Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Curso de Engenharia de Energia

DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO DA EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO DE VOLUMES DE CONTROLE BASEADO EM ELEMENTOS

Autor: Pedro Paulo Dunice van Els
Orientador: Fabio Alfaia da Cunha

Brasília, DF
2016

PEDRO PAULO DUNICE VAN ELS

DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO DA EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO DE VOLUMES DE CONTROLE BASEADO EM ELEMENTOS

Monografia submetida ao curso de graduação em Engenharia de Energia da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharelem Engenharia de Energia.

Orientador:Prof. Dr. Fábio Alfaia da Cunha.

Brasília, DF
2016

DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO DA EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO DE VOLUMES DE CONTROLE BASEADO EM ELEMENTOS

Pedro Paulo Dunice van Els

Monografia submetida como requisito parcial para obtenção do Título de Bacharel em Engenharia de Energia da Faculdade UnB Gama - FGA, da Universidade de Brasília, em 23/11/2016 apresentada e aprovada pela banca examinadora abaixo assinada:

Prof. Doutor: Fábio Alfaia da Cunha, UnB/ FGA
Orientador

Prof. Doutor: Fábio Cordeiro de Lisboa, UnB/ FGA
Membro Convidado

Prof. Doutor: Luciano Gonçalves Noleto, UnB/ FGA
Membro Convidado

Brasília, DF
2016

BDM
Biblioteca Digital de Monografias

IDENTIFICAÇÃO

Autor: PEDRO PAULO DUNICE VAN ELS			
RG: 272874-2	**CPF:** 025857591-30	**E-mail:** PEDRO.DUNICE@GMAIL.COM	
Telefone: (61) 3562 7436		**Celular:** (61) 98587076	
Título: DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO DA EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO DE VOLUMES DE CONTROLE BASEADO EM ELEMENTOS			
Palavras-chave: MÉTODOS NUMÉRICOS; TRANSFERÊNCIA DE CALOR; CVFEM			
Departamento: FGA		**Curso:** ENGENHARIA DE ENERGIA	
Data de apresentação: 05/12/2016			

INFORMAÇÃO DE ACESSO AO DOCUMENTO:

Liberação para publicação: (X) Total ( ) Parcial*

Em caso de publicação parcial, especifique os capítulos a serem retidos: _____

Havendo concordância com a publicação eletrônica, torna-se imprescindível o envio do arquivo em formato digital da monografia **completa.**

*A restrição poderá ser mantida por até um ano a partir da data de autorização da publicação. A extensão deste prazo suscita justificativa junto a UnB-BCE. O resumo e os metadados ficarão sempre disponibilizados.

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

Os referidos autores:

a) Declaram que o documento entregue é seu trabalho original, e que detém o direito de conceder os direitos contidos nesta licença. Declaram também que a entrega do documento não infringe, tanto quanto lhe é possível saber, os direitos de qualquer outra pessoa ou entidade.

b) Se o documento entregue contém material do qual não detêm os direitos de autor, declaram que obteve autorização do detentor dos direitos de autor para conceder à Universidade de Brasília os direitos requeridos por esta licença, e que esse material cujos direitos são de terceiros está claramente identificado e reconhecido no texto ou conteúdo do documento entregue.

Se o documento entregue é baseado em trabalho financiado ou apoiado por outra instituição que não a Universidade de Brasília, declaram que cumpriram quaisquer obrigações exigidas pelo respectivo contrato ou acordo.

Declaro estar ciente de que as mídias contendo o documento serão descartadas pela BCE após sua inclusão na Biblioteca Digital de Monografias.

TERMO DE AUTORIZAÇÃO

Na qualidade de titular dos direitos de autor do conteúdo supracitado, autorizo a Biblioteca Central da Universidade de Brasília a disponibilizar a obra, gratuitamente, de acordo com a licença pública *Creative Commmons* Licença 3.0 Unported por mim declarada sob as seguintes condições:

Permitir uso comercial de sua obra?
(X) Sim ( ) Não

Permitir modificações em sua obra?
( ) Sim
( ) Sim, contanto que outros compartilhem pela mesma licença
(X) Não

A obra continua protegida por Direito Autoral e/ou por outras leis aplicáveis. Qualquer uso da obra que não o autorizado sob esta licença ou pela legislação autoral é proibido.

Brasília, 06 / 02 / 2017
Local Data

Pedro Paulo Dunice
Assinatura do Autor

RESUMO

O método dos volumes de controle baseado em elementos, do inglês, *Control Volume Finite Element Method* (CVFEM) é um importante ferramenta matemática que vem ganhando muitos adeptos nos últimos anos com o crescimento do poder de processamento dos computadores modernos. Esse método é capaz de obter soluções para diversos problemas de engenharia e vem sendo largamente utilizado na indústria por meio de programas comerciais.

Procura-se aqui desenvolver uma visão geral sobre o CVFEM e a sua aplicação em problemas de transferência de calor. O objetivo é fazer com que esse método não seja uma total “caixa preta” para um futuro usuário de um programa comercial.

Para atingir tal objetivo serão testados neste trabalho problemas no contexto da transferência de calor em regime permanente e em regime transiente. Os problemas propostos são de geometria 3D, e devido à forma genérica de como o método é desenvolvido, geometrias complexas podem ser trabalhadas. O programa desenvolvido suporta diferentes condições de contorno: temperatura definida, fluxo de calor definido e fluxo de calor com coeficiente de convecção definido. Neste programa podem ser utilizados diferentes materiais para análise de transferência de calor conjugada. Para tal fim foram utilizados o Matlab (para desenvolver o código computacional numérico) e o Gambit (para gerar a malha computacional).

De forma a verificar a qualidade dos resultados é primeiramente observado o efeito de convergência do código computacional com o aumento do número de nós da malha e então a diferença entre os resultados numéricos e os resultados analíticos. O código computacional conseguiu desempenhar as funcionalidades adicionadas em vários testes desenvolvidos ao longe deste trabalho, para todos os caso o erro em comparação com os resultados analíticos foi sempre inferior a 1% demonstrando assim a confiabilidade e maleabilidade do método CVFEM, .

ABSTRACT

The Control Volume Finite Element Method (CVFEM) is an important mathematical tool that its popularity is increasing in last few years at the same pace that the modern computer processing capacity increases. This method is capable of obtaining solutions to a diversity of engineering problems and is being broadly applied in the industry via commercial software.

This paper aims to develop a generic vision about the CVFEM and it's potential applications in heat transfer problems. The objective is to take this "black box" sign out for a future user of a commercial computational fluid dynamics software.

This paper covers steady state and transient heat transfer problems. All the cases are of simple 3D geometry, and due to the way the method is accessed complex geometries can be applied. The method also covers many different boundary conditions: Fixed temperature, fixed heat flux, fixed convection coefficient and heat flux through different materials. In order to do so, it uses the mathematical tool Matlab (solver) and the drawing tool Gambit (mesh generator).

In order to verify the results quality it was firstly analyzed the convergence effect of the computational script with the increase of point on the computational mesh, then the difference between the analytical result and the calculated ones. The developed script was able to access all the functionalities added thought a variety of tests, where the global error was always under the 1% limit, consolidating, this way, the malleability and reliability of the CVFEM method.

SUMÁRIO

LISTA DE SÍMBOLOS E VARIÁVEIS

SÍMBOLOS / ÍNDICES / EXPOENTES / VARIÁVEIS

k	Condutividade térmica [W/mK]
t	Tempo [s]
T	Temperatura [K]
x, y, z	Coordenadas cartesianas [m]
ϕ	Variável discretizada
Γ	Coeficiente de difusão
ρ	Densidade [kg/m³]
a,b,c,d	coeficientes da função de forma/ pontos médios aresta elemento
q, r ,s,t	Baricentro faces dos elementos
V	Volume[m³]
A	Área [m²]
S	Superfície
N	Função de forma
Q	Fluxo de calor [W/m²]
c	calor especifico [J/Kg K]

SÍMBOLOS MATEMÁTICOS

δ	derivada parcial
d	derivada
$\int$	Integral aberta
$\oint$	Integral de superfície fechada
$\sum$	Somatório
∇	Operador diferencial (gradiente ou divergente)
Δ	Operador Laplaciano
n	vetor normal
$\hat{\imath}, \hat{\jmath}, \hat{k}$	Direção componentes de vetor
Ω	Contorno do sistema

LISTA DE FIGURAS

1. INTRODUÇÃO

Os métodos numéricos, os quais são muito antigos, já existiam no início do século XX. Porém nos últimos anos esses vem ganhando cada vez mais espaço no mercado. Nos últimos anos temos presenciado um aumento da capacidade de processamento, armazenamento dos computadores modernos, sejam esses os supercomputadores que trabalham para grandes empresas e governos, ou também os mais simples computadores pessoais (Versteeg, 2007). Com esse aumento mais do que exponencial da facilidade de acesso aos computadores as simulações numéricas vêm sendo cada vez mais empregadas em grandes e pequenos projetos de engenharia, sempre paralelamente aos métodos experimentais e analíticos.

Estamos presenciando nessas primeiras décadas do século XXI o uso comercial dos primeiros programas computacionais de engenharia. Esses abordam problemas estruturais, fluidodinâmica, eletromagnetismo ou mesmo termodinâmica. Porém esses programas comerciais funcionam como uma caixa-preta para o usuário, o qual muitas vezes não sabe o que está realmente acontecendo, principalmente para programas de uso em computadores “pessoais”. Muitos softwares tais como Fluent®, Abaqus® entre centenas de outros disponíveis não oferecem essa oportunidade do usuário de interpretar e modificar o código computacional que analisa a geometria da malha computacional e resolve o problema numérico, sendo os programas com código abertos de uso mais acadêmico tais como o OpenFoam.

Saber o que está realmente acontecendo em um programa CFD (*Computacional Fluid Dynamics*) além de ser academicamente interessante, pode auxiliar na tomada de decisões ao afirmar a qualidade e validade dos resultados, melhorar a qualidade dos resultados e otimizar o tempo de processamento do projeto.

Com este cenário em mente este trabalho tem como objetivo desenvolver um código computacional fundamentado no método de volumes de controle baseado em elementos finitos utilizando o software Matlab. Este método desenvolvido primeiramente por Baliga em 1980 será aplicado no escopo da transferência de calor. Serão utilizadas malhas computacionais

tridimensionais de elementos finitos tetraédricos geradas pelo software Gambit exportadas em formato genérico (*.neu) para solucionar problemas no regime permanente e transiente, com condição de contorno de temperatura prescrita ou fluxo. Por último a transferência de calor conjugada também é avaliada. Em todos os casos os resultados numéricos são comparados aos resultados analíticos de forma verificar a validade destes

As atividades propostas serão desenvolvidas em diferentes seções. Nas primeiras seções será explicado o método CVFEM e desenvolvidas a equações das funções de forma e de interpolação. Na seção seguinte os problemas em regime permanente e transiente são abordados e a aplicação da condição de contorno de temperatura prescrita. Na seqüência problemas de condição de fluxo prescrito e com coeficiente de convecção fixo. Por último problemas de transferência de calor conjugada.

2. OBJETIVOS

O objetivo deste trabalho é desenvolver um programa, através do software MatLab, capaz de resolver numericamente problemas de transferência de calor transiente e em regime permanente. O método no qual este programa se baseia é o Método de Volumes de Controle Baseado em Elementos (CVFEM).

Os objetivos específicos são:

I. Estudar os fundamentos de transferência de calor;
II. Realizar uma revisão bibliográfica sobre os métodos numéricos mais utilizados;
III. Apresentar todos os passos necessários para a discretização da equação de transferência de calor tridimensional transiente através do método de volumes de controle baseado em elementos;
IV. Entender dinâmica do programa *Gambit*®, utilizado para gerar malhas tridimensionais não estruturadas;
V. Programar um código computacional difusivo 3D transiente;
VI. Programar condições de contorno de fluxo calor e de temperatura prescrita.
VII. Programar condição de contorno de fluxo com coeficiente de temperatura definido;
VIII. Programar transferência de calor conjugada;
IX. Validação dos códigos gerados em soluções analíticas e soluções encontradas na literatura;

3. TRANSFERÊNCIA DE CALOR

A transferência de calor é a ciência da energia. Controlar a transformação da energia em suas diversas formas sempre foi algo de interesse do ser humano. Hoje a ciência e a tecnologia desenvolvidas ao longo dos séculos fazem possível que tenhamos controle sobre essas transformações. Como observado a energia obedece a três leis fundamentais: O primeiro desses princípios é sobre a conservação da energia: “a energia não pode ser criada nem destruída, somente modificada de uma forma para a outra” (Kreith, 1992). A segunda lei que deve ser respeitada é sobre o fluxo de energia, onde segundo Kreith “não é possível existir um processo cujo único resultado seja a transferência líquida de calor de uma região de temperatura mais baixa para um de temperatura mais alta” e por último a lei zero define que “se dois corpos estiverem em equilíbrio térmico com um terceiro, estarão em equilíbrio térmico entre si.”(Çengel, 2003).

Como definido anteriormente a partir da observação direta da segunda lei temos que o calor deve fluir sempre do meio mais quente para um meio mais frio, em outras palavras se existir um gradiente de temperatura em um meio sólido o calor irá fluir para a região de temperatura mais alta para a de mais temperatura mais baixa. Como o objetivo deste trabalho é analisar como se dá a condução de calor em um meio sólido tudo o que for desenvolvido aqui segue essas três leis.

O escopo que será aqui abordado abrange quatro problemas clássicos de transferência de calor. Primeiramente os sólidos com temperatura definidas. Em seguida sólidos com fluxo de calor definido na fronteira. Sólidos com coeficientes de convecção definidos na fronteira. E por último problemas de fluxo de calor conjugado.Este escopo abrange problemas reais de equilíbrio térmico, isolamento de fornos, trocadores de calor, reservatórios térmicos e muitos outros.

4. EQUAÇÃO DA ENERGIA

As equações que regem a fluido dinâmica, termodinâmica, eletromagnetismo, transferência de calor e outros fenômenos físicos e químicos podem ser representadas de forma matemática, geralmente fazendo uso do Teorema do Transporte de Reynolds: (White, 1998)

$$\frac{\delta \rho \emptyset}{\delta t} + \nabla(\rho\, v\emptyset) - \nabla(k\nabla\emptyset) - S_{\emptyset} = 0 \qquad (4.1)$$

A equação 4.1 foi formulada a partir de um processo evolutivo no estudo da transferência de calor. Por ser tão abrangente a variável ϕ é posta da forma explicita e então especificada para cada caso.

O objetivo aqui é resumir a equação 4.1 para resolver problemas de transferência de calor em regime permanente em materiais sólidos e posteriormente abordar problemas transientes. Logo o termo correspondente à derivada temporal será nulo. Será também considerado que: por ser um sólido as partículas do domínio não se movem em relação ao próprio domínio, como ocorreria em um gás, por exemplo. Para isso a taxa de variação da velocidade no espaço é nula, ou seja, o segundo termo, o advectivo é nulo. E por último caso tenha alguma fonte de ϕ, ou neste caso, uma fonte de calor, esta deverá ser apresentada como uma condição de contorno e sendo assim na equação geral o termo de fonte é nulo também. Após estas simplificações obtemos a equação 4.2, que é a da condução de calor tridimensional (Çengel, 2003).

$$\nabla(\Gamma\nabla\emptyset) = 0 \qquad (4.2)$$

O coeficiente de difusão será constante também, ou seja, não se modifica ao longo do material sólido, somente o Laplaciano da variável ϕ multiplicado pelo coeficiente de difusão.

$$\Gamma\Delta\emptyset = 0 \qquad (4.3)$$

Essa equação será utilizada para a 1ª etapa deste trabalho. Em seguida o termo transiente que foi desconsiderado da equação 4.1 será adicionado à equação 4.3 e assim teremos a equação 4.4 que é relativa ao mesmo tipo de problema de condução de calor em 3D, porém com a possibilidade de analisar o problema proposto até que este alcance a estabilidade térmica.

$$\frac{\delta\rho\emptyset}{\delta t} + \Gamma\Delta\emptyset = 0 \quad (4.4)$$

Considerando o coeficiente de difusão igual ao da equação 4.5 temos que a variável genérica ϕ é correspondente à temperatura. (Maliska,2010)

$$\Gamma = \frac{k}{c} \quad (4.5)$$

Respeitando o balanço das unidades e grandezas físicas, pelos métodos já bem desenvolvidos na literatura, para que a variável genérica ϕ represente a temperatura o coeficiente de difusão Γ deve ser igual ao coeficiente de difusão térmica do material divido pela condutividade térmica do material. Resultado na equação 4.5. (Maliska,2010)

$$\frac{\delta\rho c\emptyset}{\delta t} + k\Delta\emptyset = 0 \quad (4.6)$$

A equação 4.6 deve vir associada às condições de contorno e iniciais do problema. As condições de contorno podem ser de dois tipos: Neumman, as quais definem o valor da derivada nas fronteiras do domínio Equação 4.7, ou condições de Dirichlet, as quais especificam o valor da função no contorno Equação 4.8. E por último, também são necessárias as condições iniciais, as quais definem o estado do domínio no tempo inicial como na Equação 4.9. (Patankar,1941)

$$\nabla\emptyset|_{\Omega} = \mathrm{Q} \quad (4.7)$$
$$\emptyset(\Omega, \mathrm{t}) = \mathrm{T} \quad (4.8)$$
$$\emptyset(\mathrm{x}, \mathrm{y}, \mathrm{z}, 0) = \mathrm{T}_0 \quad (4.9)$$

5. SIMULAÇÃO

5.1 - CONCEITOS BÁSICOS

É importante antes de prosseguir definir alguns conceitos básicos, expressões e palavras brasileiras ou estrangeiras que estão sempre na área da simulação numérica computacional. Segue abaixo uma pequena lista.

CFD-Computacional Fluid Dynamics – Dinâmica dos fluidos computacional, ou fluidodinâmica computacional.

Código computacional – Processo que aplica o princípio de transformar as equações diferenciais em equações de diferenças, com o objetivo de solucioná-las. O código conhecido como "*solver"* na literatura estrangeira gera os arquivos de resultados que são passado para a etapa de pós-processamento.

Pré-processamento – Nesta etapa são obtidas as informações geométricas do objeto a ser trabalhado. Inicia-se com a definição da forma dos elementos, seguido pela geração da malha e a interpretação da malha segundo o método CVFEM no início do código computacional

Pós-processamento – O pós-processamento é como o usuário irá visualizar os resultados que o código computacional gerou. Essa etapa consiste em gerar tabelas, gráficos, cortes de seção e até animações.

Elemento – Figura geométrica única dentro da malha. Composto por 4 nós que ligados formam um tetraedro. Os elementos estão distribuídos por todo o volume do material.

Marcha no tempo – Processo de avançar na linha do tempo partindo de um instante t para um instante $t + \Delta t$.

Verificação – Verificar se as equações foram resolvidas corretamente.

Discretizar – Particionar um todo em partes com menor complexidade,

Workbench – Bancada de trabalho – Tela em um software que mostra as ferramentas com as quais o usuário pode interagir.

5.2– MÉTODOS NUMÉRICOS

A equação 4.6 necessita que complexos métodos matemáticos sejam utilizados para que uma solução analítica seja obtida a partir dela. Se o problema no qual equação for aplicada envolver geometrias complexas, múltiplas condições de contorno, material com propriedades variável entre diversos outros agravantes que dificultariam a solução analítica, esses métodos seriam ainda mais complicados.

Para contornar esse problema na engenharia faz-se uso dos métodos numéricos. Os mais comuns são: (I) Método das Diferenças Finitas, (II) Método dos Elementos Finitos, (III) Método dos Volumes Controle e (IV) Método dos Volumes de Controle Baseado em Elementos Finitos.

I - MÉTODO DAS DIFERENÇAS FINITAS

Este método faz uso da definição da derivada, como sendo a tacha de variação de uma variável contínua num dado intervalo. Ao colocar o tamanho deste intervalo tendendo a um número suficientemente pequeno é possível obter bons valores da variável que está sendo aproximada (Tannehill, 1984). Porém esse método traz sérias limitações quanto à forma do ambiente de trabalho, sendo difícil sua aplicação para geometrias irregulares e que necessitem de uma malha mais refinada em alguns pontos.(Tombarevé,2013)

II - MÉTODO DOS ELEMENTOS FINITOS (MEF)

O Método dos Elementos Finitos foi desenvolvido para cobrir uma das maiores deficiências do Método das Diferenças Finitas. O MDF deve trabalhar com os contornos na direção do sistema de coordenada. Por esse motivo o MDF trabalha melhor com geometrias mais regulares. O MEF utiliza em cada

elemento uma interpolação apropriada. Neste método uma função de interpolação é multiplicada pela equação diferencial e este produto integrado no elemento.(Tombarevé,2013)

III - MÉTODO DOS VOLUMES FINITOS (MVF)

O Método dos Volumes finitos utiliza a mesma idéia de separar o domínio em vários pequenos elementos. A equação diferencial é integrada diretamente no volume de controle. Neste método as funções de interpolação são utilizadas para aproximar as incógnitas nos pontos de integração.(Tombarevé,2013)

IV - MÉTODO DOS VOLUMES DE CONTROLE BASEADO EM ELEMENTOS FINITOS

O CVFEM é um método que junta às vantagens do MVF com as vantagens do MEF. Nesse método elementos poliédricos formam o volume de controle, e esses são construído seccionando os elementos finitos. Esse método, além de ser fisicamente fácil de compreender, é bastante flexível quanto à geometria do domínio analisado.(Tombarevé,2013)

5.3 - MALHA COMPUTACIONAL

Neste trabalho serão utilizadas malhas computacionais obtidas com o software Gambit®. Este software é capaz de gerar malhas em 2D e 3D, das mais variadas geometrias, ficando isso critério do usuário. O Gambit® gera uma malha computacional a partir de um desenho feito pelo usuário e exporta a malha e todas as informações necessárias em um arquivo *.neu. O padrão que este arquivo segue é explicado no Anexo I.

As malhas utilizadas aqui serão: dois cubos unitários com diferentes números de elementos; uma semi-esfera; e um módulo de uma parede composta.

Para os cubos unitários o objetivo de se utilizar as duas malhas para o mesmo problema é para realizar testes de convergência. A malha mais grossa tem 95 pontos e 271 elementos distribuídos por todo o volume do cubo. A

malha mais fina tem 848 pontos e 3612 elementos. Uma representação dessa malha pode ser vista na figura 1

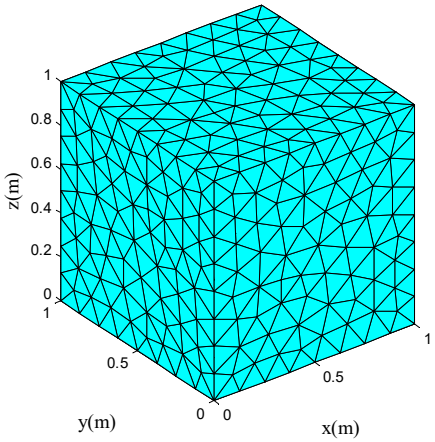

Figura 1- Malha de 848 pontos

Cinco malhas da semi esfera serão utilizadas neste trabalho, essas malhas contém 86, 495, 1826, 3305, 4465, pontos e 241, 2007, 8494, 16166 e 22239 elementos. Estas malhas serão utilizadas para analisar o problema clássico de aquecimento de um ovo sob condição de contorno de fluxo de calor convectiva. Uma representação desta malha pode ser vista na figura 2

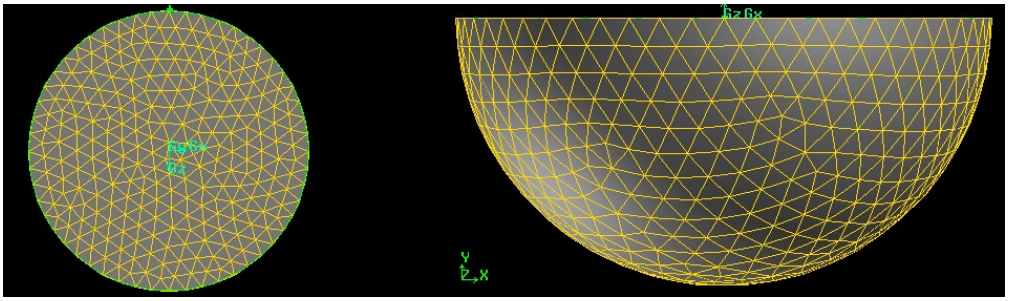

Figura 2 - Malha Semicírculo

A última malha utilizada para o caso da transferência de calor conjugada foi a malha presente na figura 3. Esta contém 8886 ponto e 44720 elementos.

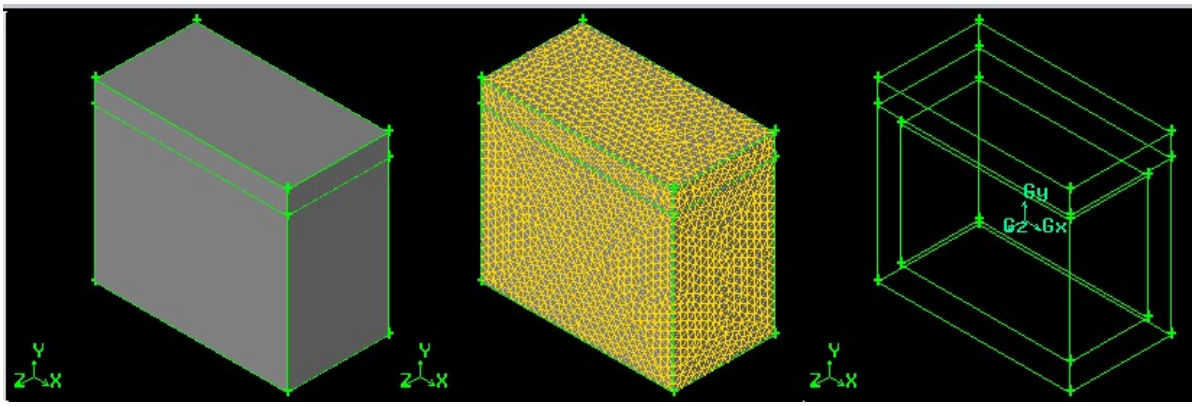

Figura 3 – Geometria e malha do módulo da parede

6. EQUAÇÃO GERAL REGIME PERMANENTE

6.1-FORMULAÇÃO

A formulação do método tem início com a equação 4.3 a qual considera somente o termo difusivo. A integral desta sobre um volume de controle fechado qualquer resulta na equação 6.1.

$$\int_{\mathrm{VC}} \nabla(k\nabla\emptyset)\, dV = 0 \tag{6.1}$$

Em seguida com o **teorema da divergência de Gauss** obtém-se a equação 6.2. (Wolfram alpha, 2017)

$$\oint_A \; k\nabla\emptyset n dA = 0 \tag{6.2}$$

Esta equação pode ser dividida de forma que cada face do volume de controle contribua para a integral inteira, ou seja:

$$\oint_A \; k\nabla\emptyset_p n dA = \int_{A1} k\nabla\emptyset_1 n dA_1 + \int_{A2} k\nabla\emptyset_2 n dA_2 + F_{nb} = 0 \tag{6.3}$$

Onde ϕ_1, ϕ_2 e etc, são pontos que estão ligados ao ponto ϕ_p e F_{nb} representa todas as faces entre os pontos vizinhos.Onde o subscrito “np” indica *neighbors* do inglês “vizinhos” e “p” o ponto de interesse. Até que a integral fechada seja reconstituída. A equação 6.3 pode ser interpretada como o somatório das contribuições de todas as faces multiplicada pelo coeficiente de condução e a temperatura no ponto oposto que liga a face ao nó central p.

$$a_p\emptyset_p = \sum a_{nb}\emptyset_{nb} + b_p \tag{6.4}$$

6.2-APROXIMAÇÃO FUNÇÕES DE FORMA

Uma função de interpolação linear para um ponto qualquer dentro de um tetraedro utilizado neste método pode ser apresentada em termos de suas funções de base da seguinte maneira:

$$\emptyset = N_1\emptyset_1 + N_2\emptyset_2 + N_3\emptyset_3 + N_4\emptyset_4 \tag{6.5}$$

$$\emptyset = \emptyset(x, y, z) \tag{6.6}$$

Os sub índices indicam os quatro vértices do tetraedro sendo as referências a eles indicadas pela equação 6.5 a qual será entendida como os vértices B,C e D do tetraedro além do centróide do mesmo no ponto O

$$\emptyset_1 = \emptyset(x_1, y_1, z_1) = \emptyset_A \quad (6.7)$$
$$\emptyset_2 = \emptyset(x_2, y_2, z_2) = \emptyset_B \quad (6.8)$$
$$\emptyset_3 = \emptyset(x_3, y_3, z_3) = \emptyset_C \quad (6.9)$$
$$\emptyset_4 = \emptyset(x_4, y_4, z_4) = \emptyset_D \quad (6.10)$$
$$\emptyset_0 = \emptyset(x_0, y_0, z_0) = \emptyset_o \quad (6.11)$$

Na equação 6.5 temos que os termos N_1,N_2, N_3, N_4 podem ser expandido como na forma das equações 6.12(Tombarevé,2013)

$$N_1 = \frac{a_1 + b_1 * x + c_1 * y + d_1 * z}{6 * V_{ABCD}} \quad (6.12)$$

Termos estes que descrevem a interpolação linear entre os pontos do tetraedro. Estes podem ser obtidos com uma combinação específica do produto escalar pelo produto vetorial dos 5 pontos seguindo as equações propostas. (Tombarevé,2013)

$$N_1 = \overrightarrow{BO}(\overrightarrow{BD} \times \overrightarrow{BC}) \quad (6.13)$$
$$N_2 = \overrightarrow{AO}(\overrightarrow{AC} \times \overrightarrow{AD}) \quad (6.14)$$
$$N_3 = \overrightarrow{AO}(\overrightarrow{AB} \times \overrightarrow{AD}) \quad (6.15)$$
$$N_4 = \overrightarrow{AO}(\overrightarrow{AC} \times \overrightarrow{AB}) \quad (6.16)$$

Onde os termos duplos são dados pela subtração dos vetores como no exemplo abaixo:

$$\overrightarrow{BO} = \vec{O} - \vec{B} \quad (6.17)$$
$$\overrightarrow{BO} = \{x_0 - x_2, y_0 - y_2, z_0 - z_2\} \quad (6.18)$$

Da equação 6.13 utilizando o determinado na equação 6.18 teremos a matriz de exemplo

$$\begin{vmatrix} \hat{\imath} & \hat{\jmath} & \hat{k} \\ x_4 - x_2 & y_4 - y_2 & z_4 - z_2 \\ x_3 - x_2 & y_3 - y_2 & z_3 - z_2 \end{vmatrix} \quad (6.19)$$

Cujo determinante é igual a:

$$[(y_4 - y_2)(z_3 - z_2) - (y_3 - y_2)(z_4 - z_2)]\hat{\imath} +$$
$$[(x_3 - x_2)(z_4 - z_2) - (x_4 - x_2)(z_3 - z_2)]\hat{\jmath} +$$
$$[(x_4 - x_2)(y_3 - y_2) - (x_3 - x_2\ (y_4 - y_2)]\hat{k} = \overrightarrow{BD} \times \overrightarrow{BC} \quad (6.20)$$

Acrescentando os últimos termos correspondentes ao produto escalar fora dos parênteses é fácil obter a equação 6.21

$$(y_4z_3 - y_4z_2 - y_2z_3 + y_2z_2 - y_3z_4 + y_3z_2 + y_2z_4 - y_2z_2) * (x - x_2) +$$
$$(x_3z_4 - x_3z_2 - x_2z_4 + x_2z_2 - x_4z_3 + x_4z_2 + x_2z_3 - y_2z_2) * (y - y_2) +$$
$$x_4y_3 - x_4y_2 - x_2y_3 + x_2y_2 - x_3y_4 + x_3y_2 + x_2y_4 - x_2y_2) * (z - z_2) =$$
$$\overrightarrow{BO}\left(\overrightarrow{BD} \times \overrightarrow{BC}\right) = N_1 \quad (6.21)$$

Seguindo a equação 6.12 devemos obter os termos a_1, b_1, c_1 e d_1 da equação 6.21 separando somente os termos independentes de x,y e z termos o resultado do coeficiente a_1. Somente os termos que multiplicam por x temos b_1, somente os termos que multiplicam por y temos c_1 e somente os que multiplicam por z temos d_1:

$$-x_2y_4z_3 + x_2y_3z_4 - x_3y_2z_4 + x_4y_2z_3 - x_4y_3z_2 + x_3y_4z_2 = a_1 \quad (6.22)$$
$$y_4z_3 - y_4z_2 - y_2z_3 - y_3z_4 + y_3z_2 + y_2z_4 = b_1 \quad (6.23)$$
$$x_3z_4 - x_3z_2 - x_2z_4 - x_4z_3 + x_4z_2 + x_2z_3 = c_1 \quad (6.24)$$
$$x_4y_3 - x_4y_2 - x_2y_3 - x_3y_4 + x_3y_2 + x_2y_4 = d_1 \quad (6.25)$$

Neste caso para que as funções de forma estejam completas é necessário que o último termo referente volume do tetraedro seja calculado. Utilizando a equação 6.26 para cada elemento é possível completar a formulação para as funções de forma (Tombarevé,2013).

$$V_e = \frac{1}{6}\begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} \quad (6.26)$$

A título de conferir se o cálculo do volume dos elementos está retornando valores certos, é recomendável que todos os pequenos volumes dos elementos sejam somados, esta soma deve apresentar o volume do objeto inteiro. Este teste está presente no código computacional desenvolvido. No primeiro caso investigado a soma deve ser o volume do cubo unitário,

Para conferir se as funções de forma N_1, N_2, N_3 e N_4 estão corretas é recomendável que o seguinte teste seja feito para auxiliar na validação do código computacional.

$$N_1\emptyset_o + N_2\emptyset_o + N_3\emptyset_o + N_4\emptyset_o = 1 \quad (6.27)$$

Esse teste deve ser aplicado a todos os elementos. Se ele for verdadeiro mesmo assim, isso significa que não há erros no desenvolvimento da função de forma dos elementos. Esse tipo de erro é bastante comum, pois na

formulação dessa etapa são utilizadas muitas variáveis com sub índices diferentes e que se permutam, como na equação 6.23.

Ficando assim completa a equação 6.5 seguindo a simples expansão nos outros vértices dos procedimentos até então realizados. As equações dos coeficiente das funções de forma podem ser visualizadas no código computacional desenvolvido.

6.3-GRADIENTE DAS FUNÇÕES DE FORMA

O interessante neste método é que se usarmos funções de forma lineares para descrever o comportamento da variável ϕ no interior do volume de controle e do elemento é que essas equações de 1º grau as variáveis *x,y* e *z* são independentes. Essa sutileza na escolha faz possível uma grande simplificação nos cálculos que seriam necessários e ainda assim trás resultados muito bons para problemas difusivos como a condução de calor num sólido.

Como a função ϕ(x,y,z) esta sendo tratada aqui de forma linear dentro do elemento temos que o gradiente desta função é dada pela equação 6.28. A integral de superfície fechada na equação 6.3é considerando um nó central P (por exemplo) e todas as superfícies normais que rodeiam este nó (faces dos subvolumes de controle)

$$\nabla \emptyset = \frac{d\emptyset}{dx}\hat{\imath} + \frac{d\emptyset}{dy}\hat{\jmath} + \frac{d\emptyset}{dz}\hat{k} \tag{6.28}$$

Sendo assim estes são constantes em cada direção *x, y* e *z*

$$\nabla \emptyset = b\emptyset\hat{\imath} + c\emptyset\hat{\jmath} + d\emptyset\hat{k} \tag{6.29}$$

Por serem constantes estes termos podem “sair da integral” a qual ficará somente responsável por resolver a área associada, ou seja, o vetor normal da superfície o qual também vem dividido em cada uma das direções i, j e k segundo a orientação da face que será integrada. Como cada um dos termos da equação 6.29 estão associados a uma direção i, j e k temos que a substituição da equação 6.29 na equação 6.3 resulta que os termos que não apontam na mesma direção sejam nulos e no fim desta seqüência temos que somente sobra os termos da equação 6.30 abaixo

$$\oint_s k\nabla\emptyset nds = k(b_1 n_x A + c_1 n_y A + d_1 n_z A)\,\emptyset_1 + k\big(b_2 n_x A + c_2 n_y A + d_2 n_z A\big)\emptyset_2 + k\big(b_3 n_x A + c_3 n_y A + d_3 n_z A\big)\emptyset_3 + (\mathrm{F}_{nb}) = 0 \quad (6.30)$$

Onde F_nb expressa a contribuição de todos os nós ligados ao nó central. É bom ter em mente que cada elemento finito é composto por quatro sub volumes de controle, então a equação 6.30 acima deve ser calculada para cada um dos nós dentro de um elemento, com as respectivas áreas normais entre eles e os coeficientes da função de forma associado. Quando esta etapa for realizada para todos os pontos da malha esta será equivalente ao último termo F_{nb}. Assim a integral fechada engloba totalmente cada um dos pontos. Esta junção será realizada na seção 6.4 e 6.5 após serem definidos os vetores normais e as faces associadas.

6.4-VETORES NORMAIS E FACES INTERNAS

Ao observar a equação 6.30 é possível ver que cada ponto de um elemento qualquer, tem uma equação de área normal associada quando conectado entre os demais 3 pontos do mesmo elemento. Esses valores são diretamente utilizados no momento de se atribuir as funções de forma e suas respectivas derivadas a cada elemento em análise. Na equação 6.34 é possível ver, tomando o nó simbólico P que o vetor normal se dá entre dois pontos de interesse, o ponto 1 e 3. Por exemplo,o vetor normal que passa pelos pontos 1,c e 3 pode ser dado pelo produto vetorial dos pontos que formam essa face divididos em cada direção x,y,z em detrimento do vetor normal unitário multiplicado pela área da face entre 1 e 3.

Para calcular esse vetor normal é necessário primeiramente definir as coordenadas dos pontos que compõe a área associada então a geometria do problema será exemplificada.

Cada elemento finito contem 12 pontos que compõem cada uma das faces internas. Os pontos “*a”, “b”, “c”, “d”, “e” e “f”* se encontram no meio de cada aresta do elemento tetraédrico. Os pontos “*q”, “r”, “s”*e “*t”* se encontram no baricentro de cada uma das faces que compõem o tetraedro e por o ponto “*o”*representa o centróide do elemento. Estes pontos podem ser visualizados nas Figuras 4 e 5.

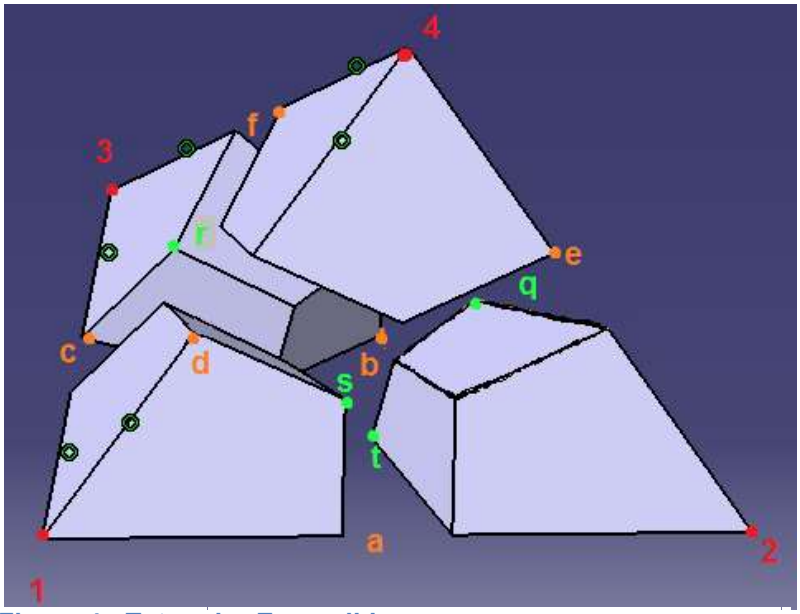

Figura 4 - Tetraedro Expandido

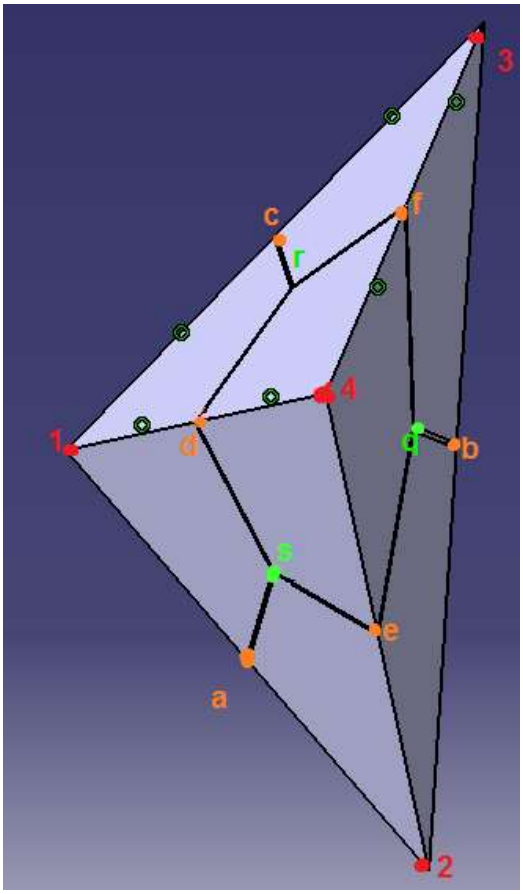

Figura 5 - Tetraedro vista superior

Em verde “q”,“r”,“s”, “t” em alaranjado “a”,“b”,“c”,“d”,“e”,“f” em vermelho “1”,“2”,“3”,“4”.

Para facilitar a visualização e o desenvolvimento das equações, a seguinte notação será adotada: Pi= [Xi, Yi, Zi], onde o P com algum sub índice irá simbolizar um ponto qualquer no espaço com as suas 3 coordenadas X,Y e Z.

As coordenadas cartesianas desses pontos auxiliares podem ser obtidas pela média aritmética dos pontos do elemento que estão associados a estes. As equações 6.31 a 6.33 mostram como obter para alguns deles. O mesmo raciocínio é seguido para obter as coordenadas dos demais pontos auxiliares em cada uma das direções cartesianas.

$$P_a = \frac{P_1+P_2}{2} \quad (6.31)$$

$$P_r = \frac{P_1+P_3+P_4}{3} \quad (6.32)$$

$$P_o = \frac{P_1+P_2+P_3+P_4}{4} \quad (6.33)$$

Seguindo esta abordagem dos vetores normais em cada face é possível obter cada uma das 6 áreas associadas utilizando os pontos auxiliares descritos anteriormente. Essas áreas são: ATOS, CTOR, DSOR, BQOT, EQOS e FROQ. As equações 6.34 e 6.35 descrevem algum desses elementos. Sendo no nome adotado para os vetores normais: o número referente a origem do vetor normal e a letra o destino do mesmo. Pela Figura 6 é possível ver a qual área o vetor normal esta associado. Nesta estão presentes as áreas ATOS, CTOR e DSOR.

$$n1a= (1/2)*(Pa-Po)X(Pt-Ps) \quad (6.34)$$

$$n1d=(1/2)*(Pd-Po)X(Ps-Pr) \quad (6.35)$$

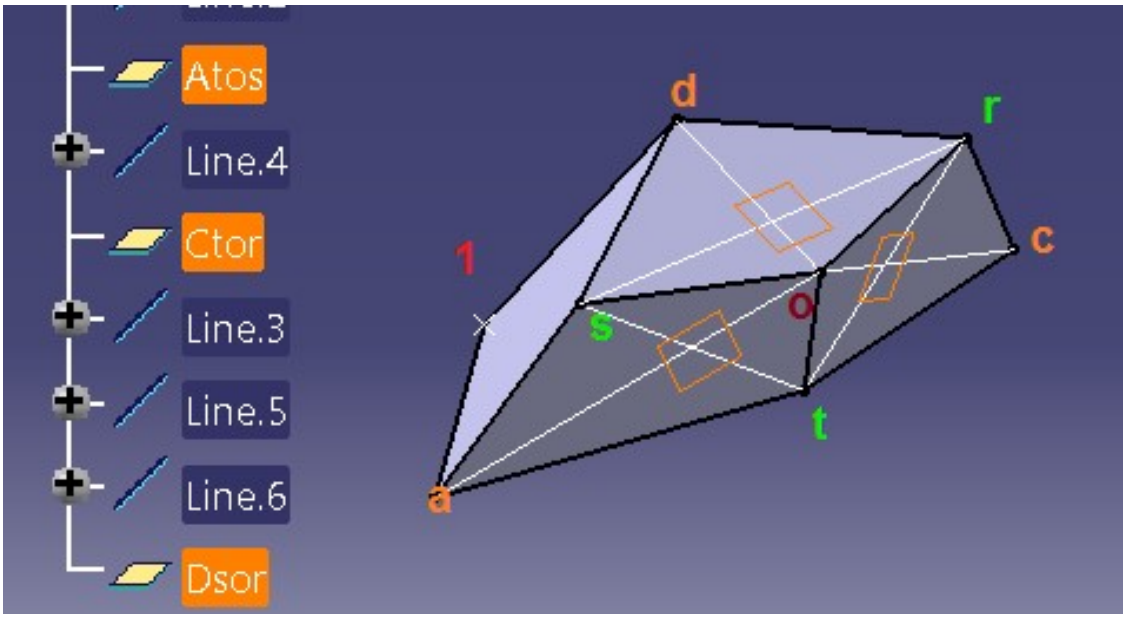

Figura 6 - Volume De Controle Ligado ao Nó 1 Com Elemento Normal de Área Associado

Pela Figura 6 é possível ver os dois vetores que formam a área ATOS deste elemento de controle. O produto cruzado entre esses dois vetores retorna o vetor normal da área ATOS que está entre os pontos 1 e 2 .Mesmo raciocínio deve ser seguido para todos as outras 5 áreas do elemento e então os vetores normais entre os 4 ponto serão obtidos. As equações 6.34 e 6.35 determinam como obter vetor n1a e n1d, associados às áreas ATOS e DSOR, respectivamente.

6.5 - SUB VOLUMES DE CONTROLE

Para dar continuidade ao método CFVEM a equação 6.3, que representa a integral fechada da área, será simplificada nas 3 primeiras integrais abertas que irão representar as três áreas do subvolume de controle associados a um nó central qualquer. Para ilustrar será usado um nó P quando o *loop* por cada elemento passar novamente em um elemento que contenha esse mesmo nó P outras 3 integrais de área irão se somar às 3 já existentes, caso o nó P não se encontre nos contornos do volume de controle esse processo irá se repetir até que todos os elementos que sejam formados pelo nó P somem suas

contribuições para o nó e assim a integral de fluxo de área fechada envolvendo o nó P é restaurada.

Se o nó P se encontrar no contorno da geometria que esta sendo analisada a integral fechada do fluxo por área não é restaurada. Dessa forma as condições de contorno do problema vêm a completar a equação neste ponto. Esses dois exemplos são mostrados na Figura 7 em 2D para facilitar a visualização, porém é importante ter em mente que o problema aqui proposto é um problema 3D e a geometria é ligeiramente mais complexa.

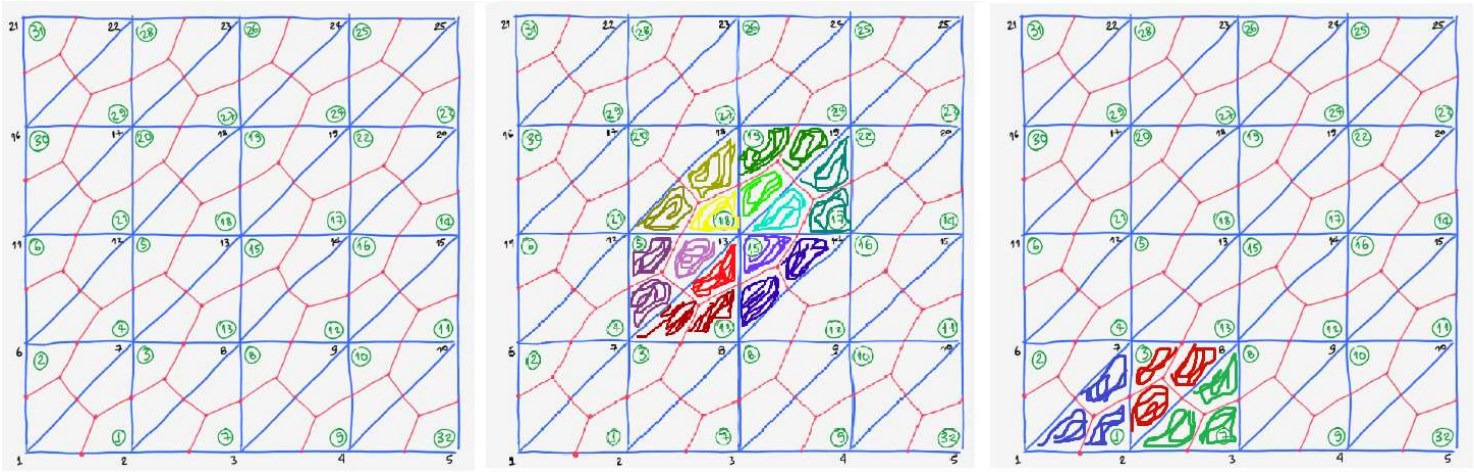

Figura 7- Loop Elementos CVFEM **fonte: (Cunha,2010) com adaptações**

No loop nos elementos na figura 7 acima é possível ver como a integral é reconstituída em um volume de controle. Na imagem da esquerda é mais fácil visualizar a geometria:os pontos são identificados de 1 a 25 em preto, os elementos são identificados de 1 a 32 em verde, as linhas vermelhas indicam os volumes de controle e as linhas azuis os elementos finitos. Na Seção do meio está presente o loop nos elementos para um volume de controle fora das bordas. Nesta parte é possível ver que as contribuições dos subvolumes são somadas progressivamente ao nó central 13. Primeiro a contribuição do elemento 5 em roxo depois do 13 em vermelho, do 15 em azul e assim por diante até que a integral de fluxo por área fechada seja restaurada. No quadro da direita o loop nos elementos é mostrado para o caso de um volume de controle que se encontra nas fronteiras da malha, neste caso o nó central 2. O loop inicia no elemento 1 em azul , depois no 3 em vermelho e por ultimo no 7 em verde. Cada elemento vai somando suas contribuições ao nó central até que todos os subvolumes de controle sejam somados. Neste caso por se

encontrar na fronteira do sistema a equação necessita das condições de contorno para completar os termos que ainda faltam.

A representação deste método em 3D utilizando elementos finitos tetraédricos pode ser observada na figura 8 abaixo. No quadro à esquerda somente um subvolume de controle está representado, no centro é ilustrado um volume de controle incompleto, de forma a ilustrar como os sub-volumes de controle se conectam e em seguida como seria a visualização do volume de controle completo após todas as contribuições de todos os nós que estão conectados ao nó “P” se somarem.

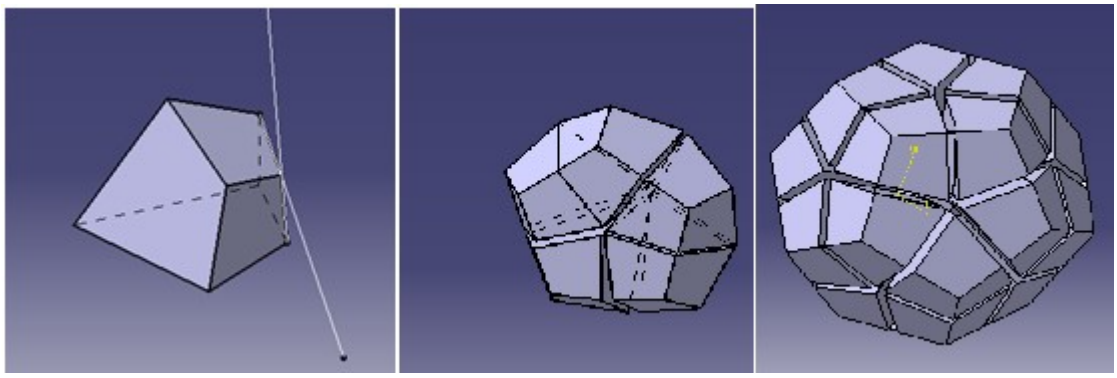

Figura 8 – Sub-volumes de controle e construção do volume de controle

6.6-CONDIÇÕES DE CONTORNO

Como anteriormente explicado para caso o nó esteja na fronteira a integral de fluxo dentro de um elemento não se completa então é necessário que os nós que estejam no contorno tenham a contribuição das condições de contorno do problema. Para problema de condução de calor em um sólido existem 3 tipo básicos de condição de contorno: (I) temperatura especificada na superfície, (II) fluxo de calor prescrito na superfície e (III) condição de fluxo de calor prescrito com coeficiente de convecção especificado na superfície.

A mais simples entre as três é a primeira. Com a temperatura especificada na superfície basta que o nó que esteja na superfície tenha a temperatura especificada e toda a equação que descreve o fluxo de calor no ponto do contorno seja igualada a zero, ou seja, deve-se zerar o coeficiente a_{nb} da equação 6.4 e igualar a 1 o coeficiente a_n da mesma equação, sobrando somente o termo da equação 6.36 abaixo.

$$\emptyset_p = \emptyset_{contorno} \tag{6.36}$$

A Figura 9 ilustra como que a matriz de coeficiente fica antes e depois de se aplicar esse tipo de condição de contorno.

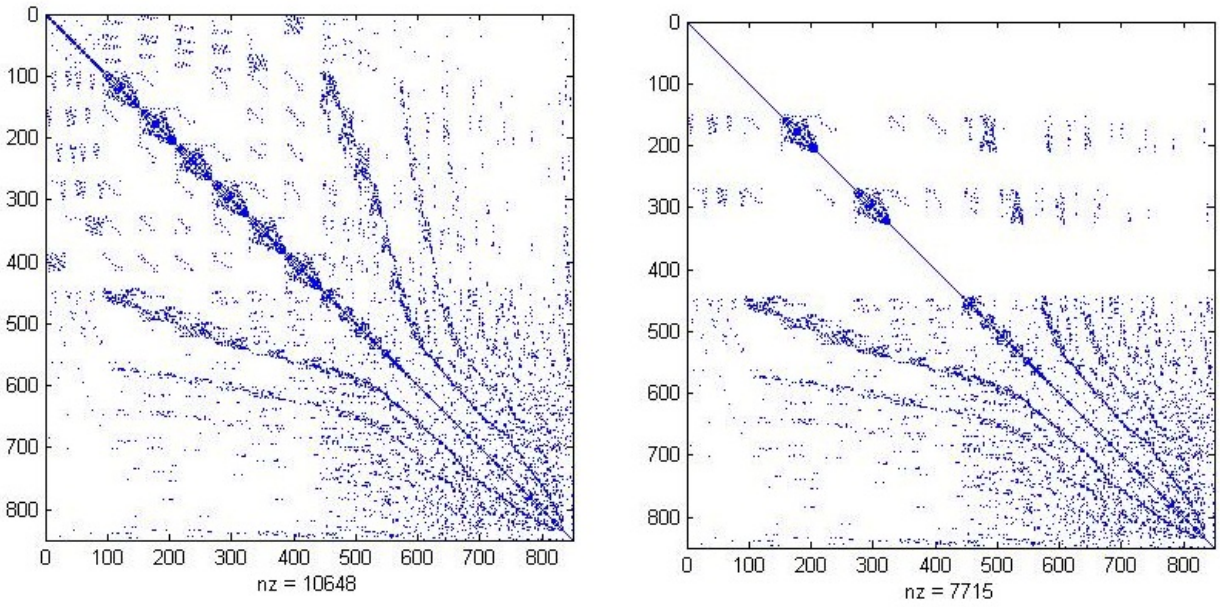

Figura 9 - Matriz Coeficiente Antes (Esquerda) E Depois (Direita) Das Condições De Contorno

O comando spy(_matriz_) foi utilizado para gerar a imagem. Este comando mostra um gráfico que aponta as coordenadas dos termos não nulos com pontos azuis.Neste exemplo a temperatura estava definida nas faces 1, 4, 5 e 6.

O segundo tipo de condição de contorno é ligeiramente mais complicado de formular. Pois neste a equação da condução deve ser completada. Para isso temos uma matriz que indica quais os nós estão no contorno e desses nós quais são os triângulos que são formados nas superfícies do contorno,sabendo disso multiplica-se o fluxo de calor em cada ponto pela área de triangulo. Assim cada ponto irá trazer suas contribuições para os pontos que o formam contorno fechando a integral de fluxo.

O último tipo de condição de contorno é feito igualando o fluxo de calor de convecção com o de fluxo de calor no elemento. O coeficiente de convecção térmica é utilizado nesse processo.

6.7-VALIDAÇÃO DO CÓDIGO COMPUTACIONAL

Para validar o código que esta sendo desenvolvido, será adotada a seguinte metodologia: (I) refinar malha, (II)comparar com resultados analíticos.Será considerado um problema para o código em regime permanente: com temperatura definida em 4 das 6 bordas do plano xy, com demais faces isoladas.

6.8-RESULTADOS

A primeira verificação proposta é refinando a malha. Se o código computacional funcionar para uma malha mais grosseira, ele também deve funcionar para uma malha mais refinada. Esse tipo de validação só é aconselhável para problemas difusivos, caso fosse considerado o termo advectivo esse tipo de teste não apresenta resultados muito bons(Tombarevé,2013).

Neste caso foram adotadas as seguintes condições de contorno: temperatura definida nas 4 bordas do plano xy, sendo 3 faces com T=0ºC e a última com distribuição senoidal de temperatura. Demais faces isoladas. Nesta primeira parte o foco principal não é as condições de contorno, as quais serão abordadas novamente na próxima etapa. O objetivo é observar a variação da suavidade dos resultados com a variação da densidade da malha.

Observa-se na figura 10 que tanto para a malha de 95 pontos, quanto para a malha de 848 pontos os valores interpolados apresentam uma correlação direta. Sendo assim o código computacional é dito como aprovado neste teste.

Para a segunda etapa segue a explicação do problema escolhido: 3 das faces do plano xy terão temperatura constante igual a zero e a ultima face terá distribuição de temperatura que obedece a equação 6.37, demais faces isoladas.

$$T = T_m * \mathrm{sen}\left(\frac{\pi x}{L}\right) \tag{6.37}$$

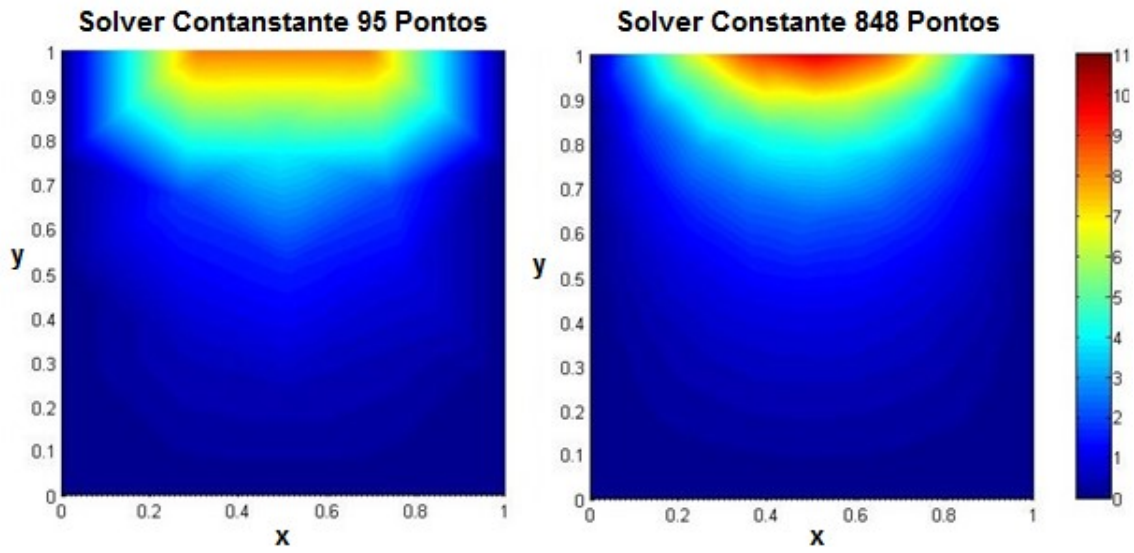

Figura 10 - Convergência código computacional utilizando malha de 95 Pontos (Esquerda) e 848 Pontos (Direita)

A solução analítica dessa equação em um problema tridimensional é bastante complicado, a solução analítica para um problema 2D seguindo as mesmas condições de contorno conforme a Figura 11, T_m =10ºC, T=0ºC em x=0 , T=0ºC em x=0 ,

T=0 em x=L , T= $T_m * \text{sen}(\frac{\pi x}{L})$ em y=b é dada por :

$$T(x,y) = \frac{T_m \sinh\left(\pi\frac{y}{L}\right)}{\sinh\left(\frac{\pi b}{L}\right)} sen\left(\pi\frac{x}{L}\right) \quad (6.38)$$

Os resultados obtidos pelo programa e os resultados obtidos analiticamente esta dispostos na figura 12.

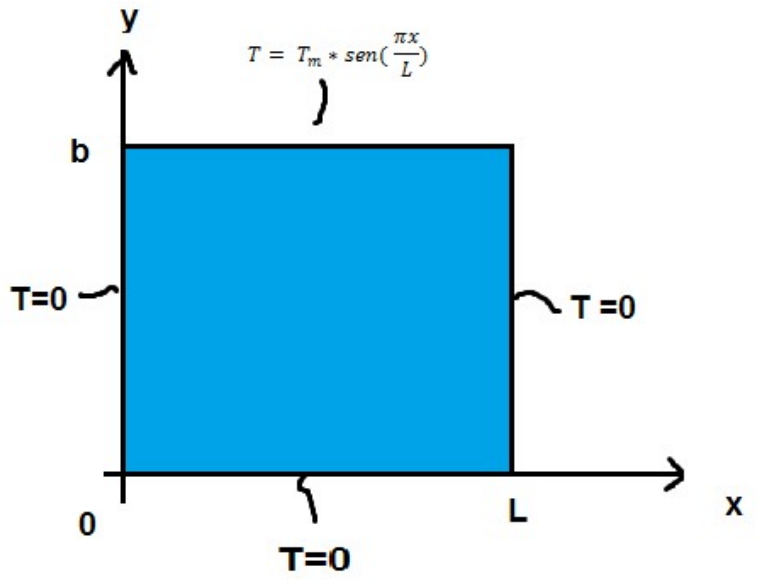

Figura 11 - Problema a Resolver

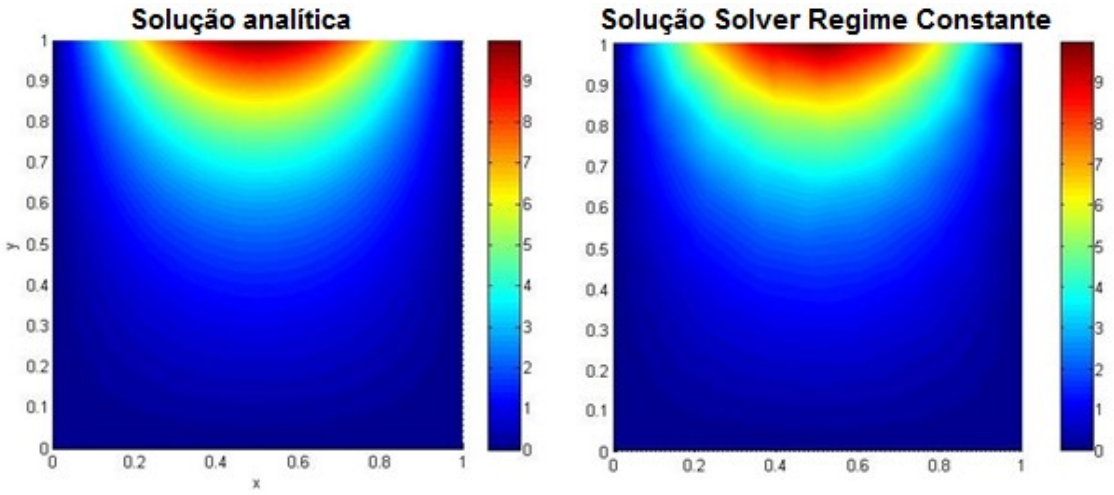

Figura 12 - Validação Regime Permanente

É possível ver que os resultados calculados por este código são muito parecidos com os calculados analiticamente. Estando assim o código apresentando resultados corretos.

O código leva em consideração uma geometria em 3D com a variação temperatura de entrada tanto em X quanto em Y, ou seja, é possível ver como se dá a distribuição da temperatura dentro material em outras direções como na figura13 abaixo, que mostra vários cortes do plano XY em Z= 1, em Z=0.8 ,Z=0.6 e Z=0.

É possível ver que a temperatura vai diminuindo até atingir 0ºC na região próximo à borda onde a temperatura é definida em 0 ºC em Z=0.

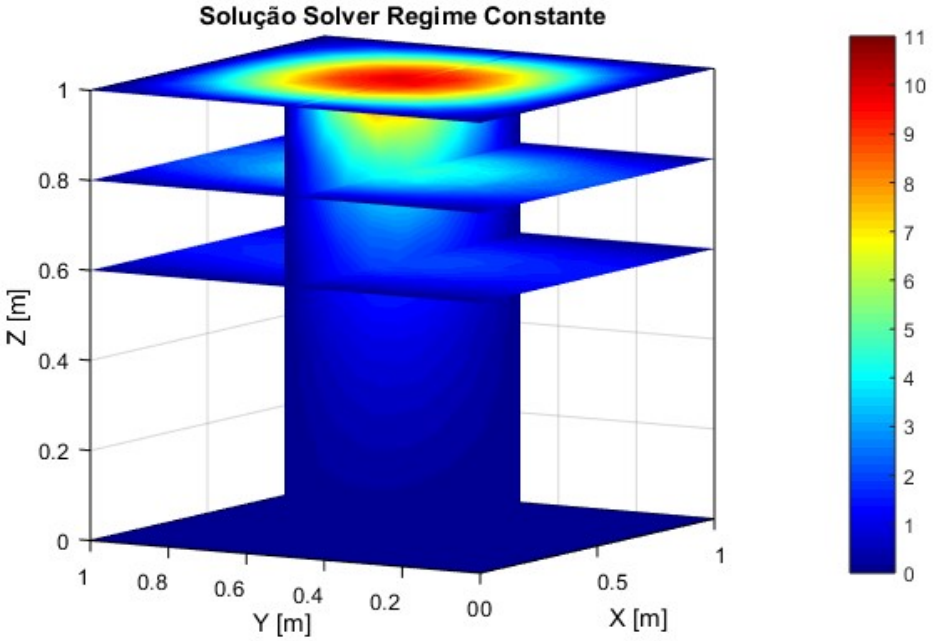

Figura 13 - Isoplanos

7. EQUAÇÃO GERAL TRANSIENTE

7.1-FORMULAÇÃO

Para trabalhar o termo de calor transiente terei como ponto de partida a equação 4.6 e então incluir o termo transiente (Patankar,1941).Primeiramente é feita a integral no tempo e no espaço da equação obtendo a equação 7.1 abaixo

$$\int_{VC}\int_{t}^{t+\Delta t}\frac{d\rho c\emptyset}{dt}dtdV-\int_{t}^{t+\Delta t}\int_{\mathrm{VC}}\nabla(k\nabla\emptyset)\,dVdt=0 \qquad (7.1)$$

O segundo termo desta equação já foi desenvolvido em parte neste trabalho.O mesmo raciocínio será tomado então, ou seja, aplicar teorema da divergência de Gauss na integral e separar a superfície de controle em várias pequenas superfícies já com a função de forma linear.

$$\int_{VC}\int_{t}^{t+\Delta t}\frac{d\rho c\emptyset}{dt}dtdV-\int_{t}^{t+\Delta t}(k\nabla\emptyset_1 nA_1+\ k\nabla\emptyset_2 nA_2+\ \ldots)\,dt=0 \qquad (7.2)$$

O segundo termo forma a integral no espaço já desenvolvido. O primeiro termo pode ser avaliado da seguinte forma.

$$\int_{VC}\int_{t}^{t+\Delta t}\frac{d\rho c\emptyset}{dt}dtdV=\rho c\left(\emptyset_p-\emptyset_p^0\right)\Delta V \qquad (7.3)$$

Onde o sobrescrito 0 se refere à temperatura no tempo t, e quando não há sobrescrito se refere à temperatura no tempo t + Δt.

Mesmo raciocínio pode ser adotado para o segundo termo da equação 7.2

$$\int_{t}^{t+\Delta t}(k\nabla\emptyset_1 nA_1+\ k\nabla\emptyset_2 nA_2+\ \ldots)\,dt=$$

$$\theta(k\nabla\emptyset_1 nA_1+\ k\nabla\emptyset_2 nA_2+\cdots)+\ (1-\theta)(k\nabla\emptyset_1 nA_1+\ k\nabla\emptyset_2 nA_2+\cdots)^0 \qquad (7.4)$$

O θ na equação 7.4 é um coeficiente de ponderação e este deve assumir valores entre 1 e 0. A escolha desse coeficiente é uma característica importante do código. Os três valores mais usuais de se encontrar na bibliografia dedicada a esse tema são 0, 0.5 e 1, levando a três métodos (I) explicito, (II) Crank-Nicolson e (III) implícito respectivamente (Patankar,1941).

Caso a variável θ seja igual a 0 o método assume que o valor de ϕ_0 prevalece sobre o passo no tempo até que seja substituído subitamente por ϕ no instante t + Δt. Esse método é limitado pela equação 7.5 e caso essa não

seja respeitada o método pode vir a apresentar problemas de instabilidade numérica. Nesta equação é possível ver que se mais acurácia na discretização espacial for solicitada é necessário que o valor de Δt também seja muito menor.

$$\Delta t < \rho c \frac{(\Delta x)^2}{2k} \tag{7.5}$$

Caso seja adotado θ=0.5 o método de Crank-Nicolson é descrito como sendo incondicionalmente estável. Porém muitas vezes esse método pode mesmo sendo estável resultar em problemas fisicamente não realistas.

O método totalmente implícito é garantido que resultados fisicamente realistas sejam obtidos. Por esse motivo será aqui adotado que θ seja sempre igual 1 (Patankar,1941).

Devido a essa escolha feita da equação 7.4 sobra somente o primeiro termo do lado direito, o qual o resultado já é conhecido, foi desenvolvido na seção anterior e é descrito pela equação 6.4. Juntando esse termo que permanece a equação 7.4 e o termo transiente da equação 7.3 de acordo com o que estava descrito na equação 7.2 o método para regime transiente pode ser definido por:

$$\left(\rho c \Delta \mathrm{V} + a_p\right)\emptyset_p = \sum a_{nb}\emptyset_{nb} + \rho c \Delta V \emptyset_p^0 + b_p \tag{7.6}$$

7.2-CONDIÇÃO INICIAL

As mesmas condições de contorno utilizadas para validar o código anterior podem ser normalmente utilizadas nesta etapa. Como anteriormente feito à temperatura será prescrita nos nós das paredes do contorno, ou seja, a diagonal principal e tomando à partir da equação 6.45 definida na seção 6.

$$\emptyset_p = \emptyset_{contorno} \tag{7.7}$$

Como presente na equação 7.6 para calcular o primeiro ϕ_p é necessário que saibamos o valor de ϕ_p^0 sendo assim o método necessita que as condições iniciais de todos os pontos do problema sejam definidas. Após definir essas condições o método pode sofre uma iteração, essa ação irá fornecer os valores de ϕ_p estes valores deverão ser armazenados na variável ϕ_p^0 para que o método possa passar por outra iteração, e assim sucessivamente, até atingir o equilíbrio ou até o número de iterações solicitadas pelo usuário.

7.3-VERIFICAÇÃO DO CÓDIGO COMPUTACIONAL

A verificação do *solver* transiente será feita assim como anteriormente utilizando um caso teste e a mesma metodologia: (I) refinar malha, (II)comparar com resultados analíticos. O caso que será abordado será: temperatura inicial maior do que zero e 1 face a temperatura igual a zero.Demais superfícies isoladas, ou seja, fluxo de calor nulo.

7.4-RESULTADOS

Como anteriormente realizado o primeiro teste é feito com o refinamento da malha. Os resultados desse teste serão montados com amostras no tempo 50s,200s,400s e 600s para ambas as malhas de 95 pontos e 848 com passo de tempo de 1 segundo e posteriormente com passo de 0.1 segundo. Tomando uma linha central, que atravessa o cubo de uma face à outra partindo da face a zero graus até a face oposta. Sendo todos os parâmetros unitários e a condutividade térmica k, igual a 10^4W/mK.

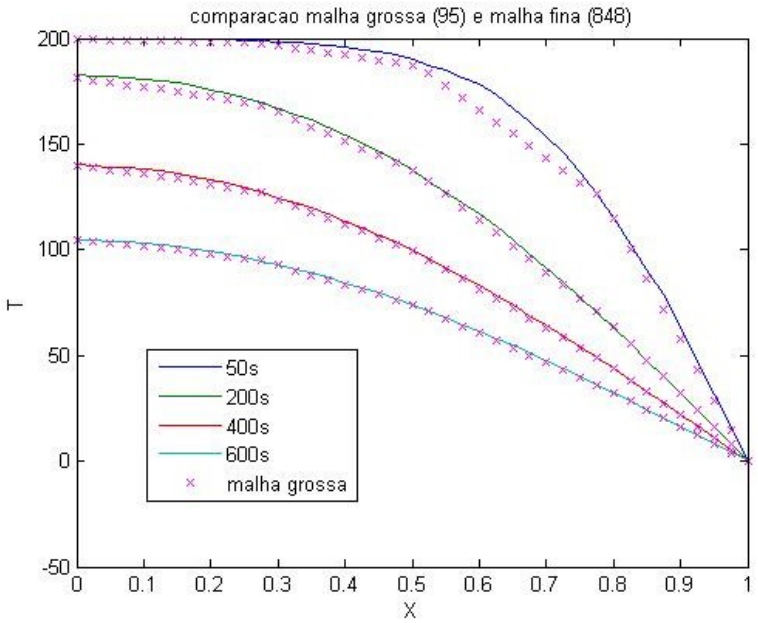

Figura 14 – Malha grossa 95 pontos e fina 848 pontos com dt =1 no momento 50s, 200s, 400s e 600s

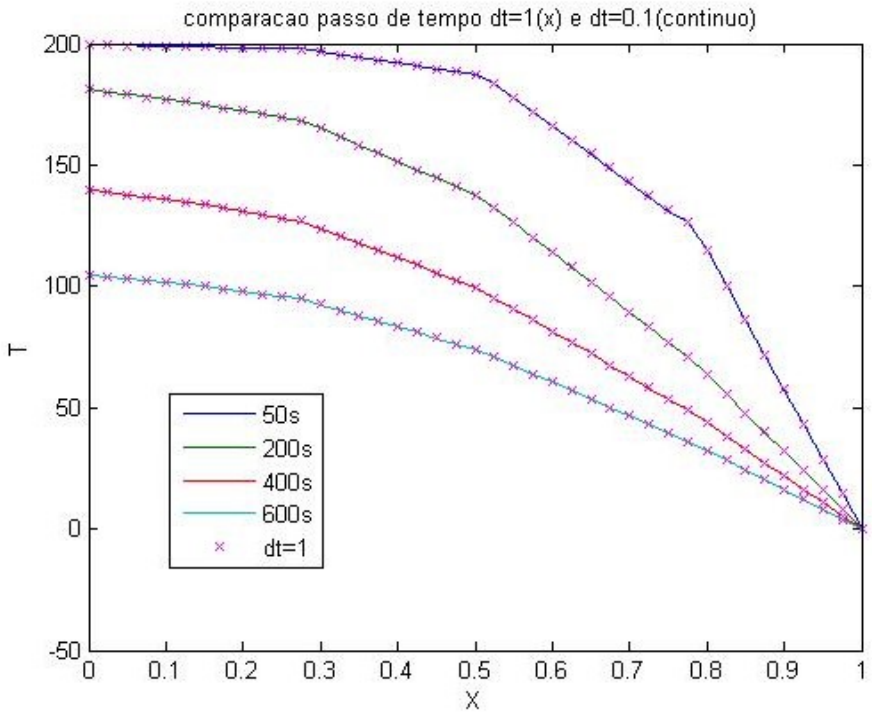

Figura 15 - Comparação passo de tempo dt=1 e dt =0.1

A comparação dos resultados na figura 14 mostra que com o refinamento da malha espacial os resultados são mais suaves o que indica que resultados melhores são encontrados. Com o aumento do passo tempo é possível ver que os resultados não diferem muito do da figura 14 anterior. Fato que já era esperado tendo em mente que este código computacional é aplicado somente para problemas difusivos.

A última análise que será feita aqui é comparar com o resultado analítico. Este resultado é bastante complicado de se obter, pois devem ser utilizadas ferramentas matemáticas avançadas para obter a equação da condução transiente 1D. A qual é dada pela equação 7.8 (Versteeg,2007)

$$T(x,t) = 4\frac{T_i}{\pi}\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{2n+1}\exp(-\alpha\lambda_n^2 t)\cos(\lambda_n^2 x) \qquad (7.8)$$

Onde:

$$\lambda_n = \frac{(2n-1)\pi}{2L} \qquad (7.9)$$

$$\alpha = \frac{k}{\rho c} \qquad (7.10)$$

Nas equações 7.10 acima o k é condutividade térmica do material, ρ é a densidade e c é calor específico. As figuras 16 e 17 trazem os resultados

sobrepostos, os analíticos e os calculados pelo código computacional para a malha mais grosseira.

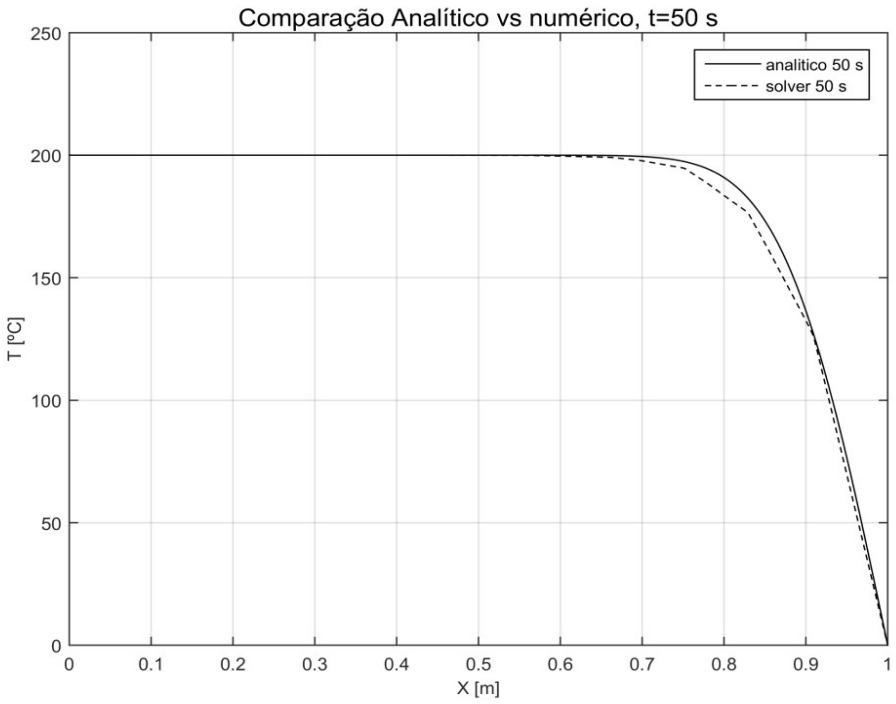

Figura 16 - Comparação analítico e numérico t = 50

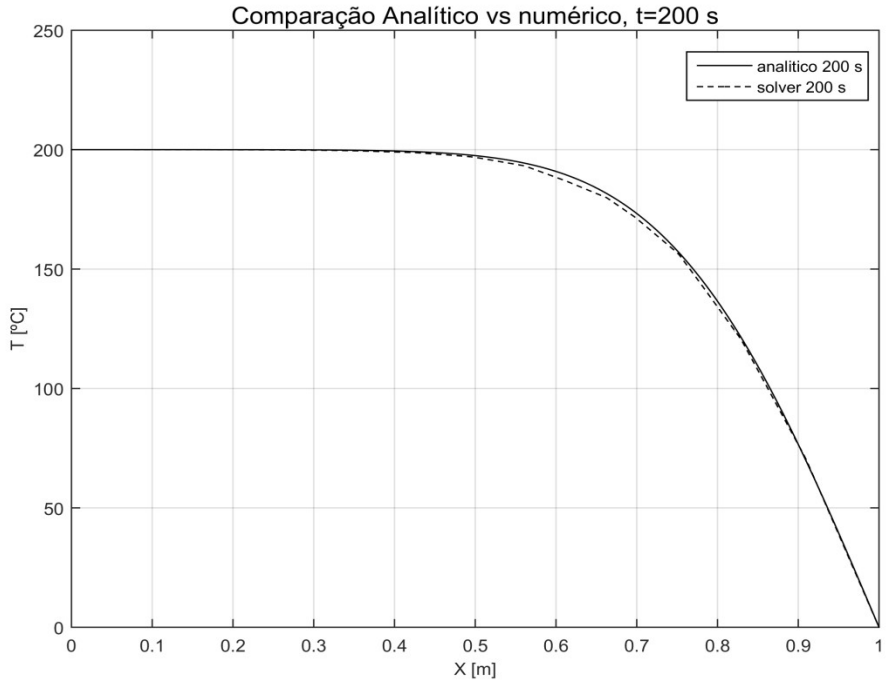

Figura 17 - Comparação analítico e numérico t = 200

É possível observa nas figuras 16 e 17 que os valore analíticos e numéricos se apresentam resultados bastante similares, portanto o código é dado como aprovado neste teste.

8. CONDIÇÃO DE CONTORNO CONVECTIVA

8.1 - FORMULAÇÃO

Como mencionado nos objetivos deste trabalho as condições de contorno com fluxo de calor prescrito sendo este com coeficiente de convecção “h” definidos também estão no escopo deste. O fluxo de calor pode ser avaliado de acordo com a equação 8.1 e 8.2.

$$Q = A * q \tag{8.1}$$

$$Q = h * A * (T_p - T_{amb}) \tag{8.2}$$

Estes termos caso o usuário solicite podem ser adicionado à equação 7.6 aplicando-a somente aos nós no contorno. O termo dependente de T_p pode ser adicionado ao lado esquerdo da equação, de forma implícita, reforçando assim a diagonal principal da matriz de coeficientes. O termo que depende de T_{amb} deve ser colocado no lado direito desta de forma explícita.

8.2 – VERIFICAÇÃO E RESULTADOS

Para a verificação do *solver* desenvolvido um caso teste clássico do aquecimento de um ovo será abordado. O problema proposto trata-se de um ovo, de 5cm de diâmetro inicialmente a temperatura de 5ºC e este é colocado em água fervendo a 95ºC. Considerando o coeficiente de convecção h=1.200W/m²K o objetivo é determinar quanto tempo seria necessária para que o centro do ovo atinja 70ºC.

Este problema cujo resultado analítico é 863.80s necessita de complexas ferramentas matemáticas para ser solucionado. A figura 18 apresenta a diferença entre o resultado analítico e os resultados obtidos pelo solver desenvolvido para diversas malhas utilizando passo de tempo fixo de dt=1s. Foram utilizadas 5 malhas com 86, 495, 1826, 3305, 4465 pontos.

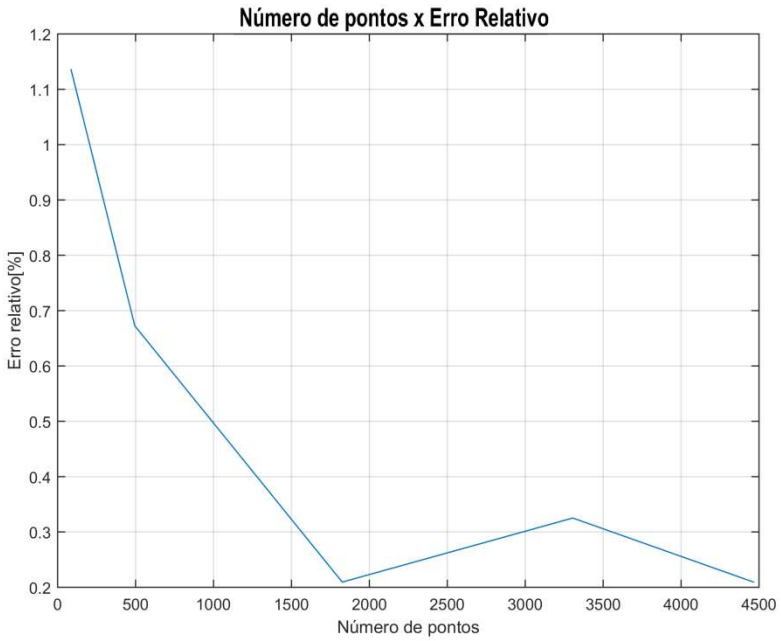

Figura 18 – Erro relativo Numérico (dt = 1) VS analítico

Observa-se da figura 18 que ao aumentar o número de pontos da malha computacional o erro sistematicamente diminui, porém o valor não atinge a precisão das casas decimais devido ao passo de tempo fixo adotado neste código.

O aumento do número de pontos também reflete na precisão do cálculo da área da superfície como pode ser observado na figura 19. Em contra partida a figura 20 mostra o aumento no tempo que o solver necessita para calcular um passo de tempo.

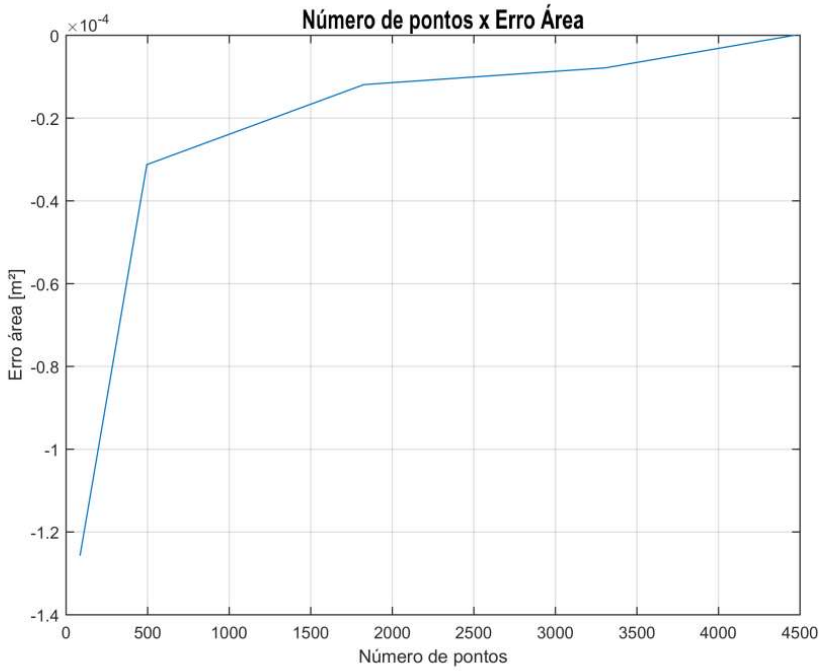

Figura 19 - Erro Área vs Número de Pontos

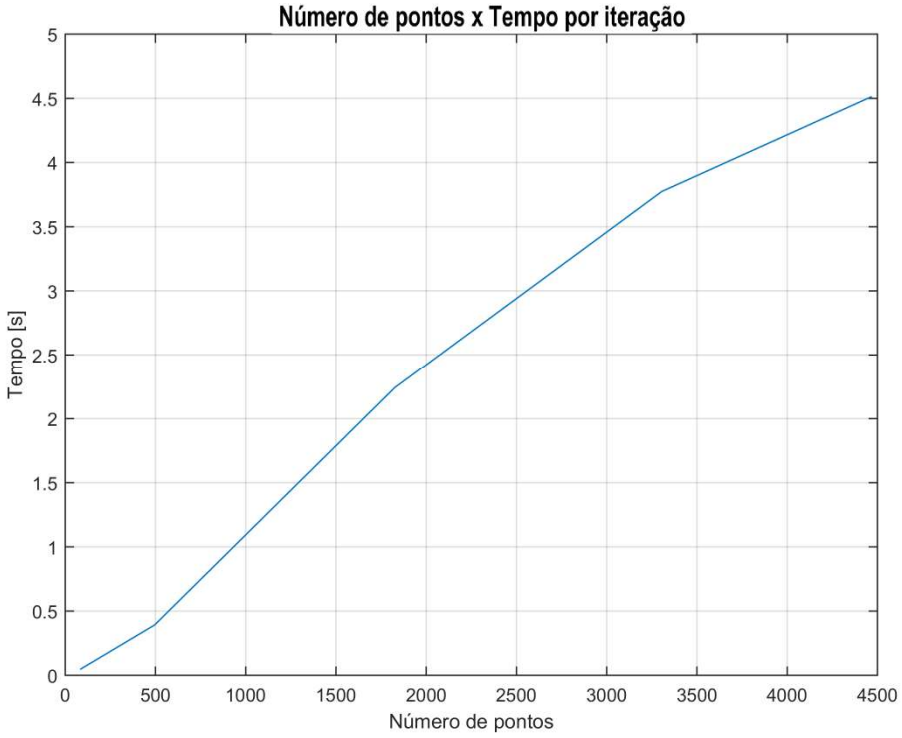

Figura 20 - Número pontos vs Tempo por iteração

Um teste interessante de ser realizado para esta condição de contorno e geometria é em relação à precisão do resultado ao variar o passo de tempo. Neste teste foi utilizada uma malha de 1245 pontos e os passos de tempo

adotados foram 0.01, 0.1, 0.5,1 2, 3, 4 e 5 segundos. Embora o tempo de processamento aumente proporcionalmente à variação do passo de tempo os resultados para os passos de tempo menores se mostram bem mais aceitáveis ao serem comparados com os valores analíticos. Os resultados se encontram na figura 21

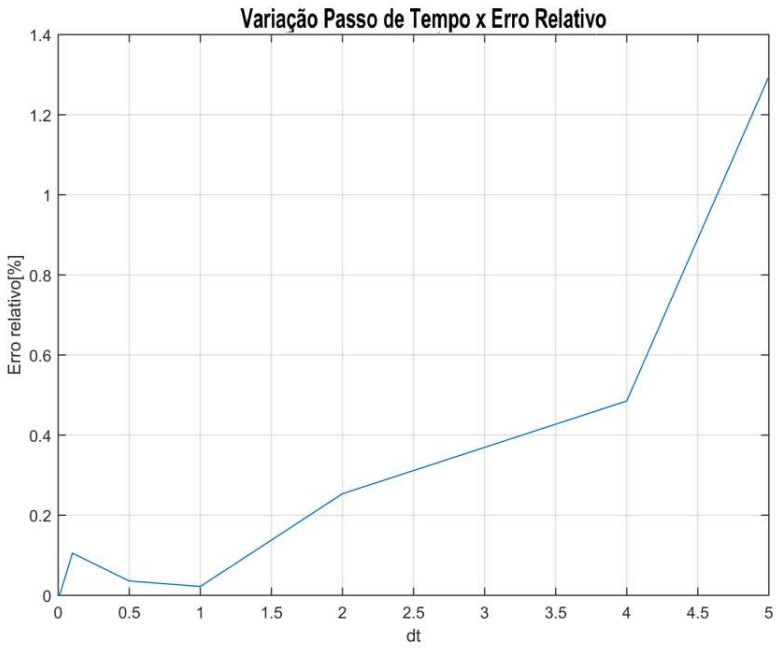

Figura 21 - Variação Passo de Tempo vs Erro Relativo

O último teste realizado foi um balanço de energia. Este pode ser feito tomando a diferença de energia interna no tempo t_1 para o tempo t_0 em todos os volumes de controle da malha computacional. Esta diferença de energia deve ser igual a energia que flui pelos contornos em um passo de tempo.

$$E_{interna} = cp * \rho * V * (T - T^0) \quad (8.3)$$

$$E_{transferida} = h * A * \left(T_p - T_{amb} \right)^* \text{ dt} \quad (8.4)$$

A figura 22 abaixo ilustra o balanço de energia obtido e este mostra que com o passar do tempo menos energia flui pelo contorno pois os nós no contorno estão com um temperatura muito próxima da temperatura ambiente enquanto o centro do ovo ainda não atingiu os 70 ºC. Fato que já era esperado tendo em vista que a problema que está sendo investigado apresenta um número de biot elevado de 47.8

A figura 23 complementa o concluído no raciocínio anterior. Onde devido ao elevado número de Biot é possível ver uma resistência ao fluxo de energia térmica. Sendo que a temperatura do volume de controle localizado no centro

da geometria analisada somente muda mais de 1ºC após 151 segundos do início do processo de aquecimento. E a temperatura dos pontos localizados no contorno muda rapidamente, sendo necessário apenas 28 segundos para que esta alcance 90% da temperatura máxima possível nestas condições (85.5 ºC).

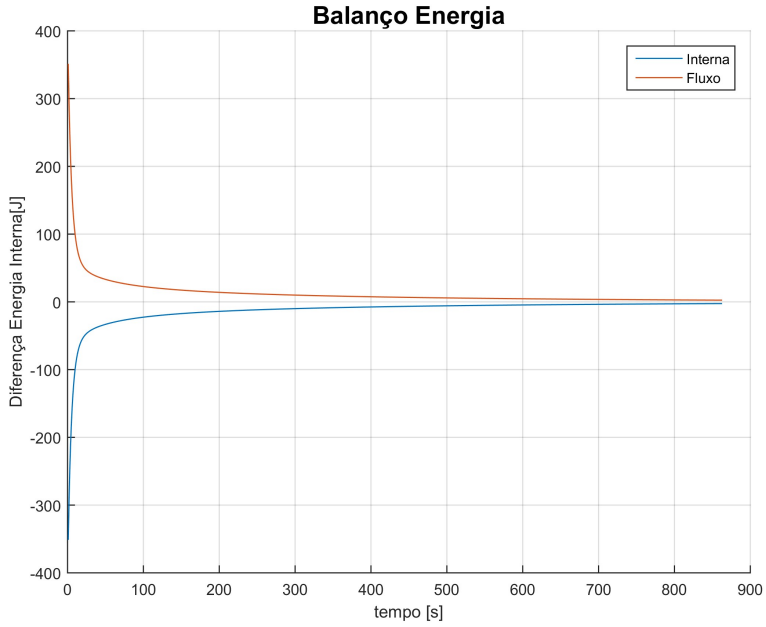

Figura 22 - Balanço de Energia Interna vs Fluxo

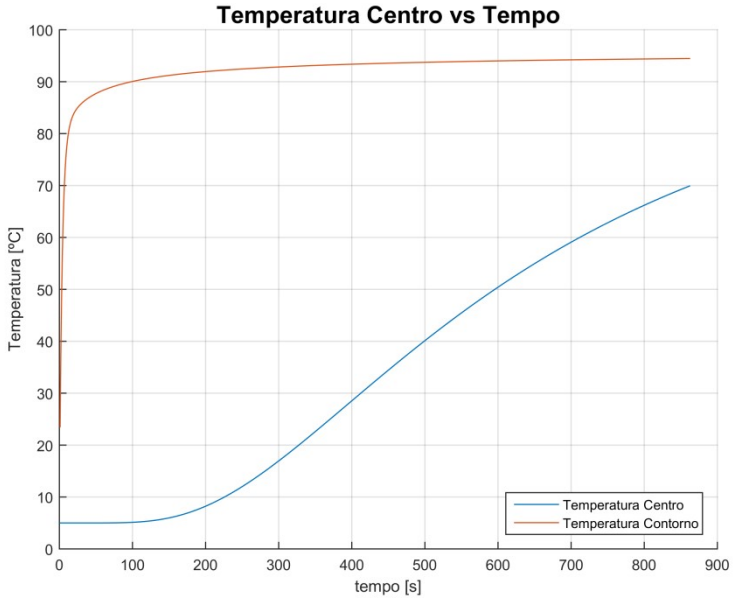

Figura 23 - Temperatura no Centro vs Temperatura no Contorno

9. FLUXO DE CALOR CONJUGADO

Para finalizar este projeto, a última funcionalidade adicionada ao código aborda casos de problemas de engenharia no qual o calor flui por diferentes materiais. Este funcionalidade é capaz de resolver problemas de transferência de calor que geralmente são simplificados utilizando a técnica de resistências térmicas. Porém o código desenvolvido é capaz de mostrar o comportamento no tempo do sistema, diferentemente das técnicas tradicionalmente utilizadas.

Esta última funcionalidade abre um gama de possibilidades para a aplicação real do código em problemas corriqueiros de engenharia. O caso escolhido neste trabalho foi a transferência de calor por uma parede composta. Esta parede é composta por três materiais: gesso, espuma de poliuretano e um tijolo.

A figura 24 abaixo ilustra o problema que está sendo explorado. Neste caso, o tijolo têm a seguinte geometria 22x16x9 cm e cobertura de 2cm de gesso no lado superior e inferior. O coeficiente de convecção do lado T1 e a temperatura T1 são respectivamente 10 W/mK e 10ºC e no lado T2 são 25W/mK e -10ºC. As propriedades do material se encontram na figura 24.

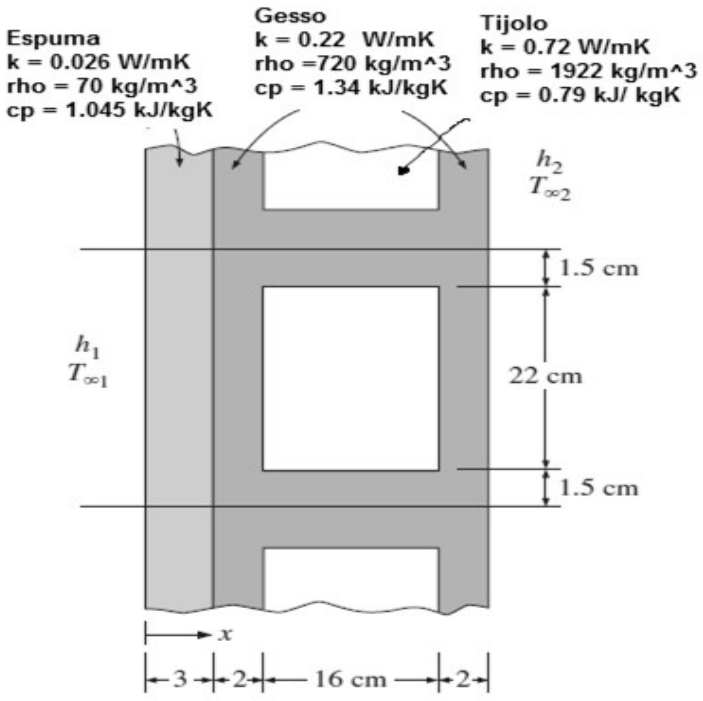

Figura 24 - Problema proposto, parede composta. Fonte: (Çengel, 2003) com adaptações

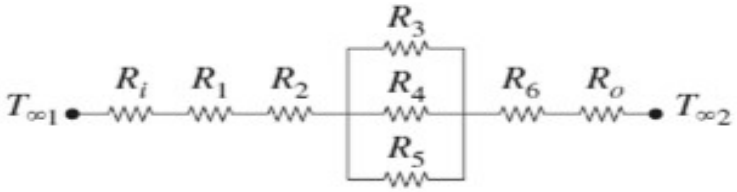

Figura 25 - Equivalentes por resistências. Fonte: (Çengel, 2003)

O passo de tempo adotado foi de 5s e a malha utilizada contêm 8886 pontos. No momento em que se observou que a temperatura atingiu o regime permanente deu-se por encerrado o teste. Os resultados obtidos analiticamente e numericamente se apresentam bastante similares como se pode observar na figura 26 e 27 abaixo.

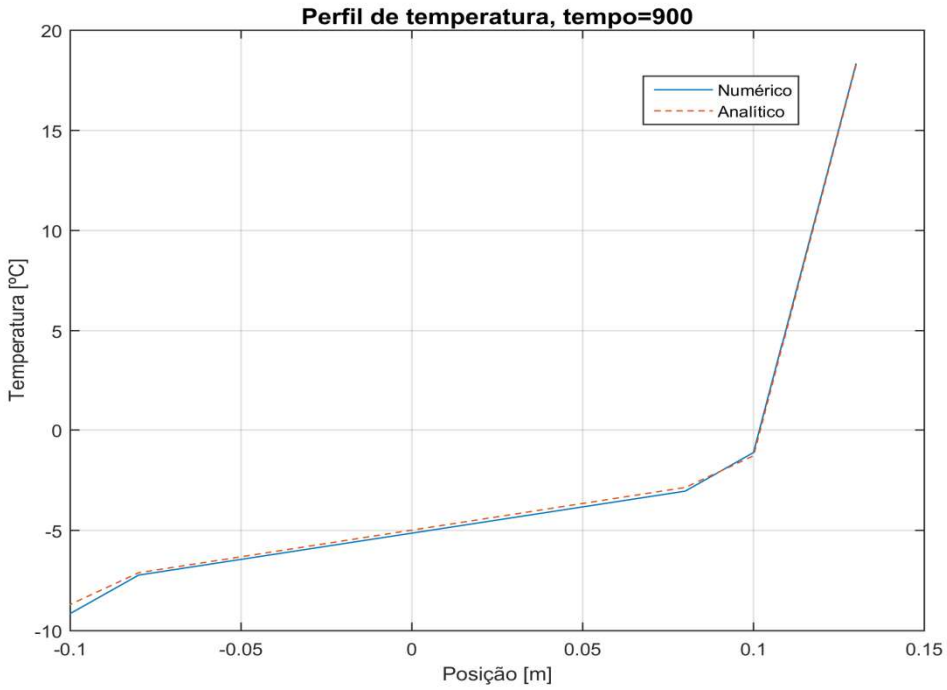

Figura 26 - Comparação numérico vs analítico

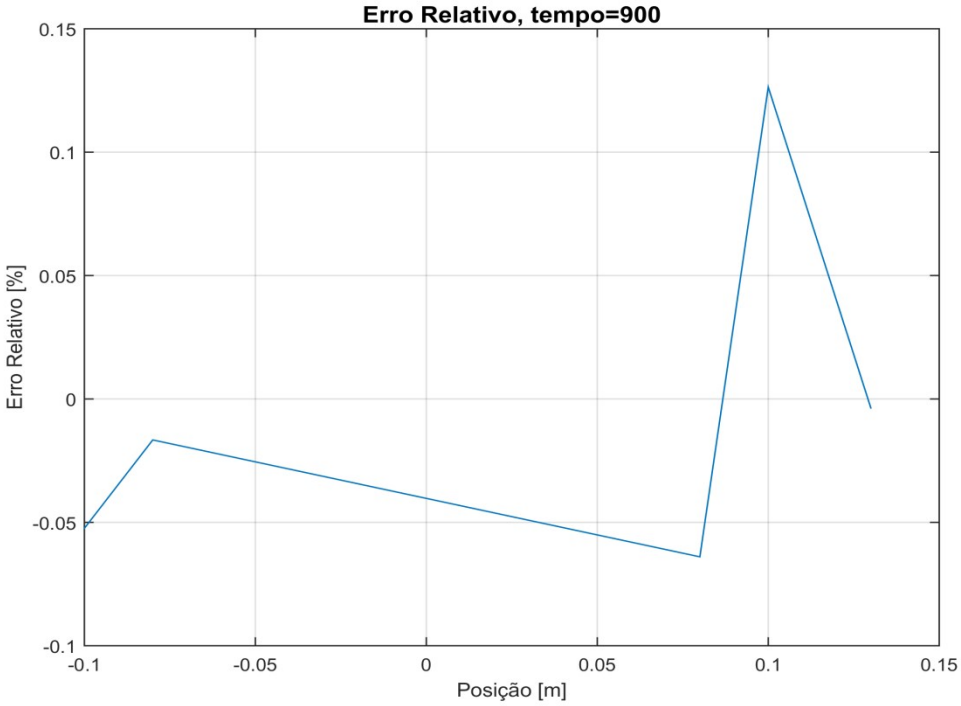

Figura 27 - Erro Relativo

Com estes resultados percebe-se que o código desenvolvido apresenta resultados muito satisfatórios com erro máximo inferior a 0.15% em comparação ao resultado analítico.

Não foi observado perfil de temperatura relevantes nas outras direções devido à similaridades nas propriedades térmicas do gesso e do tijolo.

10. CÓDIGO CVFEM

O código desenvolvido segue a estrutura mostrada na Figura 28. O arquivo inicial que deve ser executado é o "EXE_Codigo_CVFEM_Transiente.m". Este arquivo irá iniciar a seqüência que tem como objetivo último realizar o cálculo segundo o método CVFEM proposto. A etapa de pré processamento é ilustrada em verde e a de processamento em branco

10.1 - PRÉ PROCESSAMENTO

A primeira função acionada é a de ler a malha computacional que deve ser exportado em formato "Generic" do software Gambit na extensão "*.neu". Essa malha é interpretada na função "read_malha3D_ELE()" a qual retorna as principais variáveis P, E, T, Contorno, El_material. Essas variáveis carregam informações sobre a malha computacional tais como coordenadas dos pontos, conectividade, pontos que compõem os elementos, quantidades de faces no contorno, pontos que compõem o contorno e materiais que compõem cada elemento.

Em seguida as principais variáveis que definem as propriedades dos elementos, condições de contorno, condição inicial, passo de tempo e tempo inicial são alocadas.

Na seqüência o código trabalha com a geometria da malha utilizando as informações obtidas anteriormente para formar os volumes de controle e passar informações sobre os elementos para os nós. Também é acionada a função "Areas_Contorno_No_AC()" na qual as áreas dos elementos que se encontram no contorno assim como vetores normais destes e outras informações são calculadas. Após esta função o programa segue para a parte onde os vetores normais ATOS, DSOR e etc das faces dos subvolumes de controle são calculados

Finalmente o último passo dessa etapa de pré-processamento é calcular os coeficientes das funções de forma, e montar os componentes da função de interpolação linear entre os pontos segundo o método proposto.

Caso o usuário tenha selecionado o tempo inicial diferente de zero, o programa irá carregar informações de um arquivo "*.mat" que deve conter as informações no tempo inicial solicitado. E em seguida a etapa de processamento é iniciada.

10.2 – PROCESSAMENTO

Esta etapa é executada enquanto o tempo limite não é alcançado ou pode ser encerrada a qualquer momento pelo usuário utilizando o comando "Ctrl + C" na *Command Window*.do Matlab

As ações que se repetem são:

- Montar a matriz “A” que contêm o termo difusivo utilizando os coeficientes das funções de forma com um laço por todos os nós da malha numérica.

- Montar o termo transiente no vetor “B” e na matriz “A” com um laço por todos os nós da malha numérica.

- Montar as condições de contorno de forma implícita e explicita por um laço por todos os nós presente nos contornos da malha numérica. Esta etapa diferencia cada uma das condições de contorno que devem estar alocadas na matriz “G”.

- Calcular a nova temperatura para todos os pontos da malha numérica

- Salvar variáveis relevantes para a etapa de pós-processamento e/ou retomada dos cálculos para um tempo inicial diferente de zero.

Cada Contorno deve ter um G(i,j) respectivo seguindo a ordem da struct "Contorno" , onde i é o número do contorno.

Caso G(i,1) ==1

A condição é de temperatura prescrita na parede. G(i,2) deve estar a temperatura a temperatura em [ºC] e G(i,3) deve ser 0.

Caso G(i,1)==2

A condição é de fluxo de calor prescrito. G(i,2) deve estar o fluxo de calor em W/m e G(i,3) deve ser 0.

Caso G(i,1)==3

A condição é de fluxo de calor variável com coeficiente de convecção prescrito. G(i,2) deve estar o valor do coeficiente de convecção “h” em W/m^2K e G(i,3) deve estar a temperatura do fluido que interage com o contorno em [ºC]

10.3 – PÓS-PROCESSAMENTO

No código desenvolvido não está presente uma forma padrão para realizar o pós-processamento dos dados obtidos. O pós processamento é realizado de acordo com as características que se deseja investigar sendo cada gráfico obtido de forma personalizada. Os gráficos são gerados a partir dos arquivos salvos na etapa de processamento. Estes são acessados pontualmente para análise de um tempo em específico ou são acessados de forma automática por meio de um laço para análise de gráficos no qual a temperatura varia no tempo.

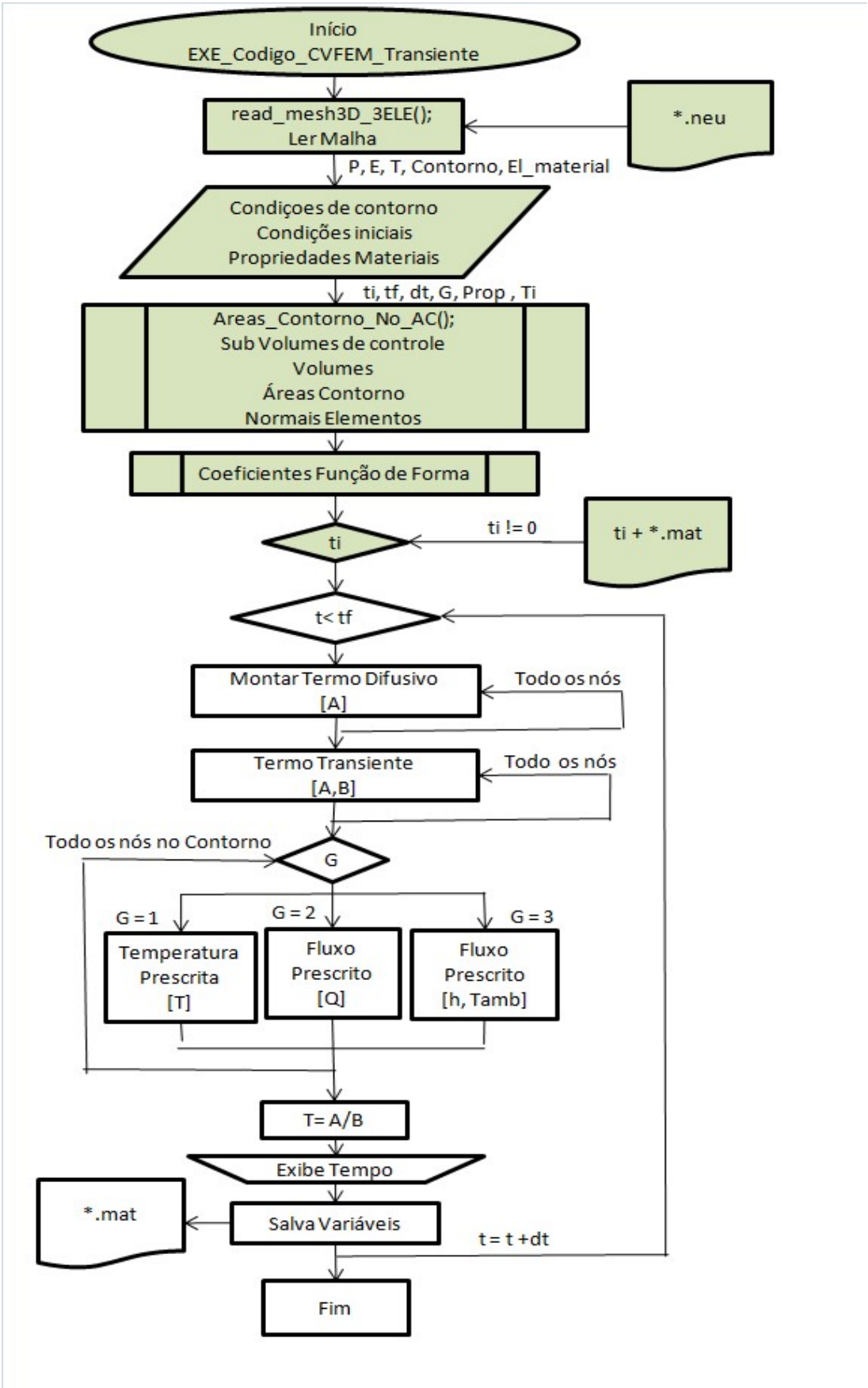

Figura 28 - Fluxograma

11. CONCLUSÃO

Os objetivos deste projeto foram alcançados, foi desenvolvido um código computacional, que utilize malhas 3D de elementos finitos tetraédricos, capaz de resolver problemas de transferência de calor em regime permanente e transiente que considere condições de calor de temperatura prescrita e de fluxo com coeficiente de convecção fixo e problemas de fluxo conjugado.

O código desenvolvido resolveu quatro problemas de transferência de calor: cujas soluções analíticas necessitaram de avançadas ferramentas matemáticas para serem obtidas. Os problemas propostos foram: condução de calor em um cubo unitário em regime constante, condução de calor em um cubo unitário em regime transiente, aquecimento de um ovo e condução de calor por uma parede composta por três materiais diferentes em regime transiente.

Este Trabalho de Conclusão de Curso teve como objetivo principal explorar o método CVFEM desenvolvendo este a partir do zero. Esta abordagem veio a enriquecer meu conhecimento no que diz respeito ao funcionamento de programas CFD e os métodos envolvidos. Diferentemente de se utilizar um software especifico para tal, desenvolver o próprio código é muito mais interessante e desafiador, além de abrir oportunidades para desenvolvimento de outras ferramentas que podem ser úteis na área de pesquisa e desenvolvimento.

Com o desenvolvimento deste código computacional um leque de oportunidades se abre para que diversas análises possam surgir através deste. O método tal como desenvolvido aqui com poucas modificações pode ser aplicado a diversas áreas da engenharia tais como a química, fluidodinâmica ou eletromagnetismo entre outras possibilidades de trabalho.

12. REFERÊNCIAS BIBLIOGRÁFICAS

Cunha, Fábio Alfaia . Modelo Matemático Para Estudo De Processos Reativos De Partículas De Carvão E Biomassa. Tese (doutorado) - Universidade de Brasília, Faculdade de Tecnologia, Departamento de Engenharia Mecânica, 2010.

Maliska, C.R. Transferência De Calor E Mecânica Dos Fluidos Computacional. 2ª edição. Rio de Janeiro : LTC, 2010

Patankar, Sushas V.Numerical Heat Tranfer and Fluid Flow - Series in computational methods in mechanics and thermal sciencies.CRC, 1941

Tannehill, John C , Dale A. Anderson. Computational Fluid Mechanics And Heat Transfer .2ª edição. Taylor & Francis, 1984

Kreith F, Manglik RM, Bohn MS. Principios de transferencia de calor. Cengage Learning Editores; 2011

Dain, Barara L. Muir A Control-Volume Finite Element Method For Three-Dimensional Ellíptic Fluid Flow And Heat Transfer.1983.217. Tese de mestrado. Departmento de Engenhraria mecânica Mc Gill University Montreal, Canada.1983

E. Tombarevé, V. R. Vollerand I. Vušanovié. Detailed CVFEM Algorithm for Three Dimensional Advection-diffusion Problems. Tech Science Press. CMES, vol.96, no.1, pp.1-29, 2013

Wolfram alpha. Computer knowledge engine. Disponível em: <https://www.wolframalpha.com/input/?i=gauss+divergence+theorem> acessado em: 01 de fevereiro de 2017.

Versteeg,H K; Malalasekera, W. An Introduction to computacional fluid dynamics: the finite Volume Method. 2ª edição. England. Pearson Education, 2007.

GAMBIT NEUTRAL FILE FORMAT. Disponível em : <https://www.sharcnet.ca/Software/Gambit/html/users_guide/ug0c.htm> acessado em 08 de novembro de 2014

Zienkiewicz O.C, Mongan K. Finite Elements & Approximation. United States. 1983.

Çengel, Yunus A.Boles, Michael A. Thermodynamics. United States.McGraw Hill. 2003.

White, Frank M. Fluid Mechanics. 4ª edição .United States: McGraw Hill, 1998

Baliga, B. R., and S. V. Patankar. "A new finite-element formulation for convection-diffusion problems." Numerical Heat Transfer 3.4 (1980): 393-409

13. APÊNDICE I – GAMBIT

O programa utilizado para gerar a malha computacional da geometria analisada foi o GAMBIT®. O GAMBIT® é um programa capaz de gerar malhas em 2D e 3D das mais complexas geometrias, utilizando o seu próprio *workbench* para desenho ou utilizando desenhos importados de outros programas específicos para desenhos técnicos em 3D tais como o Catia®, da Dassault Systèmes®, ou o Solid Works® também da Dassault Systèmes®. Neste TCC nenhum dos dois programas de desenho citados acima foi utilizado, pois não havia necessidade para tal, visto a baixa complexidade da geometria estudada. Foi feito uso somente da interface gráfica do Gambit para obter a malha de teste: um cubo de dimensões unitárias.

No final foi possível exportar um arquivo que continha todas as informações que foram produzidas neste processo. O arquivo de interesse tem a extensão *.neu. Um exemplo deste arquivo pode ser visualizado em: https://www.sharcnet.ca/Software/Gambit/html/users_guide/ug0c.htm no tópico C3.Este arquivo contém diversas informações que o usuário por de vir a precisar, porém muitas dessas informações podem ser descartadas do ponto de vista computacional, tais como nomes e linhas, pois o código computacional somente precisa das informações numéricas da malha para interpretar o arquivo de entrada.

No exemplo citado temos que logo abaixo da expressão "*NODAL COORDINATES 1.2.1*" esta presente as coordenadas X, Y e Z de todos os pontos da malha. Estes números são armazenados na variável "P" no código gerado. Em seguida no arquivo *.neu está registrado os nós que compõem cada elemento finito. Abaixo da expressão "*ELEMENTS/CELLS 1.2.1*" temos que cada linha representa um elemento e as colunas 4 em diante estão inscritos os nós que fazem parte do elemento. Este números são armazenado na variável "T" do código computacional.Abaixo da expressão "*BOUNDARY CONDITIONS 1.2.1*" estão os nós que se encontram nas superfícies do elemento. Estes nós são importantes para se determinar as condições de contorno no código. Este estão armazenados na variável "Contorno(ii).N_CC(jj)" onde "ii" simboliza a face e "jj" o número do ponto da face.

A ultima variável "E" do código computacional que simboliza todos os elementos do contorno independentemente das faces onde este se encontram e dos nós que compõem este, é obtida por meio de um pequeno tratamento das variáveis P e T obtidas do arquivo *.neu. Esta variável "E" é necessária para determinar as condições de contorno quando uma condição de contorno de Neumann, ou seja, fluxo prescrito é utilizada nas fronteiras. Este tratamento pode ser observado no programa read_mesh3D_ELE.

14. APÊNDICE II – SCRIPT CVFEM

EXE_Codigo_CVFEM_Transiente.m

```
% Este código foi desenvolvido como parte do Trabalho de Conclusão de
Curso
% para o curso de Engenharia de Energia da Universidade de
Brasília/Campus
% Gama no segundo semestre do ano de 2016
% Entitulado :DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO
DA EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O
MÉTODO DE VOLUMES DE CONTROLE BASEADO EM ELEMENTOS
% por Pedro Paulo Dunice van Els
% sob Orientação do Prof Dr Fabio Alfaia da Cunha

% Este trabalha junto a outros códigos, que devem estar na mesma
pasta.
% Segue a lista:
% read_mesh3D_3ELE.m
% Areas_Contorno_No_AC.m
% Area_triangulo.m
% ATUALIZA_VARIAVEIS.m
% Normal_Triangulo.m
% EXE_Codigo_CVFEM_Transiente.m

clc,clear all;
% CARREGAR A MALHA COMPUTACIONAL
[P,T,E,SAIDA,Contorno,Material]= read_mesh3D_3ELE('modulo.neu');
% CONFIRA SE NA PASTA DE TRABALHO EXISTE UMA PASTA COM O NOME DA
VARIAVEL
% 'destino' ONDE SERÁ SALVO OS ARQUIVOS GERADOS POR ESTE CODIGO
MANTENHA AS
% PASTAS SEMPRE ORGANIZADAS.
destino= 'parede';
X=P(:,1);
Y=P(:,2);
Z=P(:,3);
[ntl ,ntc] = size(T);
[npl ,npc] = size(P);
[nel ,nec] = size(E);

%DEFINIR PROPRIEDADES NOS ELEMENTOS`
Prop.k =  zeros(1,ntl);
Prop.rho= zeros(1,ntl);
Prop.cp=  zeros(1,ntl);
Prop.k(Material(1).El_material) = 0.72 ;%W/mK % TIJOLO BRICK
Prop.k(Material(2).El_material) = 0.22 ;     %GESSO PLASTER
Prop.k(Material(3).El_material) = 0.026 ;     % ESPUMA FOAM

Prop.rho(Material(1).El_material)=1922 ;%kg/m3
Prop.rho(Material(2).El_material)=720;
Prop.rho(Material(3).El_material)=70;

Prop.cp(Material(1).El_material)= 0.79  *1000; %j/kgk
Prop.cp(Material(2).El_material)=1.34*1000 ;
Prop.cp(Material(3).El_material)=1.045*1000;
```

```
Prop.P_k =  zeros(1,npl);
Prop.P_rho= zeros(1,npl);
Prop.P_cp=  zeros(1,npl);


% % DEFINIR CONDICAO DE CONTORNO
% % Cada Contorno deve ter um G(i,j) respectivo seguindo a ordem da
% % struct "Contorno"
% % A forma como esta trabalha G se apresenta da seguinte forma
% % if G(i,1) ==1
% %     temepratura definida na parede
% %     em G(i,2 )deve esta a temperatura a temperatura
% %     G(i,3)deve ser 0
% % if G(i,1)==2
% %     fluxo de calor defino
% %     G(i,2)deve estar o fluxo de calo em W/m
% %     G(i,3)deve ser 0
% % if G(i,1)==3
% %     fluxo de calor variavel com coeficiente de conveccao definido
% %     G(i,2)deve estar o valor do coeficiente de convecao h em
W/m^2K
% %     G(i,3)deve esta a temperatura do fluido que interage com o
contorno
% %
G=[2 0 0
    2 0 0
    3 25 -10
    3 10 20];


%DEFINIR CONDICOES INICIAIS E DE OPERACAO
Tinicial = 5;%200% 5; %K
VS.Temp= ones(npl,2) * Tinicial;

VS.tempo(1)=0;
VS.dt =50;
ti= 0; %s
tf=100000;%s

% atribuir matriz de conectividade, cada linha é um elemento
T1=T(:,1);
T2=T(:,2);
T3=T(:,3);
T4=T(:,4);
% % % MONTAR VOLUMES DE CONTROLE E PROCESSAR MALHA NUMÉRICA
% % %atribui 1 Geo.X1,Geo.Y1,Geo.Z1 para cada ponto1,2,3,4 de um
elemento
% % % Geo. stands for Geometria
Geo.X1=X(T1);
Geo.X2=X(T2);
Geo.X3=X(T3);
Geo.X4=X(T4);

Geo.Y1=Y(T1);
Geo.Y2=Y(T2);
Geo.Y3=Y(T3);
Geo.Y4=Y(T4);

Geo.Z1=Z(T1);
Geo.Z2=Z(T2);
```

```
Geo.Z3=Z(T3);
Geo.Z4=Z(T4);

%definir o centroide do elementro
Geo.Xo=(Geo.X1+Geo.X2+Geo.X3+Geo.X4)/4;
Geo.Yo=(Geo.Y1+Geo.Y2+Geo.Y3+Geo.Y4)/4;
Geo.Zo=(Geo.Z1+Geo.Z2+Geo.Z3+Geo.Z4)/4;
% definir centroide das faces

Geo.Xr= (Geo.X1+Geo.X3+Geo.X4)/3;
Geo.Yr= (Geo.Y1+Geo.Y3+Geo.Y4)/3;
Geo.Zr= (Geo.Z1+Geo.Z3+Geo.Z4)/3;

Geo.Xs= (Geo.X1+Geo.X2+Geo.X4)/3;
Geo.Ys= (Geo.Y1+Geo.Y2+Geo.Y4)/3;
Geo.Zs= (Geo.Z1+Geo.Z2+Geo.Z4)/3;

Geo.Xt= (Geo.X1+Geo.X2+Geo.X3)/3;
Geo.Yt= (Geo.Y1+Geo.Y2+Geo.Y3)/3;
Geo.Zt= (Geo.Z1+Geo.Z2+Geo.Z3)/3;

Geo.Xq= (Geo.X2+Geo.X3+Geo.X4)/3;
Geo.Yq= (Geo.Y2+Geo.Y3+Geo.Y4)/3;
Geo.Zq= (Geo.Z2+Geo.Z3+Geo.Z4)/3;
%defionir valores médios nas arestas entre os pontos ABCD do tetraedro
Geo.Xa=(Geo.X1+Geo.X2)/2;
Geo.Ya=(Geo.Y1+Geo.Y2)/2;
Geo.Za=(Geo.Z1+Geo.Z2)/2;

Geo.Xb=(Geo.X3+Geo.X2)/2;
Geo.Yb=(Geo.Y3+Geo.Y2)/2;
Geo.Zb=(Geo.Z3+Geo.Z2)/2;

Geo.Xc=(Geo.X1+Geo.X3)/2;
Geo.Yc=(Geo.Y1+Geo.Y3)/2;
Geo.Zc=(Geo.Z1+Geo.Z3)/2;

Geo.Xd=(Geo.X1+Geo.X4)/2;
Geo.Yd=(Geo.Y1+Geo.Y4)/2;
Geo.Zd=(Geo.Z1+Geo.Z4)/2;

Geo.Xe=(Geo.X2+Geo.X4)/2;
Geo.Ye=(Geo.Y2+Geo.Y4)/2;
Geo.Ze=(Geo.Z2+Geo.Z4)/2;

Geo.Xf=(Geo.X3+Geo.X4)/2;
Geo.Yf=(Geo.Y3+Geo.Y4)/2;
Geo.Zf=(Geo.Z3+Geo.Z4)/2;



somaV=0;
VS.V= ones(ntl,1);
% definir vetores normais para os sub volumes de controle
for i =1:ntl
    Matriz_V= [ Geo.X1(i) Geo.Y1(i) Geo.Z1(i) 1
        Geo.X2(i) Geo.Y2(i) Geo.Z2(i) 1
        Geo.X3(i) Geo.Y3(i) Geo.Z3(i) 1
        Geo.X4(i) Geo.Y4(i) Geo.Z4(i) 1];
    VS.V(i) = abs(det( Matriz_V))/6;
```

```
    somaV=somaV+ VS.V(i);
end

SV.c1=VS.V/4; SV.c2=VS.V/4;  SV.c3=VS.V/4; SV.c4=VS.V/4;

V_no=zeros(length(P),1);
for i=1:length(T);
    V_no(T1(i))=SV.c1(i)+V_no(T1(i));
    V_no(T2(i))=SV.c2(i)+V_no(T2(i));
    V_no(T3(i))=SV.c3(i)+V_no(T3(i));
    V_no(T4(i))=SV.c4(i)+V_no(T4(i));
end

%passar as propriedades dos elementos Para os pontos
% para utilizar no termo transiente e nas condiçoes de contorno
for i=1:ntl
    Prop.P_k(T1(i))= Prop.P_k(T1(i)) + Prop.k(i) * SV.c1(i);
    Prop.P_k(T2(i))= Prop.P_k(T2(i)) + Prop.k(i) * SV.c2(i);
    Prop.P_k(T3(i))= Prop.P_k(T3(i)) + Prop.k(i) * SV.c3(i);
    Prop.P_k(T4(i))= Prop.P_k(T4(i)) + Prop.k(i) * SV.c4(i);


    Prop.P_cp(T1(i))= Prop.P_cp(T1(i)) + Prop.cp(i) * SV.c1(i);
    Prop.P_cp(T2(i))= Prop.P_cp(T2(i)) + Prop.cp(i) * SV.c2(i);
    Prop.P_cp(T3(i))= Prop.P_cp(T3(i)) + Prop.cp(i) * SV.c3(i);
    Prop.P_cp(T4(i))= Prop.P_cp(T4(i)) + Prop.cp(i) * SV.c4(i);


    Prop.P_rho(T1(i))= Prop.P_rho(T1(i)) + Prop.rho(i) * SV.c1(i);
    Prop.P_rho(T2(i))= Prop.P_rho(T2(i)) + Prop.rho(i) * SV.c2(i);
    Prop.P_rho(T3(i))= Prop.P_rho(T3(i)) + Prop.rho(i) * SV.c3(i);
    Prop.P_rho(T4(i))= Prop.P_rho(T4(i)) + Prop.rho(i) * SV.c4(i);

end
for i=1: npl
    Prop.P_k (i) = Prop.P_k (i) / V_no(i);
    Prop.P_cp(i)  = Prop.P_cp (i)/ V_no(i);
    Prop.P_rho (i)= Prop.P_rho(i)/ V_no(i);
end

Areas_Contorno_No_AC();
% CALCULO DOS VETORES NORMAIS ATOS , DSOR CTOR E ETC PARA TODOS OS
% SUBVOLUMES DE CONTROLE
tic
Nor.n1a = [ ((Geo.Ys - Geo.Yt).*(Geo.Za - Geo.Zo))/2 - ((Geo.Ya -
Geo.Yo).*(Geo.Zs - Geo.Zt))/2, ((Geo.Xa - Geo.Xo).*(Geo.Zs -
Geo.Zt))/2 - ((Geo.Xs - Geo.Xt).*(Geo.Za - Geo.Zo))/2, ((Geo.Xs -
Geo.Xt).*(Geo.Ya - Geo.Yo))/2 - ((Geo.Xa - Geo.Xo).*(Geo.Ys -
Geo.Yt))/2];
Nor.n1d = [ ((Geo.Yr - Geo.Ys).*(Geo.Zd - Geo.Zo))/2 - ((Geo.Yd -
Geo.Yo).*(Geo.Zr - Geo.Zs))/2, ((Geo.Xd - Geo.Xo).*(Geo.Zr -
Geo.Zs))/2 - ((Geo.Xr - Geo.Xs).*(Geo.Zd - Geo.Zo))/2, ((Geo.Xr -
Geo.Xs).*(Geo.Yd - Geo.Yo))/2 - ((Geo.Xd - Geo.Xo).*(Geo.Yr -
Geo.Ys))/2];
Nor.n1c = [ ((Geo.Yc - Geo.Yo).*(Geo.Zr - Geo.Zt))/2 - ((Geo.Yr -
Geo.Yt).*(Geo.Zc - Geo.Zo))/2, ((Geo.Xr - Geo.Xt).*(Geo.Zc -
Geo.Zo))/2 - ((Geo.Xc - Geo.Xo).*(Geo.Zr - Geo.Zt))/2, ((Geo.Xc -
Geo.Xo).*(Geo.Yr - Geo.Yt))/2 - ((Geo.Xr - Geo.Xt).*(Geo.Yc -
Geo.Yo))/2];
```

```
Nor.n2b = [ ((Geo.Yq - Geo.Yt).*(Geo.Zb - Geo.Zo))/2 - ((Geo.Yb -
Geo.Yo).*(Geo.Zq - Geo.Zt))/2, ((Geo.Xb - Geo.Xo).*(Geo.Zq -
Geo.Zt))/2 - ((Geo.Xq - Geo.Xt).*(Geo.Zb - Geo.Zo))/2, ((Geo.Xq -
Geo.Xt).*(Geo.Yb - Geo.Yo))/2 - ((Geo.Xb - Geo.Xo).*(Geo.Yq -
Geo.Yt))/2];
Nor.n2e = [ ((Geo.Ye - Geo.Yo).*(Geo.Zq - Geo.Zs))/2 - ((Geo.Yq -
Geo.Ys).*(Geo.Ze - Geo.Zo))/2, ((Geo.Xq - Geo.Xs).*(Geo.Ze -
Geo.Zo))/2 - ((Geo.Xe - Geo.Xo).*(Geo.Zq - Geo.Zs))/2, ((Geo.Xe -
Geo.Xo).*(Geo.Yq - Geo.Ys))/2 - ((Geo.Xq - Geo.Xs).*(Geo.Ye -
Geo.Yo))/2];
Nor.n2a = [ ((Geo.Ya - Geo.Yo).*(Geo.Zs - Geo.Zt))/2 - ((Geo.Ys -
Geo.Yt).*(Geo.Za - Geo.Zo))/2, ((Geo.Xs - Geo.Xt).*(Geo.Za -
Geo.Zo))/2 - ((Geo.Xa - Geo.Xo).*(Geo.Zs - Geo.Zt))/2, ((Geo.Xa -
Geo.Xo).*(Geo.Ys - Geo.Yt))/2 - ((Geo.Xs - Geo.Xt).*(Geo.Ya -
Geo.Yo))/2];
Nor.n3c = [ ((Geo.Yr - Geo.Yt).*(Geo.Zc - Geo.Zo))/2 - ((Geo.Yc -
Geo.Yo).*(Geo.Zr - Geo.Zt))/2, ((Geo.Xc - Geo.Xo).*(Geo.Zr -
Geo.Zt))/2 - ((Geo.Xr - Geo.Xt).*(Geo.Zc - Geo.Zo))/2, ((Geo.Xr -
Geo.Xt).*(Geo.Yc - Geo.Yo))/2 - ((Geo.Xc - Geo.Xo).*(Geo.Yr -
Geo.Yt))/2];
Nor.n3f = [ ((Geo.Yq - Geo.Yr).*(Geo.Zf - Geo.Zo))/2 - ((Geo.Yf -
Geo.Yo).*(Geo.Zq - Geo.Zr))/2, ((Geo.Xf - Geo.Xo).*(Geo.Zq -
Geo.Zr))/2 - ((Geo.Xq - Geo.Xr).*(Geo.Zf - Geo.Zo))/2, ((Geo.Xq -
Geo.Xr).*(Geo.Yf - Geo.Yo))/2 - ((Geo.Xf - Geo.Xo).*(Geo.Yq -
Geo.Yr))/2];
Nor.n3b = [ ((Geo.Yb - Geo.Yo).*(Geo.Zq - Geo.Zt))/2 - ((Geo.Yq -
Geo.Yt).*(Geo.Zb - Geo.Zo))/2, ((Geo.Xq - Geo.Xt).*(Geo.Zb -
Geo.Zo))/2 - ((Geo.Xb - Geo.Xo).*(Geo.Zq - Geo.Zt))/2, ((Geo.Xb -
Geo.Xo).*(Geo.Yq - Geo.Yt))/2 - ((Geo.Xq - Geo.Xt).*(Geo.Yb -
Geo.Yo))/2];
Nor.n4e = [ ((Geo.Yq - Geo.Ys).*(Geo.Ze - Geo.Zo))/2 - ((Geo.Ye -
Geo.Yo).*(Geo.Zq - Geo.Zs))/2, ((Geo.Xe - Geo.Xo).*(Geo.Zq -
Geo.Zs))/2 - ((Geo.Xq - Geo.Xs).*(Geo.Ze - Geo.Zo))/2, ((Geo.Xq -
Geo.Xs).*(Geo.Ye - Geo.Yo))/2 - ((Geo.Xe - Geo.Xo).*(Geo.Yq -
Geo.Ys))/2];
Nor.n4f = [ ((Geo.Yf - Geo.Yo).*(Geo.Zq - Geo.Zr))/2 - ((Geo.Yq -
Geo.Yr).*(Geo.Zf - Geo.Zo))/2, ((Geo.Xq - Geo.Xr).*(Geo.Zf -
Geo.Zo))/2 - ((Geo.Xf - Geo.Xo).*(Geo.Zq - Geo.Zr))/2, ((Geo.Xf -
Geo.Xo).*(Geo.Yq - Geo.Yr))/2 - ((Geo.Xq - Geo.Xr).*(Geo.Yf -
Geo.Yo))/2];
Nor.n4d = [ ((Geo.Yd - Geo.Yo).*(Geo.Zr - Geo.Zs))/2 - ((Geo.Yr -
Geo.Ys).*(Geo.Zd - Geo.Zo))/2, ((Geo.Xr - Geo.Xs).*(Geo.Zd -
Geo.Zo))/2 - ((Geo.Xd - Geo.Xo).*(Geo.Zr - Geo.Zs))/2, ((Geo.Xd -
Geo.Xo).*(Geo.Yr - Geo.Ys))/2 - ((Geo.Xr - Geo.Xs).*(Geo.Yd -
Geo.Yo))/2];

TempoNormal = toc ;

%CALCULO DOS COEFICIENTES FUNCAO DE FORMA
tic
CFF.a1=ones(ntl,1);
CFF.a2=ones(ntl,1);
CFF.a3=ones(ntl,1);
CFF.a4=ones(ntl,1);

CFF.b1=ones(ntl,1);
CFF.b2=ones(ntl,1);
CFF.b3=ones(ntl,1);
CFF.b4=ones(ntl,1);

CFF.c1=ones(ntl,1);
```

```
CFF.c2=ones(ntl,1);
CFF.c3=ones(ntl,1);
CFF.c4=ones(ntl,1);

CFF.d1=ones(ntl,1);
CFF.d2=ones(ntl,1);
CFF.d3=ones(ntl,1);
CFF.d4=ones(ntl,1);

for i =1:ntl %calculo dos coeficiente
    CFF.a1(i)= -(Geo.X2(i)*Geo.Y4(i)*Geo.Z3(i) -
Geo.X2(i)*Geo.Y3(i)*Geo.Z4(i) + Geo.X3(i)*Geo.Y2(i)*Geo.Z4(i) -
Geo.X3(i)*Geo.Y4(i)*Geo.Z2(i) - Geo.X4(i)*Geo.Y2(i)*Geo.Z3(i) +
Geo.X4(i)*Geo.Y3(i)*Geo.Z2(i));
    CFF.a2(i)= -(Geo.X1(i)*Geo.Y3(i)*Geo.Z4(i) -
Geo.X1(i)*Geo.Y4(i)*Geo.Z3(i) - Geo.X3(i)*Geo.Y1(i)*Geo.Z4(i) +
Geo.X3(i)*Geo.Y4(i)*Geo.Z1(i) + Geo.X4(i)*Geo.Y1(i)*Geo.Z3(i) -
Geo.X4(i)*Geo.Y3(i)*Geo.Z1(i));
    CFF.a3(i)= -(Geo.X1(i)*Geo.Y4(i)*Geo.Z2(i) -
Geo.X1(i)*Geo.Y2(i)*Geo.Z4(i) + Geo.X2(i)*Geo.Y1(i)*Geo.Z4(i) -
Geo.X2(i)*Geo.Y4(i)*Geo.Z1(i) - Geo.X4(i)*Geo.Y1(i)*Geo.Z2(i) +
Geo.X4(i)*Geo.Y2(i)*Geo.Z1(i));
    CFF.a4(i)= -(Geo.X1(i)*Geo.Y2(i)*Geo.Z3(i) -
Geo.X1(i)*Geo.Y3(i)*Geo.Z2(i) - Geo.X2(i)*Geo.Y1(i)*Geo.Z3(i) +
Geo.X2(i)*Geo.Y3(i)*Geo.Z1(i) + Geo.X3(i)*Geo.Y1(i)*Geo.Z2(i) -
Geo.X3(i)*Geo.Y2(i)*Geo.Z1(i));

    CFF.b1(i)= -(Geo.Y2(i)*Geo.Z3(i) - Geo.Y3(i)*Geo.Z2(i) -
Geo.Y2(i)*Geo.Z4(i) + Geo.Y4(i)*Geo.Z2(i) + Geo.Y3(i)*Geo.Z4(i) -
Geo.Y4(i)*Geo.Z3(i));
    CFF.b2(i)= -(Geo.Y3(i)*Geo.Z1(i) - Geo.Y1(i)*Geo.Z3(i) +
Geo.Y1(i)*Geo.Z4(i) - Geo.Y4(i)*Geo.Z1(i) - Geo.Y3(i)*Geo.Z4(i) +
Geo.Y4(i)*Geo.Z3(i));
    CFF.b3(i)= -(Geo.Y1(i)*Geo.Z2(i) - Geo.Y2(i)*Geo.Z1(i) -
Geo.Y1(i)*Geo.Z4(i) + Geo.Y4(i)*Geo.Z1(i) + Geo.Y2(i)*Geo.Z4(i) -
Geo.Y4(i)*Geo.Z2(i));
    CFF.b4(i)= -(Geo.Y2(i)*Geo.Z1(i) - Geo.Y1(i)*Geo.Z2(i) +
Geo.Y1(i)*Geo.Z3(i) - Geo.Y3(i)*Geo.Z1(i) - Geo.Y2(i)*Geo.Z3(i) +
Geo.Y3(i)*Geo.Z2(i));

    CFF.c1(i)= -(Geo.X3(i)*Geo.Z2(i) - Geo.X2(i)*Geo.Z3(i) +
Geo.X2(i)*Geo.Z4(i) - Geo.X4(i)*Geo.Z2(i) - Geo.X3(i)*Geo.Z4(i) +
Geo.X4(i)*Geo.Z3(i));
    CFF.c2(i)= -(Geo.X1(i)*Geo.Z3(i) - Geo.X3(i)*Geo.Z1(i) -
Geo.X1(i)*Geo.Z4(i) + Geo.X4(i)*Geo.Z1(i) + Geo.X3(i)*Geo.Z4(i) -
Geo.X4(i)*Geo.Z3(i));
    CFF.c3(i)= -(Geo.X2(i)*Geo.Z1(i) - Geo.X1(i)*Geo.Z2(i) +
Geo.X1(i)*Geo.Z4(i) - Geo.X4(i)*Geo.Z1(i) - Geo.X2(i)*Geo.Z4(i) +
Geo.X4(i)*Geo.Z2(i));
    CFF.c4(i)= -(Geo.X1(i)*Geo.Z2(i) - Geo.X2(i)*Geo.Z1(i) -
Geo.X1(i)*Geo.Z3(i) + Geo.X3(i)*Geo.Z1(i) + Geo.X2(i)*Geo.Z3(i) -
Geo.X3(i)*Geo.Z2(i));

    CFF.d1(i)= -(Geo.X2(i)*Geo.Y3(i) - Geo.X3(i)*Geo.Y2(i) -
Geo.X2(i)*Geo.Y4(i) + Geo.X4(i)*Geo.Y2(i) + Geo.X3(i)*Geo.Y4(i) -
Geo.X4(i)*Geo.Y3(i));
    CFF.d2(i)= -(Geo.X3(i)*Geo.Y1(i) - Geo.X1(i)*Geo.Y3(i) +
Geo.X1(i)*Geo.Y4(i) - Geo.X4(i)*Geo.Y1(i) - Geo.X3(i)*Geo.Y4(i) +
Geo.X4(i)*Geo.Y3(i));
```

```
    CFF.d3(i)= -(Geo.X1(i)*Geo.Y2(i) - Geo.X2(i)*Geo.Y1(i) -
Geo.X1(i)*Geo.Y4(i) + Geo.X4(i)*Geo.Y1(i) + Geo.X2(i)*Geo.Y4(i) -
Geo.X4(i)*Geo.Y2(i));
    CFF.d4(i)= -(Geo.X2(i)*Geo.Y1(i) - Geo.X1(i)*Geo.Y2(i) +
Geo.X1(i)*Geo.Y3(i) - Geo.X3(i)*Geo.Y1(i) - Geo.X2(i)*Geo.Y3(i) +
Geo.X3(i)*Geo.Y2(i));
end
TempoCoe= toc;

%%%%%testes coeficientes
for i =1: ntl
    D.V11(i) = CFF.a1(i) + CFF.b1(i)*Geo.X1(i) + CFF.c1(i) * Geo.Y1(i)
+ CFF.d1(i)*Geo.Z1(i);
    D.V12(i) = CFF.a2(i) + CFF.b2(i)*Geo.X1(i) + CFF.c2(i) * Geo.Y1(i)
+ CFF.d2(i)*Geo.Z1(i);
    D.V13(i) = CFF.a3(i) + CFF.b3(i)*Geo.X1(i) + CFF.c3(i) * Geo.Y1(i)
+ CFF.d3(i)*Geo.Z1(i);
    D.V14(i) = CFF.a4(i) + CFF.b4(i)*Geo.X1(i) + CFF.c4(i) * Geo.Y1(i)
+ CFF.d4(i)*Geo.Z1(i);
    D.SV1(i) =D.V11(i)+D.V12(i)+D.V13(i)+D.V14(i);
end

for i =1: ntl
    D.V21(i) = CFF.a1(i) + CFF.b1(i)*Geo.X2(i) + CFF.c1(i) * Geo.Y2(i)
+ CFF.d1(i)*Geo.Z2(i);
    D.V22(i) = CFF.a2(i) + CFF.b2(i)*Geo.X2(i) + CFF.c2(i) * Geo.Y2(i)
+ CFF.d2(i)*Geo.Z2(i);
    D.V23(i) = CFF.a3(i) + CFF.b3(i)*Geo.X2(i) + CFF.c3(i) * Geo.Y2(i)
+ CFF.d3(i)*Geo.Z2(i);
    D.V24(i) = CFF.a4(i) + CFF.b4(i)*Geo.X2(i) + CFF.c4(i) * Geo.Y2(i)
+ CFF.d4(i)*Geo.Z2(i);
    D.SV2(i) = D.V21(i)+D.V22(i)+D.V23(i)+D.V24(i);
end

for i =1: ntl
    D.V31(i) = CFF.a1(i) + CFF.b1(i)*Geo.X3(i) + CFF.c1(i) * Geo.Y3(i)
+ CFF.d1(i)*Geo.Z3(i);
    D.V32(i) = CFF.a2(i) + CFF.b2(i)*Geo.X3(i) + CFF.c2(i) * Geo.Y3(i)
+ CFF.d2(i)*Geo.Z3(i);
    D.V33(i) = CFF.a3(i) + CFF.b3(i)*Geo.X3(i) + CFF.c3(i) * Geo.Y3(i)
+ CFF.d3(i)*Geo.Z3(i);
    D.V34(i) = CFF.a4(i) + CFF.b4(i)*Geo.X3(i) + CFF.c4(i) * Geo.Y3(i)
+ CFF.d4(i)*Geo.Z3(i);
    D.SV3(i) = D.V31(i) +D.V32(i) +D.V33(i) +D.V34(i);
end

for i =1: ntl
    D.V41(i) = CFF.a1(i) + CFF.b1(i)*Geo.X4(i) + CFF.c1(i) * Geo.Y4(i)
+ CFF.d1(i)*Geo.Z4(i);
    D.V42(i) = CFF.a2(i) + CFF.b2(i)*Geo.X4(i) + CFF.c2(i) * Geo.Y4(i)
+ CFF.d2(i)*Geo.Z4(i);
    D.V43(i) = CFF.a3(i) + CFF.b3(i)*Geo.X4(i) + CFF.c3(i) * Geo.Y4(i)
+ CFF.d3(i)*Geo.Z4(i);
    D.V44(i) = CFF.a4(i) + CFF.b4(i)*Geo.X4(i) + CFF.c4(i) * Geo.Y4(i)
+ CFF.d4(i)*Geo.Z4(i);
    D.SV4(i) = D.V41(i)+ D.V42(i)+ D.V43(i)+ D.V44(i);
end

%Este teste em desacordo com a equacao 6.36 do TCCFF.c1 nao volta
retorna
```

```
%valores unitarios
%porem os valores Vo1, Vo2.... sao iguais

for i =1: ntl
    D.Vo1(i) = CFF.a1(i) + CFF.b1(i)*Geo.Xo(i) + CFF.c1(i) * Geo.Yo(i)
+ CFF.d1(i)*Geo.Zo(i);
    D.Vo2(i) = CFF.a2(i) + CFF.b2(i)*Geo.Xo(i) + CFF.c2(i) * Geo.Yo(i)
+ CFF.d2(i)*Geo.Zo(i);
    D.Vo3(i) = CFF.a3(i) + CFF.b3(i)*Geo.Xo(i) + CFF.c3(i) * Geo.Yo(i)
+ CFF.d3(i)*Geo.Zo(i);
    D.Vo4(i) = CFF.a4(i) + CFF.b4(i)*Geo.Xo(i) + CFF.c4(i) * Geo.Yo(i)
+ CFF.d4(i)*Geo.Zo(i);
    D.So1(i) =D.Vo1(i)+D.Vo2(i)+D.Vo3(i)+D.Vo4(i);
end

%Termos de intergracao Difusivo D
%Nor.n1a(:,1)     -   componente X vetor normal da face entre de 1 a A
%Nor.n1a(:,2)     -   componente y vetor normal da face entre de 1 a A
%Nor.n1a(:,3)     -   componente Z vetor normal da face entre de 1 a A
%CFF.b1, CFF.c1 ,CFF.d1 -   coeficiente das funcoes de forma
tic
D.D11 = ( CFF.b1.*Nor.n1a(:,1) + CFF.c1.*Nor.n1a(:,2) +
CFF.d1.*Nor.n1a(:,3) + CFF.b1.*Nor.n1c(:,1) +CFF.c1.*Nor.n1c(:,2)
+CFF.d1.*Nor.n1c(:,3) + CFF.b1.*Nor.n1d(:,1) + CFF.c1.*Nor.n1d(:,2) +
CFF.d1.*Nor.n1d(:,3))./(VS.V *6);
D.D12 = ( CFF.b2.*Nor.n1a(:,1) + CFF.c2.*Nor.n1a(:,2) +
CFF.d2.*Nor.n1a(:,3) + CFF.b2.*Nor.n1c(:,1) +CFF.c2.*Nor.n1c(:,2)
+CFF.d2.*Nor.n1c(:,3) + CFF.b2.*Nor.n1d(:,1) + CFF.c2.*Nor.n1d(:,2) +
CFF.d2.*Nor.n1d(:,3))./(VS.V *6);
D.D13 = ( CFF.b3.*Nor.n1a(:,1) + CFF.c3.*Nor.n1a(:,2) +
CFF.d3.*Nor.n1a(:,3) + CFF.b3.*Nor.n1c(:,1) +CFF.c3.*Nor.n1c(:,2)
+CFF.d3.*Nor.n1c(:,3) + CFF.b3.*Nor.n1d(:,1) + CFF.c3.*Nor.n1d(:,2) +
CFF.d3.*Nor.n1d(:,3))./(VS.V *6);
D.D14 = ( CFF.b4.*Nor.n1a(:,1) + CFF.c4.*Nor.n1a(:,2) +
CFF.d4.*Nor.n1a(:,3) + CFF.b4.*Nor.n1c(:,1) +CFF.c4.*Nor.n1c(:,2)
+CFF.d4.*Nor.n1c(:,3) + CFF.b4.*Nor.n1d(:,1) + CFF.c4.*Nor.n1d(:,2) +
CFF.d4.*Nor.n1d(:,3))./(VS.V *6);
D.D21 = ( CFF.b1.*Nor.n2a(:,1) + CFF.c1.*Nor.n2a(:,2) +
CFF.d1.*Nor.n2a(:,3) + CFF.b1.*Nor.n2b(:,1) +CFF.c1.*Nor.n2b(:,2)
+CFF.d1.*Nor.n2b(:,3) + CFF.b1.*Nor.n2e(:,1) + CFF.c1.*Nor.n2e(:,2) +
CFF.d1.*Nor.n2e(:,3))./(VS.V *6);
D.D22 = ( CFF.b2.*Nor.n2a(:,1) + CFF.c2.*Nor.n2a(:,2) +
CFF.d2.*Nor.n2a(:,3) + CFF.b2.*Nor.n2b(:,1) +CFF.c2.*Nor.n2b(:,2)
+CFF.d2.*Nor.n2b(:,3) + CFF.b2.*Nor.n2e(:,1) + CFF.c2.*Nor.n2e(:,2) +
CFF.d2.*Nor.n2e(:,3))./(VS.V *6);
D.D23 = ( CFF.b3.*Nor.n2a(:,1) + CFF.c3.*Nor.n2a(:,2) +
CFF.d3.*Nor.n2a(:,3) + CFF.b3.*Nor.n2b(:,1) +CFF.c3.*Nor.n2b(:,2)
+CFF.d3.*Nor.n2b(:,3) + CFF.b3.*Nor.n2e(:,1) + CFF.c3.*Nor.n2e(:,2) +
CFF.d3.*Nor.n2e(:,3))./(VS.V *6);
D.D24 = ( CFF.b4.*Nor.n2a(:,1) + CFF.c4.*Nor.n2a(:,2) +
CFF.d4.*Nor.n2a(:,3) + CFF.b4.*Nor.n2b(:,1) +CFF.c4.*Nor.n2b(:,2)
+CFF.d4.*Nor.n2b(:,3) + CFF.b4.*Nor.n2e(:,1) + CFF.c4.*Nor.n2e(:,2) +
CFF.d4.*Nor.n2e(:,3))./(VS.V *6);
D.D31 = ( CFF.b1.*Nor.n3b(:,1) + CFF.c1.*Nor.n3b(:,2) +
CFF.d1.*Nor.n3b(:,3) + CFF.b1.*Nor.n3c(:,1) +CFF.c1.*Nor.n3c(:,2)
+CFF.d1.*Nor.n3c(:,3) + CFF.b1.*Nor.n3f(:,1) + CFF.c1.*Nor.n3f(:,2) +
CFF.d1.*Nor.n3f(:,3))./(VS.V *6);
D.D32 = ( CFF.b2.*Nor.n3b(:,1) + CFF.c2.*Nor.n3b(:,2) +
CFF.d2.*Nor.n3b(:,3) + CFF.b2.*Nor.n3c(:,1) +CFF.c2.*Nor.n3c(:,2)
+CFF.d2.*Nor.n3c(:,3) + CFF.b2.*Nor.n3f(:,1) + CFF.c2.*Nor.n3f(:,2) +
CFF.d2.*Nor.n3f(:,3))./(VS.V *6);
```

```
D.D33 = ( CFF.b3.*Nor.n3b(:,1) + CFF.c3.*Nor.n3b(:,2) +
CFF.d3.*Nor.n3b(:,3) + CFF.b3.*Nor.n3c(:,1) +CFF.c3.*Nor.n3c(:,2)
+CFF.d3.*Nor.n3c(:,3) + CFF.b3.*Nor.n3f(:,1) + CFF.c3.*Nor.n3f(:,2) +
CFF.d3.*Nor.n3f(:,3))./(VS.V *6);
D.D34 = ( CFF.b4.*Nor.n3b(:,1) + CFF.c4.*Nor.n3b(:,2) +
CFF.d4.*Nor.n3b(:,3) + CFF.b4.*Nor.n3c(:,1) +CFF.c4.*Nor.n3c(:,2)
+CFF.d4.*Nor.n3c(:,3) + CFF.b4.*Nor.n3f(:,1) + CFF.c4.*Nor.n3f(:,2) +
CFF.d4.*Nor.n3f(:,3))./(VS.V *6);
D.D41 = ( CFF.b1.*Nor.n4d(:,1) + CFF.c1.*Nor.n4d(:,2) +
CFF.d1.*Nor.n4d(:,3) + CFF.b1.*Nor.n4e(:,1) +CFF.c1.*Nor.n4e(:,2)
+CFF.d1.*Nor.n4e(:,3) + CFF.b1.*Nor.n4f(:,1) + CFF.c1.*Nor.n4f(:,2) +
CFF.d1.*Nor.n4f(:,3))./(VS.V *6);
D.D42 = ( CFF.b2.*Nor.n4d(:,1) + CFF.c2.*Nor.n4d(:,2) +
CFF.d2.*Nor.n4d(:,3) + CFF.b2.*Nor.n4e(:,1) +CFF.c2.*Nor.n4e(:,2)
+CFF.d2.*Nor.n4e(:,3) + CFF.b2.*Nor.n4f(:,1) + CFF.c2.*Nor.n4f(:,2) +
CFF.d2.*Nor.n4f(:,3))./(VS.V *6);
D.D43 = ( CFF.b3.*Nor.n4d(:,1) + CFF.c3.*Nor.n4d(:,2) +
CFF.d3.*Nor.n4d(:,3) + CFF.b3.*Nor.n4e(:,1) +CFF.c3.*Nor.n4e(:,2)
+CFF.d3.*Nor.n4e(:,3) + CFF.b3.*Nor.n4f(:,1) + CFF.c3.*Nor.n4f(:,2) +
CFF.d3.*Nor.n4f(:,3))./(VS.V *6);
D.D44 = ( CFF.b4.*Nor.n4d(:,1) + CFF.c4.*Nor.n4d(:,2) +
CFF.d4.*Nor.n4d(:,3) + CFF.b4.*Nor.n4e(:,1) +CFF.c4.*Nor.n4e(:,2)
+CFF.d4.*Nor.n4e(:,3) + CFF.b4.*Nor.n4f(:,1) + CFF.c4.*Nor.n4f(:,2) +
CFF.d4.*Nor.n4f(:,3))./(VS.V *6);
% %
for i =1 :ntl

    D.D11(i)=  D.D11(i)* -Prop.k(i) /Prop.cp(i);
    D.D12(i)=  D.D12(i)* -Prop.k(i) /Prop.cp(i);
    D.D13(i)=  D.D13(i)* -Prop.k(i) /Prop.cp(i);
    D.D14(i)=  D.D14(i)* -Prop.k(i) /Prop.cp(i);
    D.D21(i)=  D.D21(i)* -Prop.k(i) /Prop.cp(i);
    D.D22(i)=  D.D22(i)* -Prop.k(i) /Prop.cp(i);
    D.D23(i)=  D.D23(i)* -Prop.k(i) /Prop.cp(i);
    D.D24(i)=  D.D24(i)* -Prop.k(i) /Prop.cp(i);
    D.D31(i)=  D.D31(i)* -Prop.k(i) /Prop.cp(i);
    D.D32(i)=  D.D32(i)* -Prop.k(i) /Prop.cp(i);
    D.D33(i)=  D.D33(i)* -Prop.k(i) /Prop.cp(i);
    D.D34(i)=  D.D34(i)* -Prop.k(i) /Prop.cp(i);
    D.D41(i)=  D.D41(i)* -Prop.k(i) /Prop.cp(i);
    D.D42(i)=  D.D42(i)* -Prop.k(i) /Prop.cp(i);
    D.D43(i)=  D.D43(i)* -Prop.k(i) /Prop.cp(i);
    D.D44(i)=  D.D44(i)* -Prop.k(i) /Prop.cp(i);
end

TempoDs= toc;
tic
%CARREGAR PASSO DE VARIAVEIS NO TEMPO ti ESTIPULADO CASO ti SEJA
DIFERENTE
%DE ZERO
if ti~= 0
    Endereco =[ pwd ,'\', destino,'\',  + num2str(ti)];
    load( Endereco);

end

% INICIO MARCHA NO TEMPO PARA ti ATÉ tf
for ciclo=ti:tf/VS.dt
    [VS]=ATUALIZA_VARIAVEIS(VS);
    %MONTAR MATRIZ DAS CONTRIBUICOES DE CADA ELEMNTO
    tic
```

```
    A = spalloc(npl,npl,21*npl);
    for i=1:ntl % adiciona termo difusivo
        A(T1(i),T1(i))=D.D11(i)+A(T1(i),T1(i));
        A(T1(i),T2(i))=D.D12(i)+A(T1(i),T2(i));
        A(T1(i),T3(i))=D.D13(i)+A(T1(i),T3(i));
        A(T1(i),T4(i))=D.D14(i)+A(T1(i),T4(i));

        A(T2(i),T1(i))=D.D21(i)+A(T2(i),T1(i));
        A(T2(i),T2(i))=D.D22(i)+A(T2(i),T2(i));
        A(T2(i),T3(i))=D.D23(i)+A(T2(i),T3(i));
        A(T2(i),T4(i))=D.D24(i)+A(T2(i),T4(i));

        A(T3(i),T1(i))=D.D31(i)+A(T3(i),T1(i));
        A(T3(i),T2(i))=D.D32(i)+A(T3(i),T2(i));
        A(T3(i),T3(i))=D.D33(i)+A(T3(i),T3(i));
        A(T3(i),T4(i))=D.D34(i)+A(T3(i),T4(i));

        A(T4(i),T1(i))=D.D41(i)+A(T4(i),T1(i));
        A(T4(i),T2(i))=D.D42(i)+A(T4(i),T2(i));
        A(T4(i),T3(i))=D.D43(i)+A(T4(i),T3(i));
        A(T4(i),T4(i))=D.D44(i)+A(T4(i),T4(i));
    end

    TempoMontar = toc;
    B=zeros(npl,1);

    for i=1:npl % adiciona termo transiente
        rho= Prop.P_rho(i);
        A(i,i)=rho*V_no(i)/(VS.dt)+A(i,i);

        B(i)=rho*V_no(i)*VS.Temp(i,2)/(VS.dt)+B(i);
    end
    % ADICIONA CONDICOES DE CONTORNO
    %matriz G tras as condicoes de contorno
    NFaces=length(Contorno);

    for ii=1: NFaces
        for jj=1:length(Contorno(ii).Pontos)
            no = Contorno(ii).Pontos(jj);
            if G(ii,1) == 1 % Temperaturas prescrita no ponto
                B(no) = G(ii, 2);
                A(no,1:npl)=0;
                A(no,no)=1;
            end
            if G(ii,1) == 2 %Fluxo de calor prescrito no ponto
                B(no) = B(no)+ G(ii,2) *Contorno(ii).Area_Ponto(jj);
            end

            if G(ii,1)==3 %Fluxo de calor prescrito com coeficiente de
conveccao definido
                B(no) = B(no)+ G(ii,2) *Contorno(ii).Area_Ponto(jj) *
G(ii,3)/Prop.P_cp(no) ;% Prop.cp(no);
                A(no,no)=A(no, no) +
G(ii,2)*Contorno(ii).Area_Ponto(jj)/Prop.P_cp(no); %* VS.Temp(no,1);

            end
        end
    end
    % CALCULAR TEMPERATURA PARA NOVO TEMPO
```

```
    VS.Temp(:,1)=A\B;
    BalancoEnergia();
    % SALVA AS VARIAVEIS PARA O POS PROCESSAMENTO OU PARA INICIAR
CODIGO EM ti != 0
    Endereco =[ pwd ,'\', destino,'\',  + num2str(VS.tempo(end))];
    save( Endereco, 'X', 'Y','Z', 'VS','Prop', 'npl', 'AC',
'Contorno', 'E');


    fprintf('i: %1.0f tempo: %1.6f dt: %1.2f \n',ciclo, VS.tempo(end),
VS.dt);


end
TempoTotal= toc ;
%  FIM DO CODIGO
```

read_mesh3D_3ELE.m

```
% Este código foi desenvolvido como parte do Trabalho de Conclusão de
Curso
% para o curso de Engenharia de Energia da Universidade de
Brasília/Campus %
% Gama no segundo semestre do ano de 2016
% Entitulado :DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO
DA   %
% EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO
DE  %
% VOLUMES DE CONTROLE BASEADO EM ELEMENTOS
% por Pedro Paulo Dunice van Els
% sob Orientação do Prof Dr Fabio Alfaia da Cunha
% Este trabalha junto a outros códigos, que devem estar na mesma
pasta.    %
% Segue a lista:
% read_mesh3D_3ELE.m
% Areas_Contorno_No_AC.m
% Area_triangulo.m
% ATUALIZA_VARIAVEIS.m
% Normal_Triangulo.m
% EXE_Codigo_CVFEM_Transiente.m


function [P,T,E,SAIDA,Contorno,Material]=read_mesh3D_3ELE(namef)

fid=fopen(namef,'r');
esp=fscanf(fid,'%s',18);
SAIDA.NUMNP=fscanf(fid,'%8d',1);
SAIDA.NELEM=fscanf(fid,'%8d',1);
SAIDA.NGRPS=fscanf(fid,'%8d',1);
SAIDA.NBSETS=fscanf(fid,'%8d',1);
SAIDA.NDFCD=fscanf(fid,'%8d',1);
SAIDA.NDFVL=fscanf(fid,'%8d',1);
esp=fscanf(fid,'%s',4);
SAIDA.P=fscanf(fid,'%8d %f %f %f',[4 inf]);
SAIDA.P=SAIDA.P(2:4,:)';
```

```
esp=fscanf(fid,'%s',3);
SAIDA.T=fscanf(fid,'%8d %8d %8d %8d %8d %8d %8d',[7
inf]);SAIDA.T=SAIDA.T(4:7,:)';

for i=1:SAIDA.NGRPS
    esp=fscanf(fid,'%s',7);
    SAIDA.NMaterial(i)=fscanf(fid,'%8d',1);
    esp=fscanf(fid,'%s',4);
    SAIDA.Nome_material{i}=fscanf(fid,'%s',1);
    esp=fscanf(fid,'%s',1);
    SAIDA.Material{i}=fscanf(fid,'%8d',SAIDA.NMaterial(i));

Material(i)=struct('Nome',sprintf('%s',SAIDA.Nome_material{i}),'El_mat
erial',SAIDA.Material{i});
end

SAIDA.ORD=[2,1,3
          1,2,4
          2,3,4
          3,1,4];
P=SAIDA.P;
T=SAIDA.T;

 for i=1:SAIDA.NBSETS
    esp=fscanf(fid,'%s',4);
    Nome_contorno=fscanf(fid,'%s',1);
    esp=fscanf(fid,'%s',1);
    Ncontorno=fscanf(fid,'%8d',1);
    esp=fscanf(fid,'%s',2);
    Scontorno=fscanf(fid,'%8d %8d %8d',[3 Ncontorno]);
    for j=1:size(Scontorno,2)
    Econtorno(j,1:3)=T(Scontorno(1,j),SAIDA.ORD(Scontorno(3,j),:));
    end
    Contorno(i) =
struct('Nome',sprintf('%s',Nome_contorno),'N_CC',Econtorno);
    Econtorno=[];
%     Scontorno=[];
%     Nome_contorno=[];
%     Ncontorno=[];
 end

sgmnts=[T(:,[2,1,3]);T(:,[1,2,4]);T(:,[2,3,4]);T(:,[3,1,4])];
Nsgmnts=[1:length(T) 1:length(T) 1:length(T) 1:length(T)];
sgmntsSORT=sort(sgmnts,2);
[foo,ix,jx]=unique(sgmntsSORT,'rows');
vec=histc(jx,1:max(jx));
qx=(vec==1);
E=sgmnts(ix(qx),:);
SAIDA.nECC=Nsgmnts(ix(qx));
E(:,4)=Nsgmnts(ix(qx));

 fclose(fid);
```

Areas_Contorno_No_AC.m

```
% Este código foi desenvolvido como parte do Trabalho de Conclusão de
Curso%
```

```
% para o curso de Engenharia de Energia da Universidade de
Brasília/Campus %
% Gama no segundo semestre do ano de 2016
% Entitulado :DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO
DA   %
% EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO
DE
% VOLUMES DE CONTROLE BASEADO EM ELEMENTOS
% por Pedro Paulo Dunice van Els
% sob Orientação do Prof Dr Fabio Alfaia da Cunha
% Este trabalha junto a outros códigos, que devem estar na mesma
pasta.    %
% Segue a lista:
% read_mesh3D_3ELE.m
% Areas_Contorno_No_AC.m
% Area_triangulo.m
% ATUALIZA_VARIAVEIS.m
% Normal_Triangulo.m
% EXE_Codigo_CVFEM_Transiente.m
%

NFaces=length( Contorno);

for ii=1: NFaces
    Numero_elementos_contorno_ii=length(Contorno(ii).N_CC);
    Contorno(ii).Pontos= unique(Contorno(ii).N_CC);
    Contorno(ii).Area_Elemento= zeros(length(Contorno(ii).N_CC),1);
    Contorno(ii).Area_Ponto= zeros(length(Contorno(ii).Pontos),1);
    Contorno(ii).Normal_Elemento = zeros(length(Contorno(ii).N_CC),3);
    Contorno(ii).Meio_Elemento = zeros(length(Contorno(ii).N_CC),3);

    for jj =1:Numero_elementos_contorno_ii
        AC.P1=Contorno(ii).N_CC(jj,1);
        AC.P2=Contorno(ii).N_CC(jj,2);
        AC.P3=Contorno(ii).N_CC(jj,3);
        AC.P1X= X(AC.P1);
        AC.P1Y= Y(AC.P1);
        AC.P1Z= Z(AC.P1);
        AC.P2X= X(AC.P2);
        AC.P2Y= Y(AC.P2);
        AC.P2Z= Z(AC.P2);
        AC.P3X= X(AC.P3);
        AC.P3Y= Y(AC.P3);
        AC.P3Z= Z(AC.P3);
        AC.PositionP1= find( AC.P1==Contorno(ii).Pontos,1);
        AC.PositionP2= find( AC.P2==Contorno(ii).Pontos,1);
        AC.PositionP3= find( AC.P3==Contorno(ii).Pontos,1);
        Contorno(ii).Area_Elemento(jj)=
Area_triangulo(AC.P1X,AC.P1Y,AC.P1Z,AC.P2X,AC.P2Y,AC.P2Z,AC.P3X,AC.P3Y
,AC.P3Z);
        [N, P0]=
Normal_Triangulo(AC.P1X,AC.P1Y,AC.P1Z,AC.P2X,AC.P2Y,AC.P2Z,AC.P3X,AC.P
3Y,AC.P3Z);


        Contorno(ii).Normal_Elemento(jj,:) = N;
        Contorno(ii).Meio_Elemento(jj,:)=P0;
        Contorno(ii).Area_Ponto(AC.PositionP1) =
Contorno(ii).Area_Ponto(AC.PositionP1)
+Contorno(ii).Area_Elemento(jj)/3;
```

```
        Contorno(ii).Area_Ponto(AC.PositionP2) =
Contorno(ii).Area_Ponto(AC.PositionP2)
+Contorno(ii).Area_Elemento(jj)/3;
        Contorno(ii).Area_Ponto(AC.PositionP3) =
Contorno(ii).Area_Ponto(AC.PositionP3)
+Contorno(ii).Area_Elemento(jj)/3;
        figure(11)
        % % %         scale=0.05;
        % % % % %   N=N*scale;
        %         quiver3(P0(:), N(:));
        %         quiver3(P0(1),P0(2),P0(3), N(1),N(2),N(3));

        % %         hold on
        %         Contorno(ii).Area_Ponto(Contorno(ii).N_CC(jj,1)) =
Contorno(ii).Area_Ponto(Contorno(ii).N_CC(jj,1)) +
Contorno(ii).Area_Elemento(jj)/3;
        %         Contorno(ii).Area_Ponto(Contorno(ii).N_CC(jj,2)) =
Contorno(ii).Area_Ponto(Contorno(ii).N_CC(jj,2)) +
Contorno(ii).Area_Elemento(jj)/3;
        %         Contorno(ii).Area_Ponto(Contorno(ii).N_CC(jj,3)) =
Contorno(ii).Area_Ponto(Contorno(ii).N_CC(jj,3)) +
Contorno(ii).Area_Elemento(jj)/3;
        %
    end
end

AC.Soma = zeros(NFaces,1);

for ii =1 :NFaces
    for jj =1:length(Contorno(ii).N_CC)

        AC.Soma(ii)= Contorno(ii).Area_Elemento(jj) +AC.Soma(ii);
%Soma esta correta


    end
end
```

Normal_Triangulo.m

```
% Este código foi desenvolvido como parte do Trabalho de Conclusão de
Curso%
% para o curso de Engenharia de Energia da Universidade de
Brasília/Campus %
% Gama no segundo semestre do ano de 2016
% Entitulado :DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO
DA
%EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO
DE
% VOLUMES DE CONTROLE BASEADO EM ELEMENTOS
% por Pedro Paulo Dunice van Els
% sob Orientação do Prof Dr Fabio Alfaia da Cunha
% Este trabalha junto a outros códigos, que devem estar na mesma
pasta.    %
% Segue a lista:
% read_mesh3D_3ELE.m
```

```
% Areas_Contorno_No_AC.m
% Area_triangulo.m
% ATUALIZA_VARIAVEIS.m
% Normal_Triangulo.m
% EXE_Codigo_CVFEM_Transiente.m


function [returno, P0]= Normal_Triangulo(X1, Y1, Z1 ,X2, Y2, Z2,X3,
Y3, Z3)

Normal.P1=[X1 Y1 Z1];
Normal.P2=[X2 Y2 Z2];
Normal.P3=[X3 Y3 Z3];
Normal.P0=(Normal.P1 +Normal.P2+ Normal.P3)/3;
P0= Normal.P0;
Normal.P20= (Normal.P2-Normal.P0);
Normal.P30= (Normal.P3-Normal.P0);
normal= cross(Normal.P20, Normal.P30);
normal = normal / norm( normal ) ;
returno= normal;

%returno= Normal.D1*Normal.D2/6;
```

Area_triangulo.m

```
% Este código foi desenvolvido como parte do Trabalho de Conclusão de
Curso%
% para o curso de Engenharia de Energia da Universidade de
Brasília/Campus %
% Gama no segundo semestre do ano de 2016
%
% Entitulado :DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO
DA   %
% EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO
DE  %
% VOLUMES DE CONTROLE BASEADO EM ELEMENTOS
%
% por Pedro Paulo Dunice van Els
%
% sob Orientação do Prof Dr Fabio Alfaia da Cunha
%

% Este trabalha junto a outros códigos, que devem estar na mesma
pasta.    %
% Segue a lista:
%
% read_mesh3D_3ELE.m
%
% Areas_Contorno_No_AC.m
%
% Area_triangulo.m
%
% ATUALIZA_VARIAVEIS.m
%
```

```
% Normal_Triangulo.m
%
% EXE_Codigo_CVFEM_Transiente.m
%



function returno= Area_triangulo(AreapX1, AreapY1, AreapZ1 ,AreapX2,
AreapY2, AreapZ2,AreapX3, AreapY3, AreapZ3)

AreapP1=[AreapX1 AreapY1 AreapZ1];
AreapP2=[AreapX2 AreapY2 AreapZ2];
AreapP3=[AreapX3 AreapY3 AreapZ3];


AreapP21= (AreapP2-AreapP1);
AreapP31= (AreapP3-AreapP1);

returno= abs(norm( cross( AreapP21, AreapP31)))/2;
```

BalancoEnergia.m

```
% Este código foi desenvolvido como parte do Trabalho de Conclusão de
Curso
% para o curso de Engenharia de Energia da Universidade de
Brasília/Campus
% Gama no segundo semestre do ano de 2016
% Entitulado :DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO
DA
% EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO
DE
% VOLUMES DE CONTROLE BASEADO EM ELEMENTOS
% por Pedro Paulo Dunice van Els
% sob Orientação do Prof Dr Fabio Alfaia da Cunha
% Este trabalha junto a outros códigos, que devem estar na mesma
pasta.
% Segue a lista:
% read_mesh3D_3ELE.m
% Areas_Contorno_No_AC.m
% Area_triangulo.m
% ATUALIZA_VARIAVEIS.m
% Normal_Triangulo.m
% EXE_Codigo_CVFEM_Transiente.m


% Diferenca de Energia por Passo de tempo

for j =1: npl
    E.Interna0(j) = Prop.P_cp(j) * Prop.P_rho(j) * V_no(j) * (
VS.Temp(j,2)- Tinicial  );
    E.Interna1(j) = Prop.P_cp(j) * Prop.P_rho(j) * V_no(j) * (
VS.Temp(j,1)- Tinicial  );%% [J]
end
E.SInterna0 = sum(E.Interna0);
E.SInterna1 = sum(E.Interna1);
```

```
% Contabilizar Fluxo de calor pelo contorno
E.Q_Lost=0;
npc= length( Contorno(3).Pontos);
for i= 1: npc
    ponto = Contorno(3).Pontos(i);
    E.Q_Lost= E.Q_Lost+ Contorno(3).Area_Ponto(i) * G(3,2) *
(VS.Temp(ponto,1)-G(3,3)) ;

end
npc= length( Contorno(4).Pontos);
for i= 1: npc
    ponto = Contorno(4).Pontos(i);
    E.Q_Lost= E.Q_Lost+Contorno(4).Area_Ponto(i) * G(4,2) *
(VS.Temp(ponto,1)-G(4,3) );
end
E.Lost = E.Q_Lost * VS.dt;% [J]


E.DifC10= + E.SInterna1 - E.SInterna0;
E.Lost  =E.Lost + E.DifC10 ;
```

ATUALIZA_VARIAVEIS.m

```
% Este código foi desenvolvido como parte do Trabalho de Conclusão de
Curso
% para o curso de Engenharia de Energia da Universidade de
Brasília/Campus
% Gama no segundo semestre do ano de 2016
% Entitulado :DESENVOLVIMENTO DE UM CÓDIGO COMPUTACIONAL PARA SOLUÇÃO
DA
% EQUAÇÃO DE TRANSFERÊNCIA DE CALOR TRIDIMENSIONAL UTILIZANDO O MÉTODO
DE
% VOLUMES DE CONTROLE BASEADO EM ELEMENTOS
% por Pedro Paulo Dunice van Els
% sob Orientação do Prof Dr Fabio Alfaia da Cunha
% Este trabalha junto a outros códigos, que devem estar na mesma
pasta.
% Segue a lista:
% read_mesh3D_3ELE.m
% Areas_Contorno_No_AC.m
% Area_triangulo.m
% ATUALIZA_VARIAVEIS.m
% Normal_Triangulo.m
% EXE_Codigo_CVFEM_Transiente.m



function [VS]=ATUALIZA_VARIAVEIS(VS)
% VS.UT(:,2)=VS.UT(:,1);
% VS.VT(:,2)=VS.VT(:,1);
% VS.roF(:,2)=VS.roF(:,1);
VS.Temp(:,2) = VS.Temp(:,1);
VS.tempo(end+1)=VS.tempo(end)+VS.dt;

 %FIM DO CODIGO
```