, tetra_assemble_M, tri_convec_M
4
5      using LinearAlgebra
6      using SparseArrays
7
8      # 2D - Triangle element ------------------------
9
10     function line_length_2D(X, Y, IENbound, e)
11         v1, v2 = IENbound[:, e]
12
13         a = [X[v2] - X[v1], Y[v2] - Y[v1]]
14
15         l_elem = norm(a)
16
```

```
17          return l_elem
18      end
19
20      function tri_area(X, Y, IEN, e)
21          v1, v2, v3 = IEN[:, e]
22          return 0.5*det([1 X[v1] Y[v1]
23                          1 X[v2] Y[v2]
24                          1 X[v3] Y[v3]])
25      end
26
27      function tri_coef(X, Y, IEN, e)
28          v1, v2, v3 = IEN[:, e] # i, j, k
29
30          # ai = X[v2]*Y[v3] - X[v3]*Y[v2]
31          # aj = X[v3]*Y[v1] - X[v1]*Y[v3]
32          # ak = X[v1]*Y[v2] - X[v2]*Y[v1]
33          bi = Y[v2] - Y[v3]
34          bj = Y[v3] - Y[v1]
35          bk = Y[v1] - Y[v2]
36          ci = X[v3] - X[v2]
37          cj = X[v1] - X[v3]
38          ck = X[v2] - X[v1]
39          return bi, bj, bk, ci, cj, ck
40      end
41
42      # function shape_function(x, y, IEN, e)
43      #     v1 = IEN[e, 1]
44      #     v2 = IEN[e, 2]
45      #     v3 = IEN[e, 3]
46      #
47      #     ai, aj, ak, bi, bj, bk, ci, cj, ck  = tri_coef(X,
    Y, IEN, e)
```

```
48      #       area = tri_area(X, Y, IEN, e)
49      #
50      #       Ni = 0.5*(ai + bi*x + ci*y)/area
51      #       Nj = 0.5*(aj + bj*x + cj*y)/area
52      #       Nk = 0.5*(ak + bk*x + ck*y)/area
53      #
54      #       return Ni, Nj, Nk
55      # end
56
57      function tri_assemble_K(X, Y, IEN, npoints, ne, k)
58          iv = zeros(Int, (ne*3*3))
59          jv = zeros(Int, (ne*3*3))
60          k_data = zeros(Float64, (ne*3*3))
61          for e in 1:ne
62              area = tri_area(X, Y, IEN, e)
63              v1, v2, v3 = IEN[:, e]
64              bi, bj, bk, ci, cj, ck = tri_coef(X, Y, IEN, e)
65              iv[(e-1)*3*3+1:e*3*3] = [repeat([v1], 3); repeat
    ([v2], 3); repeat([v3], 3)]
66              jv[(e-1)*3*3+1:e*3*3] = repeat([v1, v2, v3], 3)
67
68              kx_elem = (0.25/area)*[bi*bi bi*bj bi*bk
69                                     bj*bi bj*bj bj*bk
70                                     bk*bi bk*bj bk*bk]
71              ky_elem = (0.25/area)*[ci*ci ci*cj ci*ck
72                                     cj*ci cj*cj cj*ck
73                                     ck*ci ck*cj ck*ck]
74              k_elem = k*(kx_elem + ky_elem)
75
76              k_data[(e-1)*3*3+1:e*3*3] = reshape(k_elem, (9,)
    )
77          end
```

```
78          K = sparse(iv, jv, k_data, npoints, npoints)
79          return K
80      end
81  sparse
82      function tri_assemble_M(X, Y, IEN, npoints, ne, rho, cp)
83          iv = zeros(Int, (ne*3*3))
84          jv = zeros(Int, (ne*3*3))
85          m_data = zeros(Float64, (ne*3*3))
86          # M = spzeros(npoints, npoints)
87          for e = 1:ne
88              area = tri_area(X, Y, IEN, e)
89              v1, v2, v3 = IEN[:, e]
90              iv[(e-1)*3*3+1:e*3*3] = [repeat([v1], 3); repeat
    ([v2], 3); repeat([v3], 3)]
91              jv[(e-1)*3*3+1:e*3*3] = repeat([v1, v2, v3], 3)
92              m_elem = rho*cp*(area/12.0)*[2.0  1.0  1.0
93                                           1.0  2.0  1.0
94                                           1.0  1.0  2.0]
95
96              m_data[(e-1)*3*3+1:e*3*3] = reshape(m_elem, (9,)
    )
97          end
98          M = sparse(iv, jv, m_data, npoints, npoints)
99          return M
100     end
101
102     function line_convec_M(X, Y, IENbound, npoints, nebound,
    h)
103         #M_convec = spzeros(npoints, npoints)
104         iv = zeros(Int, (nebound*2*2))
105         jv = zeros(Int, (nebound*2*2))
106         m_data = zeros(Float64, (nebound*2*2))
```

```
107         for e = 1:nebound
108             l_elem = line_length_2D(X, Y, IENbound, e)
109             v1, v2 = IENbound[:, e]
110             iv[(e-1)*2*2+1:e*2*2] = [repeat([v1], 2); repeat
    ([v2], 2)]
111             jv[(e-1)*2*2+1:e*2*2] = repeat([v1, v2], 2)
112
113             m_elem = (h*l_elem/6.0)*[2.0  1.0
114                                      1.0  2.0]
115
116             m_data[(e-1)*2*2+1:e*2*2] = reshape(m_elem, (4,)
    )
117             # for ilocal = 1:2
118             #     iglobal = IENbound[e, ilocal]
119             #     for jlocal = 1:2
120             #         jglobal = IENbound[e, jlocal]
121             #         M_convec[iglobal, jglobal] += m_elem[
    ilocal, jlocal]
122             #     end
123             # end
124         end
125         M_convec = sparse(iv, jv, m_data, npoints, npoints)
126         return M_convec
127     end
128
129     function line__flux_M(X, Y, IENbound, npoints, nebound)
130         # M_flux = spzeros(npoints, npoints)
131         iv = zeros(Int, (nebound*2*2))
132         jv = zeros(Int, (nebound*2*2))
133         m_data = zeros(Float64, (nebound*2*2))
134         for e = 1:nebound
135             l_elem = line_length_2D(X, Y, IENbound, e)
```

```
136             v1, v2 = IENbound[:, e]
137             iv[(e-1)*2*2+1:e*2*2] = [repeat([v1], 2); repeat
    ([v2], 2)]
138             jv[(e-1)*2*2+1:e*2*2] = repeat([v1, v2], 2)
139
140             m_elem = (l_elem/6.0)*[2.0  1.0
141                                    1.0  2.0]
142
143             m_data[(e-1)*2*2+1:e*2*2] = reshape(m_elem, (4,)
    )
144             # for ilocal = 1:2
145             #     iglobal = IENbound[e, ilocal]
146             #     for jlocal = 1:2
147             #         jglobal = IENbound[e, jlocal]
148             #         M_flux[iglobal, jglobal] += m_elem[
    ilocal, jlocal]
149             #     end
150             # end
151         end
152         M_flux = sparse(iv, jv, m_data, npoints, npoints)
153         return M_flux
154     end
155
156     # 3D - tetrahedron
157
158     function tri_area_trid(X, Y, Z, IENbound, e)
159         v1, v2, v3 = IENbound[:, e]
160
161         a = [X[v2] - X[v1], Y[v2] - Y[v1], Z[v2] - Z[v1]]
162         b = [X[v3] - X[v1], Y[v3] - Y[v1], Z[v3] - Z[v1]]
163
164         area = 0.5*norm(cross(a, b))
```

```
165
166         return area
167     end
168
169     function tetra_vol(X, Y, Z, IEN, e)
170         v1, v2, v3, v4 = IEN[:, e]
171
172         vol = (1.0/6.0)*det([1 X[v1] Y[v1] Z[v1]
173                              1 X[v2] Y[v2] Z[v2]
174                              1 X[v3] Y[v3] Z[v3]
175                              1 X[v4] Y[v4] Z[v4]])
176         return vol
177     end
178
179     function tetra_coef(X, Y, Z, IEN, e)
180         v1, v2, v3, v4 = IEN[:, e]
181         # ai = Z[v2]*(X[v3]*Y[v4] - X[v4]*Y[v3]) + X[v2]*(Y[
    v3]*Z[v4] - Y[v4]*Z[v3]) - Y[v2]*(X[v3]*Z[v4] - X[v4]*Z[
    v3])
182         # aj = X[v1]*(Y[v3]*Z[v4] - Y[v4]*Z[v3]) + Z[v1]*(X[
    v3]*Y[v4] - X[v4]*Y[v3]) - Y[v1]*(X[v3]*Z[v4] - X[v4]*Z[
    v3])
183         # ak = X[v1]*(Y[v2]*Z[v4] - Y[v4]*Z[v2]) + Z[v1]*(X[
    v2]*Y[v4] - X[v4]*Y[v2]) - Y[v1]*(X[v2]*Z[v4] - X[v4]*Z[
    v2])
184         # al = Z[v1]*(X[v2]*Y[v3] - X[v3]*Y[v2]) + X[v1]*(Y[
    v2]*Z[v3] - Y[v3]*Z[v2]) - Y[v1]*(X[v2]*Z[v3] - X[v3]*Z[
    v2])
185         bi = -(Y[v2] - Y[v4])*(Z[v3] - Z[v4]) + (Y[v3] - Y[
    v4])*(Z[v2] - Z[v4])
186         bj = -(Y[v3] - Y[v4])*(Z[v1] - Z[v4]) + (Y[v1] - Y[
    v4])*(Z[v3] - Z[v4])
```

```
187         bk = -(Y[v1] - Y[v4])*(Z[v2] - Z[v4]) + (Y[v2] - Y[
    v4])*(Z[v1] - Z[v4])
188         bl = -(bi + bj + bk)
189         ci = -(X[v3] - X[v4])*(Z[v2] - Z[v4]) + (X[v2] - X[
    v4])*(Z[v3] - Z[v4])
190         cj = -(X[v1] - X[v4])*(Z[v3] - Z[v4]) + (X[v3] - X[
    v4])*(Z[v1] - Z[v4])
191         ck = -(X[v2] - X[v4])*(Z[v1] - Z[v4]) + (X[v1] - X[
    v4])*(Z[v2] - Z[v4])
192         cl = -(ci + cj + ck)
193         di = -(X[v2] - X[v4])*(Y[v3] - Y[v4]) + (X[v3] - X[
    v4])*(Y[v2] - Y[v4])
194         dj = -(X[v3] - X[v4])*(Y[v1] - Y[v4]) + (X[v1] - X[
    v4])*(Y[v3] - Y[v4])
195         dk = -(X[v1] - X[v4])*(Y[v2] - Y[v4]) + (X[v2] - X[
    v4])*(Y[v1] - Y[v4])
196         dl = -(di + dj + dk)
197         return bi, bj, bk, bl, ci, cj, ck, cl, di, dj, dk,
    dl
198      end
199
200      # function tetra_shape_function(X, Y, Z, IEN, e)
201      #     ai, aj, ak, al, bi, bj, bk, bl, ci, cj, ck, cl, di
    , dj, dk, dl = tetra_coef(X, Y, Z, IEN, e)
202      #     vol = tetra_vol(X, Y, Z, IEN, e)
203      #     Ni = (1.0/(6.0*vol))*(ai + bi*X + ci*Y + di*Z)
204      #     Nj = (1.0/(6.0*vol))*(aj + bj*X + cj*Y + dj*Z)
205      #     Nk = (1.0/(6.0*vol))*(ak + bk*X + ck*Y + dk*Z)
206      #     Nl = (1.0/(6.0*vol))*(al + bl*X + cl*Y + dl*Z)
207      #     return Ni, Nj, Nk, Nl
208      # end
209
```

```
210     function tetra_assemble_K(X, Y, Z, IEN, npoints, ne, k)
211         # K = spzeros(npoints, npoints)
212         iv = zeros(Int, (ne*4*4))
213         jv = zeros(Int, (ne*4*4))
214         k_data = zeros(Float64, (ne*4*4))
215         for e = 1:ne
216             volume = tetra_vol(X, Y, Z, IEN, e)
217             v1, v2, v3, v4 = IEN[:, e]
218             iv[(e-1)*4*4+1:e*4*4] = [repeat([v1], 4); repeat
    ([v2], 4); repeat([v3], 4); repeat([v4], 4)]
219             jv[(e-1)*4*4+1:e*4*4] = repeat([v1, v2, v3, v4],
     4)
220             bi, bj, bk, bl, ci, cj, ck, cl, di, dj, dk, dl =
     tetra_coef(X, Y, Z, IEN, e)
221
222             kx_elem = (1.0/(36*volume))*[bi*bi bi*bj bi*bk
    bi*bl
223                                          bj*bi bj*bj bj*bk
    bj*bl
224                                          bk*bi bk*bj bk*bk
    bk*bl
225                                          bl*bi bl*bj bl*bk
    bl*bl]
226
227             ky_elem = (1.0/(36*volume))*[ci*ci ci*cj ci*ck
    ci*cl
228                                          cj*ci cj*cj cj*ck
    cj*cl
229                                          ck*ci ck*cj ck*ck
    ck*cl
230                                          cl*ci cl*cj cl*ck
    cl*cl]
```

```
231
232             kz_elem = (1.0/(36*volume))*[di*di di*dj di*dk
    di*dl
233                                          dj*di dj*dj dj*dk
    dj*dl
234                                          dk*di dk*dj dk*dk
    dk*dl
235                                          dl*di dl*dj dl*dk
    dl*dl]
236
237             k_elem = k*(kx_elem + ky_elem + kz_elem)
238
239             k_data[(e-1)*4*4+1:e*4*4] = reshape(k_elem,
    (16,))
240
241             #for ilocal = 1:4
242             #     iglobal = IEN[e, ilocal]
243             #     for jlocal = 1:4
244             #          jglobal = IEN[e, jlocal]
245             #          K[iglobal, jglobal] += k_elem[ilocal,
    jlocal]
246             #     end
247             #end
248
249         end
250         K = sparse(iv, jv, k_data, npoints, npoints)
251         return K
252     end
253
254     function tetra_assemble_M(X, Y, Z, IEN, npoints, ne,
    rho, cp)
255         # M = spzeros(npoints, npoints)
```

```
256         iv = zeros(Int, (ne*4*4))
257         jv = zeros(Int, (ne*4*4))
258         m_data = zeros(Float64, (ne*4*4))
259         for e = 1:ne
260             volume = tetra_vol(X, Y, Z, IEN, e)
261             v1, v2, v3, v4 = IEN[:, e]
262             iv[(e-1)*4*4+1:e*4*4] = [repeat([v1], 4); repeat
    ([v2], 4); repeat([v3], 4); repeat([v4], 4)]
263             jv[(e-1)*4*4+1:e*4*4] = repeat([v1, v2, v3, v4],
     4)
264
265             m_elem = rho*cp*(volume/20.0)*[2.0  1.0  1.0  1.0
266                                            1.0  2.0  1.0  1.0
267                                            1.0  1.0  2.0  1.0
268                                            1.0  1.0  1.0  2.0]
269             m_data[(e-1)*4*4+1:e*4*4] = reshape(m_elem,
    (16,))
270
271             # for ilocal = 1:4
272             #     iglobal = IEN[e, ilocal]
273             #     for jlocal = 1:4
274             #         jglobal = IEN[e, jlocal]
275             #         M[iglobal, jglobal] += m_elem[ilocal,
    jlocal]
276             #     end
277             # end
278         end
279         M = sparse(iv, jv, m_data, npoints, npoints)
280         return M
281     end
282
283     function tri_convec_M(X, Y, Z, IENbound, npoints,
```

```
    nebound, h)
284         # M_convec = spzeros(npoints, npoints)
285         iv = zeros(Int, (nebound*3*3))
286         jv = zeros(Int, (nebound*3*3))
287         m_data = zeros(Float64, (nebound*3*3))
288         for e = 1:nebound
289             area = tri_area_trid(X, Y, Z, IENbound, e)
290             v1, v2, v3 = IENbound[:, e]
291             iv[(e-1)*3*3+1:e*3*3] = [repeat([v1], 3); repeat
    ([v2], 3); repeat([v3], 3)]
292             jv[(e-1)*3*3+1:e*3*3] = repeat([v1, v2, v3], 3)
293
294             m_elem = (h*area/12.0)*[2.0  1.0  1.0
295                                     1.0  2.0  1.0
296                                     1.0  1.0  2.0]
297             m_data[(e-1)*3*3+1:e*3*3] = reshape(m_elem, (9,)
    )
298             # for ilocal = 1:3
299             #     iglobal = IENbound[e, ilocal]
300             #     for jlocal = 1:3
301             #         jglobal = IENbound[e, jlocal]
302             #         M_convec[iglobal, jglobal] += m_elem[
    ilocal, jlocal]
303             #     end
304             # end
305         end
306         M_convec = sparse(iv, jv, m_data, npoints, npoints)
307         return M_convec
308     end
309
310     function tri_flux_M(X, Y, Z, IENbound, npoints, nebound)
311         # M_flux = spzeros(npoints, npoints)
```

```
312         iv = zeros(Int, (nebound*3*3))
313         jv = zeros(Int, (nebound*3*3))
314         m_data = zeros(Float64, (nebound*3*3))
315         for e = 1:nebound
316             area = tri_area_trid(X, Y, Z, IENbound, e)
317             v1, v2, v3 = IENbound[:, e]
318             iv[(e-1)*3*3+1:e*3*3] = [repeat([v1], 3); repeat
    ([v2], 3); repeat([v3], 3)]
319             jv[(e-1)*3*3+1:e*3*3] = repeat([v1, v2, v3], 3)
320
321             m_elem = (area/12.0)*[2.0  1.0  1.0
322                                   1.0  2.0  1.0
323                                   1.0  1.0  2.0]
324             m_elem = reshape(m_elem, (9,))
325             m_data[(e-1)*3*3+1:e*3*3] = m_elem
326             # for ilocal = 1:3
327             #     iglobal = IENbound[e, ilocal]
328             #     for jlocal = 1:3
329             #         jglobal = IENbound[e, jlocal]
330             #         M_convec[iglobal, jglobal] += m_elem[
    ilocal, jlocal]
331             #     end
332             # end
333         end
334         M_flux = sparse(iv, jv, m_data, npoints, npoints)
335         return M_flux
336     end
337 end
```

A.3 Condições de contorno - boundary_conditions.jl

```
module BoundaryConditions

export boundaryIEN, dirichlet_bc!

function boundaryIEN(name, namelist, IENbound)
    neb = 0
    for n in namelist
        if n == name
            neb += 1
        end
    end
    IENb = zeros(Int64, size(IENbound, 1), neb)
    i = 1
    for j = 1:lastindex(IENbound, 2)
        if namelist[j] == name
            IENb[:, i] = IENbound[:, j]
            i += 1
        end
    end
    return IENb, neb
end

function dirichlet_bc!(K, f, T_p, cc, ccName, names)
    for i = cc
        for name = names
            if ccName[i] == name
                # moving column i to the right handside
                f += - K[:, i]*T_p
                K[:, i] .= 0.0
                # imposing trivial equation at line i
                K[i, :] .= 0.0
                K[i, i] = 1.0
```

```
33                      f[i] = T_p
34                  end
35              end
36          end
37          return K, f
38  end
39
40  end
```

A.4 Notebook para Simulação

Adaptado para script em Julia.

```
1  using PyCall, LinearAlgebra, SparseArrays, DelimitedFiles,
       Trapz
2
3  include("../mefmatrices.jl")
4  include("../boundary_conditions.jl")
5
6  pushfirst!(pyimport("sys")."path", "")
7  py"""
8  import sys
9  sys.path.append('../')
10 import meshreader
11 """
12
13 # Propriedades fisicas
14 k = 231.0;  # condutividade termica [W/m deg.C]
15
16 # gas data
17 T_g = vec(readdlm("../tgas.csv"));
18 h_g = vec(readdlm("../htcoeff.csv"));
19 theta = collect(range(0,2*pi,length(h_g)));
20 I1 = trapz(theta, h_g);
```

```
21 I2 = trapz(theta, T_g.*h_g);
22 hg_avg = I1/(4*pi);
23 Tg_avg = I2/(4*pi*hg_avg) + 273.05;
24 Tg_avg, hg_avg
25
26 # dados do refrigerante
27 T_ci = 40.0;
28 T_co = 80.0;
29 T_c = (T_ci + T_co)/2
30 # agua
31 k_a = 0.651;
32 rho_a = 985.46;
33 nu_a = 0.478e-6;
34 mu_a = rho_a*nu_a;
35 Pr_a = 3.02;
36
37 # etilenoglicol
38 k_e = 0.260;
39 rho_e = 1087.66;
40 nu_e = 4.75e-6;
41 mu_e = rho_e*nu_e;
42 Pr_e = 51;
43
44 u = 0.65
45
46 c0 = 0.348
47 n = 0.592
48 c1 = 0.8
49 D = 95.0e-3
50 A_min = 90.0*(140.0 - 95.0)
51 A_i = pi*(40)^2 / 4
52
```

```
53 G_amax =  rho_a*u*(A_i/A_min)
54 G_emax =  rho_e*u*(A_i/A_min)
55 Re_a = G_amax*D/mu_a
56 Re_e = G_emax*D/mu_e
57 println(Re_a, " ", Re_e)
58
59 h_a = c1*1.13*c0*(Re_a^n)*(Pr_a^(1/3))*k_a/D;
60 h_e = c1*1.13*c0*(Re_e^n)*(Pr_e^(1/3))*k_e/D;
61 println(h_a, " ", h_e)
62
63 # Mesh
64 X, Y, Z, IEN, IENbound, IENboundElem, npoints, ne, cc,
       ccName, msh = py"meshreader.tetra_meshreader"("block_v2.
       msh")
65 # mm para m
66 X = X.*10^-3
67 Y = Y.*10^-3
68 Z = Z.*10^-3
69 IEN .+= 1;
70 IENbound .+= 1;
71 cc .+= 1;
72
73 @showtime K_cond = AssemblyMef.tetra_assemble_K(X, Y, Z, IEN
       , npoints, ne, k);
74
75 # C.C.
76 IEN_gas, ne_gas = BoundaryConditions.boundaryIEN("gas",
       IENboundElem, IENbound);
77 IEN_cool, ne_cool = BoundaryConditions.boundaryIEN("cool",
       IENboundElem, IENbound);
78
```

```
79 @showtime K_a = AssemblyMef.tri_convec_M(X, Y, Z, IEN_cool,
       npoints, ne_cool, h_a);
80 @showtime K_e = AssemblyMef.tri_convec_M(X, Y, Z, IEN_cool,
       npoints, ne_cool, h_e);
81 @showtime K_g = AssemblyMef.tri_convec_M(X, Y, Z, IEN_gas,
       npoints, ne_gas, hg_avg);
82
83 # matrizes
84 K1 = K_cond + K_a + K_g;
85 K2 = K_cond + K_e + K_g;
86
87 f1 = K_a*(ones(npoints)*T_c) + K_g*(ones(npoints)*Tg_avg);
88 f2 = K_e*(ones(npoints)*T_c) + K_g*(ones(npoints)*Tg_avg);
89
90 @showtime T1 = K1\f1;
91 @showtime T2 = K2\f2;
92
93 py"meshreader.results_3D"(X, Y, Z, T1, "T[deg.C]", msh, "
       case1")
94 py"meshreader.results_3D"(X, Y, Z, T2, "T[deg.C]", msh, "
       case2")
95
96 println(maximum(T1), ", ", maximum(T2))
```

Apêndice B

Desenhos

Todas as medidas estão em milímetros, e o bloco do motor possui um plano de simetria paralelo à vista lateral. Vale ressaltar que todos os cilindros são equidistantes entre si, e com relação àqueles cujo diâmetro foi mostrado, também são de mesmo diâmetro, como por exemplo, no corte que aparece na figura B.4, existem cilindros concêntricos, todos os cilindros menores tem o mesmo diâmetro, mostrado em apenas um deles.

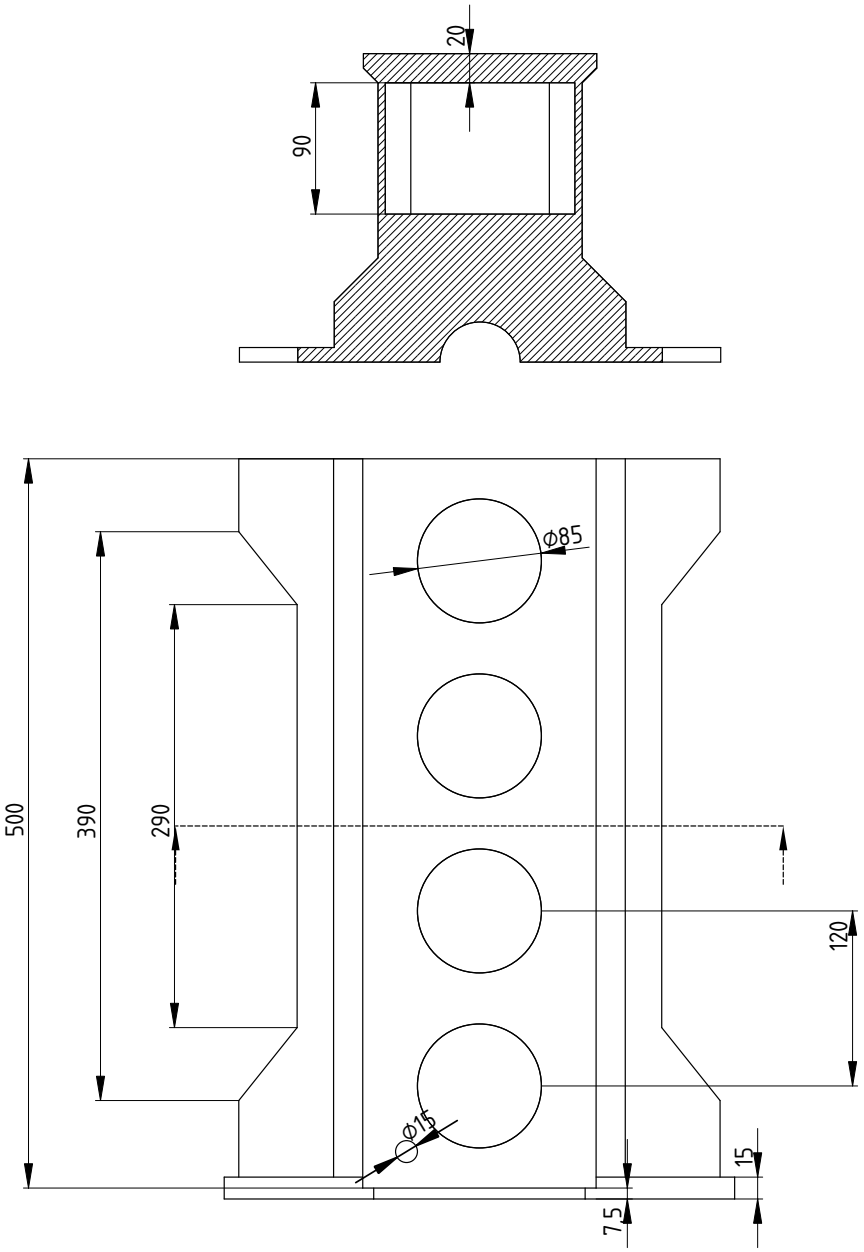


Figura B.1: Vista superior com corte da vista traseira.

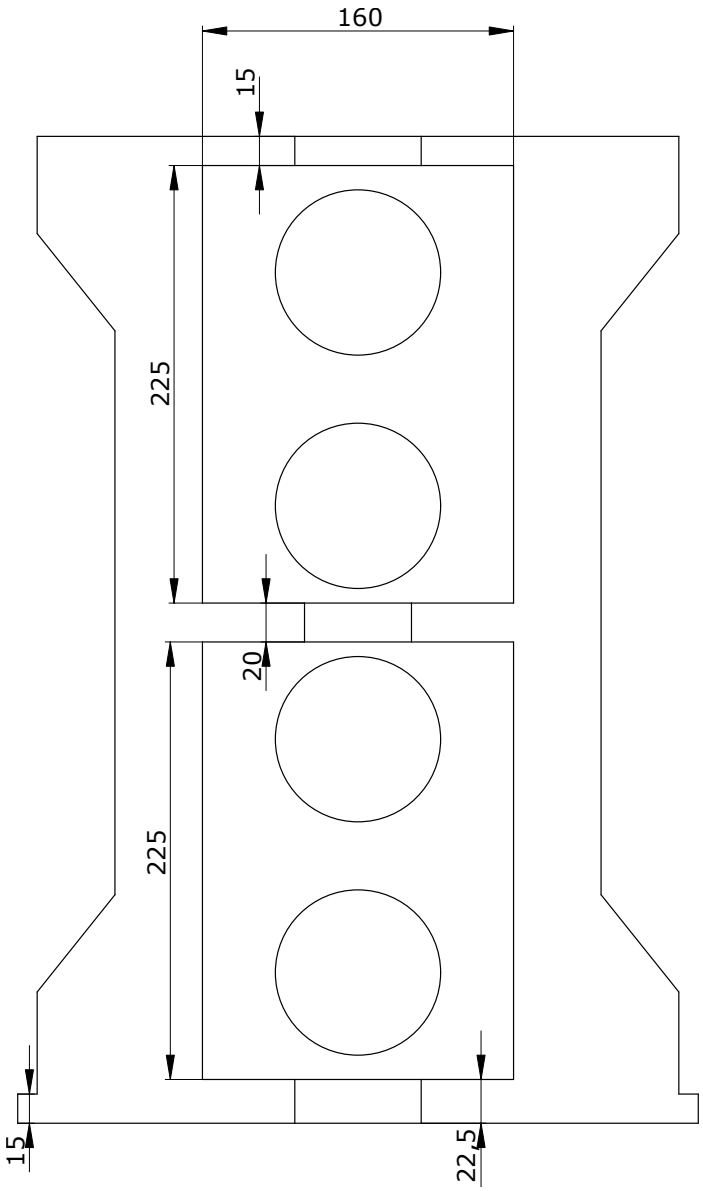


Figura B.2: Vista inferior.

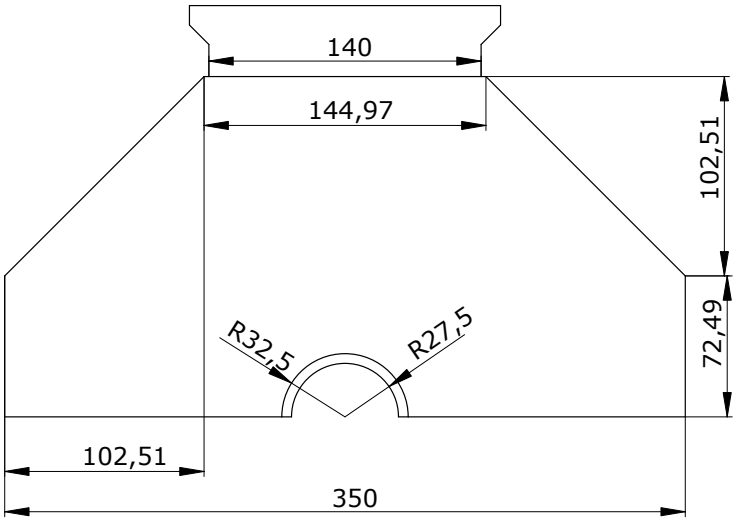


Figura B.3: Vista traseira.

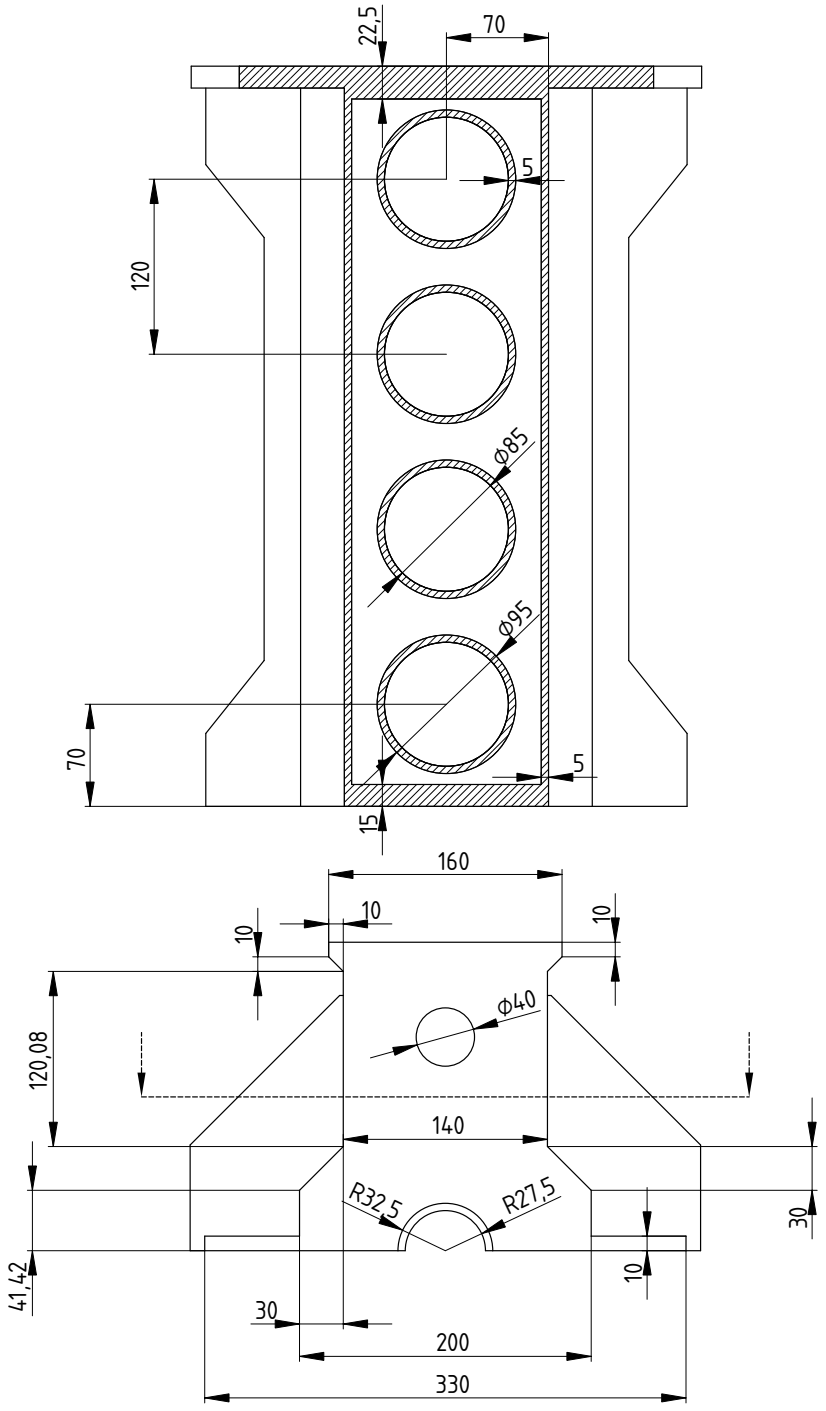


Figura B.4: Vista frontal com um corte da vista superior.