



MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2019/05.21.21.02-TDI

SIMULAÇÃO NUMÉRICA DA ABLAÇÃO DO PTFE EM UM PROPULSOR DE PLASMA PULSADO USANDO OPENFOAM

Lucas Vinícius de Souza

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Combustão e Propulsão, orientada pelo Dr. Rodrigo Intini Marques, aprovada em 27 de maio de 2019.

URL do documento original:

<<http://urlib.net/8JMKD3MGP3W34R/3TBHPRE>>

INPE
São José dos Campos
2019

PUBLICADO POR:

Instituto Nacional de Pesquisas Espaciais - INPE
Gabinete do Diretor (GBDIR)
Serviço de Informação e Documentação (SESID)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

CONSELHO DE EDITORAÇÃO E PRESERVAÇÃO DA PRODUÇÃO INTELLECTUAL DO INPE - CEPPII (PORTARIA Nº 176/2018/SEI-INPE):**Presidente:**

Dra. Marley Cavalcante de Lima Moscati - Centro de Previsão de Tempo e Estudos Climáticos (CGCPT)

Membros:

Dra. Carina Barros Mello - Coordenação de Laboratórios Associados (COCTE)

Dr. Alisson Dal Lago - Coordenação-Geral de Ciências Espaciais e Atmosféricas (CGCEA)

Dr. Evandro Albiach Branco - Centro de Ciência do Sistema Terrestre (COCST)

Dr. Evandro Marconi Rocco - Coordenação-Geral de Engenharia e Tecnologia Espacial (CGETE)

Dr. Hermann Johann Heinrich Kux - Coordenação-Geral de Observação da Terra (CGOBT)

Dra. Ieda Del Arco Sanches - Conselho de Pós-Graduação - (CPG)

Silvia Castro Marcelino - Serviço de Informação e Documentação (SESID)

BIBLIOTECA DIGITAL:

Dr. Gerald Jean Francis Banon

Clayton Martins Pereira - Serviço de Informação e Documentação (SESID)

REVISÃO E NORMALIZAÇÃO DOCUMENTÁRIA:

Simone Angélica Del Ducca Barbedo - Serviço de Informação e Documentação (SESID)

André Luis Dias Fernandes - Serviço de Informação e Documentação (SESID)

EDITORAÇÃO ELETRÔNICA:

Ivone Martins - Serviço de Informação e Documentação (SESID)

Cauê Silva Fróes - Serviço de Informação e Documentação (SESID)



MINISTÉRIO DA CIÊNCIA, TECNOLOGIA, INOVAÇÕES E COMUNICAÇÕES
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

sid.inpe.br/mtc-m21c/2019/05.21.21.02-TDI

SIMULAÇÃO NUMÉRICA DA ABLAÇÃO DO PTFE EM UM PROPULSOR DE PLASMA PULSADO USANDO OPENFOAM

Lucas Vinícius de Souza

Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espaciais/Combustão e Propulsão, orientada pelo Dr. Rodrigo Intini Marques, aprovada em 27 de maio de 2019.

URL do documento original:

<<http://urlib.net/8JMKD3MGP3W34R/3TBHPRE>>

INPE
São José dos Campos
2019

Dados Internacionais de Catalogação na Publicação (CIP)

Souza, Lucas Vinícius de.
So89s Simulação numérica da ablação do PTFE em um propulsor de plasma pulsado usando OpenFOAM / Lucas Vinícius de Souza. – São José dos Campos : INPE, 2019.
xxii + 152 p. ; (sid.inpe.br/mtc-m21c/2019/05.21.21.02-TDI)

Dissertação (Mestrado em Engenharia e Tecnologia Espaciais/Combustão e Propulsão) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2019.
Orientador : Dr. Rodrigo Intini Marques.

1. PPT. 2. CFD. 3. OpenFOAM. 4. C++. I.Título.

CDU 629.7.036.74



Esta obra foi licenciada sob uma Licença [Creative Commons Atribuição-NãoComercial 3.0 Não Adaptada](https://creativecommons.org/licenses/by-nc/3.0/).

This work is licensed under a [Creative Commons Attribution-NonCommercial 3.0 Unported License](https://creativecommons.org/licenses/by-nc/3.0/).


Aluno (a): *Lucas Vinícius de Souza*

Título: "SIMULAÇÃO NUMÉRICA DA ABLAÇÃO DO PTFE EM PROPULSOR DE PLASMA PULSADO USANDO OPENFOAM"

Aprovado (a) pela Banca Examinadora em cumprimento ao requisito exigido para obtenção do Título de **Mestre** em

**Engenharia e Tecnologia
Espaciais/Combustão e Propulsão**

Dr. Rodrigo Intini Marques

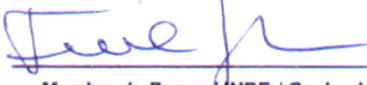


Presidente / Orientador(a) / INPE / Cachoeira Paulista - SP

Participação por Video - Conferência

Aprovado Reprovado

Dr. Fernando de Souza Costa

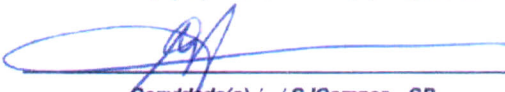


Membro da Banca / INPE / Cachoeira Paulista - SP

Participação por Video - Conferência

Aprovado Reprovado

Dr. Gilberto Marrega Sandonato



Convidado(a) / . / SJC Campos - SP

Participação por Video - Conferência

Aprovado Reprovado

Este trabalho foi aprovado por:

maioria simples

unanimidade

AGRADECIMENTOS

À CAPES, agradeço pelo apoio financeiro concedido durante parte do meu mestrado. E por fim, agradeço a todos os demais que de alguma forma contribuíram para o desenvolvimento deste trabalho.

RESUMO

O propulsor de plasma pulsado é um motor para veículos espaciais que gera empuxo a partir da ejeção de plasma produzido por descargas elétricas pulsadas de alta tensão sobre a superfície de um material dielétrico e ablativo. Foi desenvolvido e utilizado já no começo dos programas espaciais, na década de 1960. Sua análise e aperfeiçoamento desde então foi baseada, principalmente, em medições experimentais e leis empíricas derivadas de tais medições. Entre os principais problemas do PPT pode-se apontar o baixo empuxo e a baixa eficiência ($\sim 10\%$). A complexidade de fenômenos ocorrendo em um PPT não é capturada por leis empíricas e modelos de circuitos elétricos, havendo necessidade de ferramentas de análise mais completas. Para sanar este problema, no presente trabalho foi desenvolvido um código numérico que permita a análise numérica do escoamento de plasma no PPT, bem como a análise da distribuição de temperatura dentro do propelente sólido. O programa foi baseado na biblioteca C++ de código aberto OpenFOAM, desenvolvida, principalmente, para simulações de dinâmica de fluidos computacional, mas tendo também capacidades multifísicas. A validação do código numérico foi feita por meio da comparação dos resultados numéricos obtidos com dados experimentais já disponíveis na literatura sobre PPTs.

Palavras-chave: PPT. CFD. OpenFOAM. C++.

PULSED PLASMA THRUSTER PTFE NUMERICAL SIMULATION USING OPENFOAM

ABSTRACT

The pulsed plasma propellant is a space vehicle engine that generates thrust from the ejection of plasma produced by pulsed high voltage electrical discharges on the surface of a dielectric and ablative material. It was developed and used early in space programs in the 1960s. Its analysis and refinement has since been based primarily on experimental measurements and empirical laws derived from such measurements. Among the main problems of the PPT is the low thrust and the low efficiency (~10%). The complexity of phenomena occurring in a PPT is not captured by empirical laws and electric circuit models, and there is a need for more complete analysis tools. To solve this problem, in the present work a numerical code was developed that allows the numerical analysis of the plasma flow in the PPT, as well as the analysis of the temperature distribution inside the solid propellant. The program was based on the OpenFOAM open source C++ library, developed mainly for computational fluid dynamics simulations, but also having multiphysical capabilities. Numerical code validation was done by comparing the numerical results obtained with experimental data already available in the literature on PPTs.

Palavras-chave: PPT. CFD. OpenFOAM. C++.

LISTA DE FIGURAS

	<u>Pág.</u>
1.1 Diagrama esquemático de um PPT.	5
1.2 Força de Lorentz no PPT.	6
1.3 Ciclo do PPT com a formação de LTA.	8
1.4 Geometria retangular.	9
1.5 Geometria retangular sem mola.	10
1.6 Geometria coaxial.	11
1.7 Geometria coaxial com propelente cônico e vela de ignição.	11
2.1 Representação esquemática da camada cinética.	16
2.2 Representação esquemática da camada cinética e hidrodinâmica.	19
2.3 Resumo do processo de discretização.	28
2.4 Discretização de Volume Finito.	30
2.5 Estrutura da biblioteca OpenFOAM.	33
2.6 Estrutura de diretório de um caso OpenFOAM.	35
2.7 Estrutura de um código OpenFOAM.	36
2.8 Equação no código C++ da biblioteca OpenmFOAM.	38
2.9 Passos para o desenvolvimento de um código.	43
3.1 Circuito simulado pelo código <i>RLC_Foam</i>	46
3.2 Detalhe da malha usada para simulação do aquecimento no PTFE.	49
3.3 Malha para simulação do aquecimento na barra de PTFE vista em 3D.	50
3.4 Fluxograma de execução do código <i>rhoCentralFoam</i>	55
3.5 Malha para simulação do escoamento, vista no plano YX.	57
3.6 Malha para simulação do escoamento vista em 3D.	58
3.7 Fluxograma de execução do código <i>PPT_Foam</i>	68
4.1 Cavidade do propulsor e distribuição de densidade, pressão, temperatura e velocidade ao longo do eixo central, nos instantes de 0 e $20\mu s$	70
4.2 Cavidade do propulsor e distribuição de densidade, pressão, temperatura e velocidade ao longo do eixo central, nos instantes de 40 e $60\mu s$	71
4.3 Cavidade do propulsor e distribuição de densidade, pressão, temperatura e velocidade ao longo do eixo central, nos instantes de 80 e $100\mu s$	72
4.4 Barra de PTFE e distribuição de temperatura ao longo do eixo central, em diferentes instantes de tempo.	73
4.5 Barra de PTFE e distribuição de temperatura ao longo do eixo central, em diferentes instantes de tempo.	74

4.6	Formas de onda de corrente e carga sobrepostas.	75
4.7	Pontos de sondagem.	76
4.8	Temperaturas nos pontos 0, 1 e 2.	77
4.9	Densidades nos pontos 0, 1 e 2.	78
4.10	Detalhe ampliado da Figura 4.9.	78
4.11	Velocidade no ponto 1 (V1) e velocidade do som no meio (Vs).	79
4.12	Energia do fluido, Número de Mach e constante adimensional α	80
4.13	Resíduos das variáveis de simulação.	81
4.14	Gráficos das principais propriedades plotados em conjunto para melhor comparação.	82
5.1	Corrente calculada pelo MACH2.	84
5.2	Formas de onda de corrente e carga sobrepostas.	84
5.3	Perfil de temperatura dentro do PTFE, calculado pelo código MACH2.	86
5.4	Perfil de temperatura dentro do PTFE, calculado pelo código <i>rhoCentralFoam</i>	86
5.5	Comparação entre a massa ablacionada experimental e predita pelo código MACH2.	88
5.6	Massa ablacionada em função do tempo predita pelo código MACH2.	89
5.7	Comparação entre o <i>impulse-bit</i> experimental e predito pelo código MACH2.	89
5.8	<i>Impulse-bit</i> em função do tempo predito pelo código MACH2.	90
5.9	Gráficos das principais propriedades.	91

LISTA DE TABELAS

	<u>Pág.</u>
3.1 Constantes físicas usadas no código <i>RLC_Foam</i>	47
3.2 Geometria da malha para o código <i>HeatPropagation</i>	48
3.3 Condições de contorno para a Temperatura T (K) na malha do PTFE.	51
3.4 Condição inicial para a temperatura na malha do PTFE.	51
3.5 Propriedades do PTFE sólido.	52
3.6 Geometria da malha para o código <i>rhoCentralFoam</i>	56
3.7 Condições de contorno para velocidade U (m/s).	59
3.8 Condições de contorno para pressão p (Pa).	60
3.9 Condições de contorno para temperatura T (K).	60
3.10 Condições iniciais para velocidade, pressão e temperatura.	61
3.11 Tabela com os dados do campo <i>thermoType</i>	62
3.12 Tabela com os dados do campo <i>mixture</i>	63
5.1 Duração e amplitude das correntes obtidas com MACH2 e <i>rhoCentralFoam</i>	87
5.2 Tabela com dados experimentais.	87
5.3 Tabela com dados simulados.	87

LISTA DE ABREVIATURAS E SIGLAS

PPT	–	Propulsor de Plasma Pulsado
CFD	–	Dinâmica de Fluido Computacional
PTFE	–	Politetrafluoretileno
PPU	–	Unidade de Processamento de Potência
RLC	–	Resistência Indutor Capacitor
LTA	–	Ablação Tardia
DNS	–	Simulação Numérica Direta
LES	–	Simulação de Grandes Vórtices
DSMC	–	Simulação Numérica de Monte Carlo

LISTA DE SÍMBOLOS

m_p	–	massa de propelente ejetado
\dot{m}_p	–	fluxo de propelente ejetado
c	–	velocidade do propelente ejetado em relação ao veículo
I_{sp}	–	impulso específico
g_0	–	aceleração média da gravidade terrestre no nível do mar
J	–	densidade de corrente elétrica
B	–	campo magnético
f	–	frequência de disparos
T	–	empuxo aplicado ao satélite
I_{bit}	–	impulse bit
F	–	empuxo médio
f	–	função de distribuição de velocidades dentro da camada cinética
f_1	–	função de distribuição de velocidades no ponto 1
f_2	–	função de distribuição de velocidades no ponto 2
T_0	–	temperatura no ponto 0
T_1	–	temperatura no ponto 1
T_2	–	temperatura no ponto 2
n_0	–	densidade numérica no ponto 0
n_1	–	densidade numérica no ponto 1
n_2	–	densidade numérica no ponto 2
n_{crit}	–	densidade numérica crítica
ρ_0	–	densidade no ponto 0
ρ_1	–	densidade no ponto 1
ρ_2	–	densidade no ponto 2
P_1	–	pressão no ponto 1
P_2	–	pressão no ponto 2
V_1	–	velocidade no ponto 1
k	–	constante de Boltzmann
m	–	massa atômica
β	–	coeficiente de proporcionalidade
P_0	–	pressão de equilíbrio
P_c	–	pressão característica
T_c	–	temperatura característica
γ	–	razão dos calores específicos
U_{idf}	–	energia associada com os graus internos de liberdade
ρ	–	densidade de massa
\mathbf{u}	–	velocidade do fluido
p	–	pressão
e	–	energia interna específica
E	–	densidade de energia total

\mathbf{j}	–	fluxo de calor difusivo
\mathbf{T}	–	tensor de estresse viscoso
μ	–	viscosidade dinâmica
\mathbf{D}	–	tensor de deformação do gradiente
\mathbf{I}	–	tensor unitário
c_p	–	calor específico a pressão constante
c_v	–	calor específico a volume constante
R	–	constante do gás
\mathbf{S}_f	–	vetor de área da face
\mathbf{d}	–	distância entre centroides das células
\mathbf{d}_{fN}	–	distância entre face e centroide da célula
Ψ	–	propriedade qualquer sendo transportada
ϕ_f	–	fluxo volumétrico
c_f	–	velocidade do som na face

SUMÁRIO

	<u>Pág.</u>
1 INTRODUÇÃO	1
1.1 Propósito e Objetivo	1
1.2 Propulsão Espacial	1
1.3 Propulsão Elétrica	1
1.3.1 Propulsor de Plasma Pulsado - PPT(Pulsed Plasma Thruster)	3
1.3.2 Princípio de funcionamento	4
1.3.3 Parâmetro de desempenho	7
1.3.4 Propelente	8
1.3.5 Geometrias	9
1.3.6 Vantagens e Desvantagens	11
2 REVISÃO BIBLIOGRÁFICA	15
2.1 Ablação	15
2.1.1 Camada cinética	15
2.1.2 Camada hidrodinâmica	18
2.1.3 Correções e conjunto de equações modificadas	20
2.1.3.1 Combinação das camadas cinética e hidrodinâmica	22
2.2 Simulação do PPT	23
2.2.1 MACH2	24
2.3 Equações Governantes da Mecânica do Fluidos	25
2.4 O Processo de Discretização	27
2.4.1 Método dos volumes finitos	29
2.4.1.1 Termos convectivos	30
2.4.1.2 Termos gradientes	31
2.4.1.3 Termos laplacianos	31
2.4.1.4 Condições de contorno	32
2.5 OpenFOAM - Open Field Operation and Manipulation	33
2.5.1 Estrutura de diretório da simulação	34
2.5.2 Estrutura de diretórios do código de simulação	36
2.5.3 Vantagens	37
2.5.4 <i>RhoCentralFoam</i>	39
2.5.4.1 Tratamento dos termos convectivos	39

2.5.4.2	Tratamento dos termos gradientes	41
2.5.4.3	Tratamento dos termos laplacianos	42
2.6	Desenvolvimento do Simulador	42
3	METODOLOGIA	45
3.1	Código <i>RLC_Foam</i>	46
3.1.1	Malha e Condições de contorno	46
3.1.2	Condições iniciais	47
3.1.3	Propriedades físicas	47
3.2	Código <i>HeatPropagation</i>	48
3.2.1	Malha	48
3.2.2	Condições de contorno	51
3.2.3	Condição inicial	51
3.2.4	Propriedades Físicas	52
3.3	Código <i>rhoCentralFoam</i>	53
3.3.1	Malha	56
3.3.2	Condições de contorno	59
3.3.3	Condições iniciais	61
3.3.4	Propriedades físicas	62
3.4	Condição de Contorno de Ablação	65
3.5	Código <i>PPT_Foam</i>	67
4	RESULTADOS	69
4.1	Escoamento na Cavidade do PPT	69
4.2	Propagação de Calor no PTFE	73
4.3	Circuito RLC	75
4.4	Propriedades da Camada de Ablação	76
5	ANÁLISE DOS RESULTADOS	83
5.1	Corrente do Circuito RLC	83
5.2	Distribuição de Temperatura na Barra de PTFE	85
5.3	Massa Ablacionada e <i>Impulse-bit</i>	87
6	CONCLUSÃO	93
	REFERÊNCIAS BIBLIOGRÁFICAS	95
	APÊNDICE A - DICIONÁRIOS DE CONFIGURAÇÃO	101
A.1	Diretório <i>constant</i>	101

A.1.1	<i>thermophysicalProperties</i>	101
A.1.2	<i>transportProperties</i>	102
A.1.3	<i>turbulenceProperties</i>	103
A.2	Diretório <i>system</i>	104
A.2.1	<i>blockMeshDict</i>	104
APÊNDICE B - CONDIÇÃO DE CONTORNO DE ABLAÇÃO		109
B.1	Condição de Contorno de Ablação	109
B.1.1	<i>AblationFvPatchField.C</i>	109
B.1.2	<i>AblationFvPatchField.H</i>	115
B.1.3	<i>AblationFvPatchFields.C</i>	121
B.1.4	<i>AblationFvPatchFields.H</i>	122
B.1.5	<i>AblationFvPatchFieldsFwd.H</i>	124
APÊNDICE C - PPT_FOAM		127
C.1	Código Base do PPT_Foam	127
C.1.1	<i>directionInterpolate.H</i>	127
C.1.2	<i>createFluidMesh.H</i>	128
C.1.3	<i>createSolidMesh.H</i>	128
C.1.4	<i>createFields.H</i>	128
C.1.5	<i>createSolidFields.H</i>	132
C.1.6	<i>createFieldRefs.H</i>	136
C.1.7	<i>readFluxScheme.H</i>	136
C.1.8	<i>CreateAblationFields.H</i>	136
C.1.9	<i>solveCurrent.H</i>	139
C.1.10	<i>solveHeatPropagation.H</i>	139
C.1.11	<i>UpdateAblationFields.H</i>	140
C.1.12	<i>centralCourantNo.H</i>	142
C.1.13	<i>setRDeltaT.H</i>	144
C.1.14	<i>PPT_Foam.C</i>	144

1 INTRODUÇÃO

1.1 Propósito e Objetivo

O objetivo deste trabalho é o desenvolvimento de um programa, baseado em código aberto, para o modelamento dos principais processos físicos que ocorrem no Propulsor de Plasma Pulsado (PPT). Os códigos atuais de simulação são comerciais e/ou de código fechado, obrigando a realização de testes exaustivos para determinação dos parâmetros ideais de projeto. Esta ferramenta de simulação pode ser útil na análise e projeto de futuros propulsores, facilitando o teste de conceitos e diminuindo o esforço experimental.

1.2 Propulsão Espacial

A propulsão espacial por foguete é aquela capaz de alterar o momentum linear e/ou angular de um corpo, através da ejeção de uma massa propelente (fluido de trabalho) armazenada no corpo. Pode ser dividida em três grandes grupos: Propulsão Química, Propulsão Nuclear e Propulsão Elétrica (SUTTON; BIBLARZ, 2001).

Na propulsão química, o propelente é ejetado utilizando-se a energia liberada do próprio propelente, através de uma reação química de combustão ou decomposição catalítica. Ou seja, a energia que ejeta o fluido de trabalho está contida no próprio fluido. Normalmente, utiliza-se uma tubeira com seção convergente-divergente para aceleração dos gases aquecidos resultantes da reação química (SUTTON; BIBLARZ, 2001).

A propulsão nuclear utiliza uma fonte de radioisótopos para aquecer o fluido de trabalho, ou seja, a energia responsável por acelerar o propelente está separada do mesmo. O sistema de aceleração, através de uma tubeira de seção convergente-divergente, é o mesmo da propulsão química (SUTTON; BIBLARZ, 2001).

1.3 Propulsão Elétrica

A propulsão elétrica utiliza eletricidade para acelerar o fluido de trabalho e gerar empuxo. Tem sido considerada para aplicações espaciais desde o começo dos programas espaciais, com a primeira missão espacial de um propulsor elétrico sendo feita já na década de 1960 (BURTON; TURCHI, 1998). Apesar disso, a propulsão elétrica começou a popularizar-se apenas no meio da década de 1990, com o desenvolvimento de tecnologias que disponibilizaram uma potência elétrica suficientemente grande nos satélites (JAHN; LYMAN, 1969).

Dependendo do mecanismo utilizado para conversão da energia elétrica em energia cinética, divide-se a propulsão elétrica em três grandes categorias: eletrotérmica, eletrostática e eletromagnética. Na propulsão eletrotérmica, a eletricidade é usada apenas para o aquecimento do propelente, permitindo sua posterior expansão em uma tubeira. Na propulsão eletrostática, um campo elétrico é usado para acelerar um propelente carregado eletricamente (ionizado). Na propulsão eletromagnética, as correntes elétricas presentes no fluido de trabalho, geralmente um plasma, interagem com um campo magnético, causando a aceleração do fluido (MARIN, 2014)(JAHN; LYMAN, 1969).

Entre os principais usos da propulsão elétrica destacam-se: apontamento preciso de instrumentos, controle de atitude (GATSONIS et al., 2016), descarregamento de rodas de reação, transferência de órbita, cancelamento de arrasto, viagem interplanetária.

Para avaliação do desempenho de diferentes tipos de propulsores, alguns parâmetros são usados, estes parâmetros permitem uma comparação de diferentes sistemas propulsivos (SUTTON; BIBLARZ, 2001). Na propulsão elétrica utiliza-se, entre outros, os parâmetros de empuxo e impulso específico (GOEBEL; KATZ, 2008).

O empuxo é definido como a força exercida sobre o veículo espacial devido à exaustão de propelente. É calculado conforme a Equação 1.1(SUTTON; BIBLARZ, 2001):

$$F = m \frac{d\mathbf{v}}{dt} = m\mathbf{a} = -\frac{dm}{dt}\mathbf{v}_e = \frac{dm}{dt}\mathbf{v}_{rel} \quad (1.1)$$

Onde $-\mathbf{v}_e = \mathbf{v}_{rel}$ é a velocidade relativa do propelente em relação ao foguete. A Equação 1.1 é chamada equação do empuxo e sua dedução simplificada é apresentada abaixo:

$$\mathbf{P}(t) = (m + \delta m)\mathbf{v}$$

$$\mathbf{P}(t + \Delta t) = m(\mathbf{v} + \Delta\mathbf{v}) + \delta m(\mathbf{v} - \mathbf{v}_e)$$

$$0 = \Delta\mathbf{P} = \mathbf{P}(t + \Delta t) - \mathbf{P}(t) = m\Delta\mathbf{v} - \delta m\mathbf{v}_e$$

Como:

$$m(t + \Delta t) - m(t) = \Delta m = -\delta m$$

Tem-se:

$$\Delta \mathbf{v} = -\frac{\Delta m}{m} \mathbf{v}_e$$

Dividindo pela variação do tempo Δt :

$$\frac{\Delta \mathbf{v}}{\Delta t} = -\frac{\Delta m}{\Delta t} \mathbf{v}_e$$

Passando ao limite em que $\Delta t \rightarrow 0$:

$$F = m \frac{d\mathbf{v}}{dt} = m\mathbf{a} = -\frac{dm}{dt} \mathbf{v}_e = \frac{dm}{dt} \mathbf{v}_{rel}$$

O impulso específico (I_{sp}) é a razão entre a variação de momento causada por uma força aplicada durante um intervalo, também conhecida como impulso, e o peso do propelente, calculado ao nível do mar, ejetado para causar esta variação de momento. A fórmula geral considera que o fluxo de massa e o empuxo podem variar com o tempo e é dada pela Equação 1.2 (SUTTON; BIBLARZ, 2001):

$$I_{sp} = \frac{\int_{t_1}^{t_2} F dt}{g_0 \int_{t_1}^{t_2} \dot{m}_p dt} \quad (1.2)$$

Onde g_0 é a aceleração média da gravidade terrestre no nível do mar, ou seja, $g_0 = 9,807 m/s^2$. O impulso específico é medido em segundos (s), apesar de não representar tempo.

1.3.1 Propulsor de Plasma Pulsado - PPT(Pulsed Plasma Thruster)

Propulsores de Plasma Pulsado (PPT) são propulsores categorizados como eletromagnéticos, embora uma parte do empuxo produzido se deva a efeitos gás dinâmicos (MARQUES, 2009).

Em 30 de novembro de 1964, a sonda Zond 2 foi lançada com destino a Marte da base de Baikonur, na antiga União Soviética. Esta sonda foi o primeiro artefato a utilizar um sistema de propulsão elétrica, com o uso de um PPT para controle de atitude. Usos de PPTs em satélites ocorreram nos Estados Unidos nos anos de 1968, com o satélite LES-6; 1974, com o satélite SMS e em 1981, com o trio de satélites TIP/NOVA [9]. Todos estes satélites acumularam um total de 28 anos de uso bem-sucedido de PPTs (BURTON; TURCHI, 1998).

Dentre os vários tipos de propulsores elétricos existentes, o PPT destaca-se por sua simplicidade, robustez, baixo custo de fabricação e curto tempo de desenvolvimento. Essas características permitiram a utilização de PPTs já na década de 1960, apesar de sua eficiência muito baixa. Outros propulsores elétricos, como os motores de íons de xenônio, arco-jatos e motores de efeito Hall tem uma eficiência maior que o PPT. Contudo, para alcançarem níveis ótimos de eficiência, tiveram que passar por décadas de desenvolvimento em laboratório, devido a sua maior complexidade. Portanto, com o PPT, temos uma combinação única de décadas de voos de qualificação no espaço e uma grande oportunidade de aumento de eficiência destes propulsores (BURTON; TURCHI, 1998).

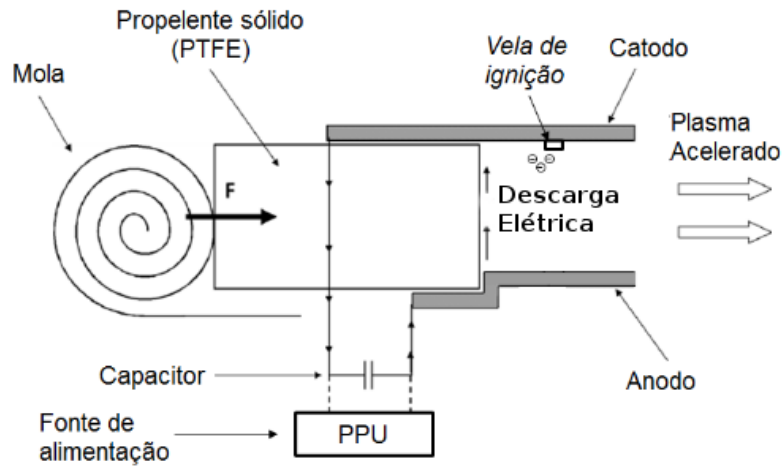
1.3.2 Princípio de funcionamento

Na Figura 1.1 vemos um diagrama esquemático de um PPT retangular padrão. Nessa figura são apresentados os componentes principais de um PPT, sendo esses componentes comuns a outros arranjos construtivos de PPT. Destacam-se na Figura 1.1 o propelente sólido em barra, a mola que abastece a câmara de descarga continuamente empurrando o propelente para a posição adequada, a fonte de alimentação do veículo espacial (*PPU - Power Processing Unit*), um capacitor (ou banco de capacitores) e a vela de ignição de descarga (MARQUES, 2009)(MARIN, 2014).

Nesta configuração específica de PPT, a barra de propelente sólido é posicionada entre dois eletrodos, sendo um deles o anodo (polo positivo da descarga) e o outro catodo (polo negativo). A mola traseira do propelente serve para empurrar o mesmo para a posição adequada conforme é consumido (BURTON; TURCHI, 1998).

A PPU do veículo espacial carrega o capacitor (ou banco de capacitores) de descarga a partir do barramento principal de energia. Além de carregar o sistema de descarga principal, a PPU também fornece energia para a vela de ignição (MARQUES, 2009).

Figura 1.1 - Diagrama esquemático de um PPT.



Fonte: Marin (2014) Adaptado de Marques (2009).

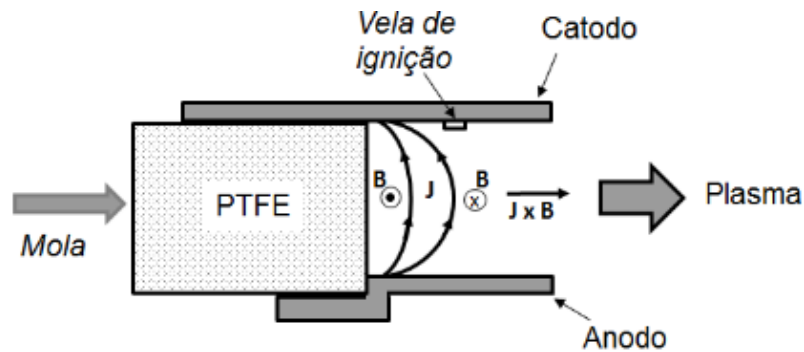
A ignição do PPT é um processo bastante complexo. A vela de ignição, geralmente feita de material semicondutor ou isolante, gera uma pequena descarga de elétrica próxima a superfície do propelente, aquecendo-o. O aquecimento causa a evaporação de uma pequena quantidade de propelente, suficiente para diminuir a resistência elétrica entre os eletrodos principais e manter a alta corrente da descarga principal. A energia e massa associadas com a descarga de ignição são muito pequenas quando comparadas à descarga principal, logo, seus valores não devem afetar o desempenho do PPT diretamente (BURTON; TURCHI, 1998).

Uma vez iniciada a descarga principal ($\sim\text{kA}, \sim\mu\text{s}$) na superfície do propelente, uma fração do mesmo sofre ablação. Este processo gera gases não ionizados e uma pequena quantidade de plasma, além de aquecer a barra de propelente. O empuxo é gerado através da combinação de dois efeitos: a aceleração e ejeção a altas velocidades do plasma, por meio da força de corpo eletromagnética; e a expansão térmica do vapor de propelente não-ionizado, devido ao gradiente de pressão (CASSADY et al., 2000).

O primeiro destes efeitos, a força eletromagnética, se deve a interação da densidade de corrente elétrica (J), que flui através do plasma, e o campo magnético (B), gerado por essa mesma corrente elétrica. O produto vetorial entre a densidade de corrente e campo magnético auto induzido gera uma força eletromagnética denominada Força de Lorentz ($J \times B$) (CASSADY et al., 2000). Esta força de corpo atua em toda a massa de plasma, acelerando-a e deformando-a conforme a mesma se desloca até a saída

do propulsor (JAHN; LYMAN, 1969). A Figura 1.2 ilustra este processo.

Figura 1.2 - Força de Lorentz no PPT.



Fonte: Marin (2014) Adaptado de Marques (2009).

O segundo efeito que contribui para o empuxo do PPT é a expansão dos gases não-ionizados resultantes da evaporação do propelente. Esses gases não sofrem a ação de nenhum tipo de força eletromagnética, sendo acelerados apenas pelo gradiente de pressão que existe entre o vapor de propelente e o vácuo no qual o propulsor opera (MARQUES, 2009).

Desta maneira existem, basicamente, dois tipos de massa no PPT: a massa de plasma, que é acelerada eletromagneticamente a altas velocidades, e a massa não-ionizada, que é acelerada, através de um gradiente de pressão termodinâmica, a baixas velocidades. Essa massa acelerada a baixas velocidades diminui consideravelmente o impulso específico do PPT, sendo a principal causa para a baixa eficiência do mesmo (MARQUES, 2009).

Essa massa ejetada a baixas velocidades é gerada por um fenômeno denominado Ablação Tardia (*LTA - Late Time Ablation*). A fonte do fenômeno é a evaporação contínua do propelente aquecido pela descarga elétrica principal do PPT. Esta evaporação acontece durante um intervalo de tempo muito maior do que o intervalo da descarga elétrica, ou seja, não há mais a presença de correntes ou campos elétricos/magnéticos capazes de acelerar esta massa. Soluções para este problema usando propulsores de descarga dupla já foram propostas (MARQUES, 2009).

Além da perda de massa por evaporação, também se perde massa por ejeção de

macro partículas do propelente, eletrodos e da vela de ignição. Assim sendo, o PPT perde massa por diferentes mecanismos, a diferentes taxas e em escalas de tempo diferentes. A Figura 1.3 ilustra um ciclo de funcionamento do PPT com o fenômeno da LTA (MARQUES, 2009).

1.3.3 Parâmetro de desempenho

Devido ao modo de operação pulsado, se faz necessária a definição de um parâmetro mais adequado para medida de desempenho de um propulsor do tipo PPT. Este parâmetro é chamado de *impulse bit* e é definido como (MARQUES, 2009):

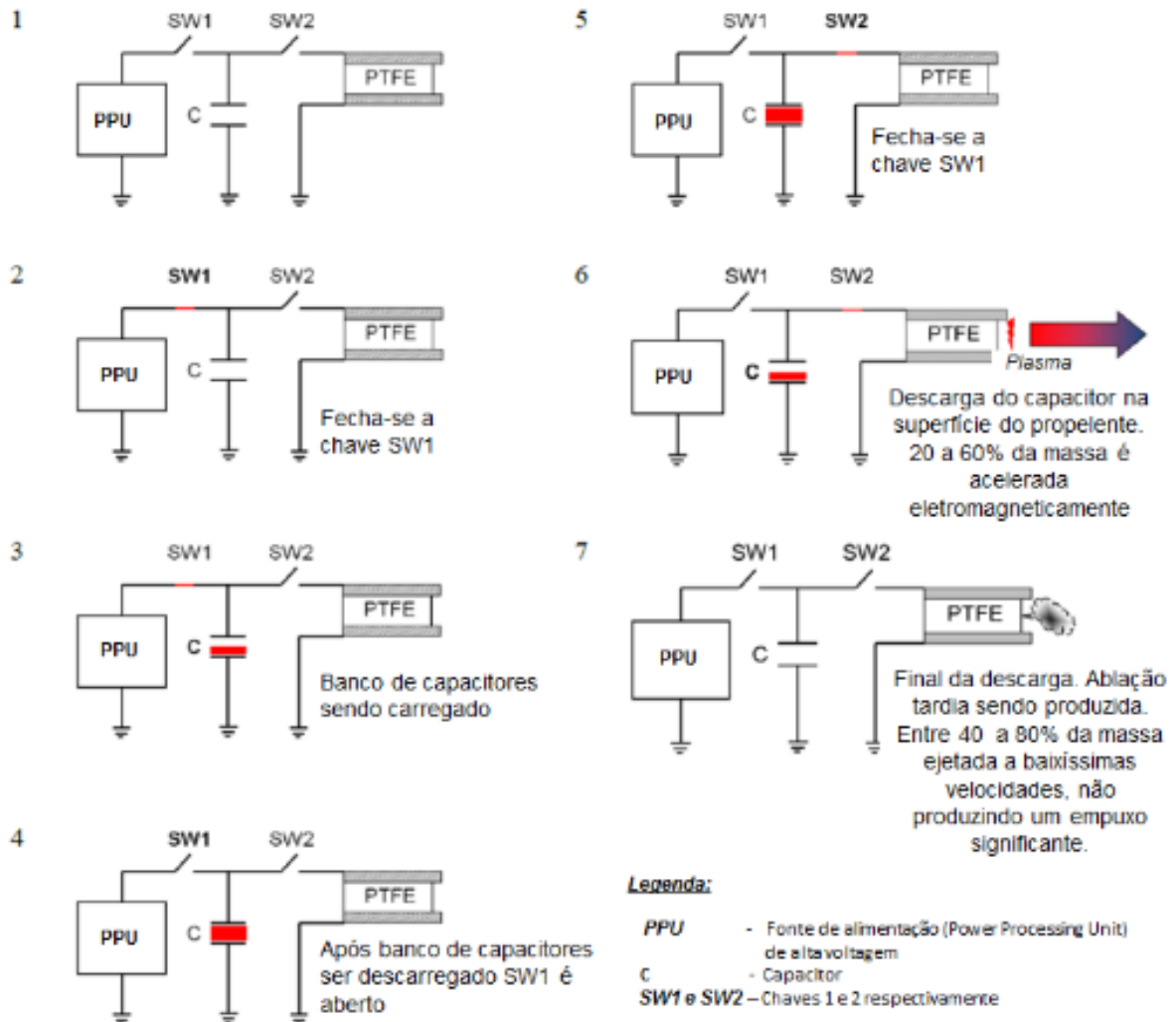
$$I_{bit} = \int_{t_0}^{t_1} |T| dt \quad (1.3)$$

Onde T representa o empuxo aplicado ao satélite e a integração é realizada durante o intervalo de tempo no qual esta força é aplicada. Este parâmetro indica o impulso aplicado ao satélite durante um único disparo do propulsor.

Com este parâmetro é possível saber qual a variação na quantidade de movimento linear do satélite, por pulso do PPT. Pode-se obter o equivalente ao empuxo médio do PPT sabendo-se a frequência de disparos (f) e o *impulse bit* (I_{bit}) (MARQUES, 2009):

$$F = f I_{bit} \quad (1.4)$$

Figura 1.3 - Ciclo do PPT com a formação de LTA.



Fonte: Marin (2014) Adaptado de Marques (2009).

1.3.4 Propelente

O propelente sólido utilizado pelo PPT é o PTFE (Politetrafluoretileno), também conhecido pelo nome Teflon®. Este é um polímero de cadeia longa, cujo monômero, o tetrafluoretileno, é $CF_2 = CF_2$, e o polímero $(-CF_2 - CF_2-)_n$. É um polímero praticamente inerte e não tóxico (BURTON; TURCHI, 1998).

Outros tipos de propelentes plásticos foram testados no PPT, incluindo Kynar®, Viton®, Fluorel®, Kel-F®, Genetron®, Halon®, Delrin®, CTFE-2300®, poli-

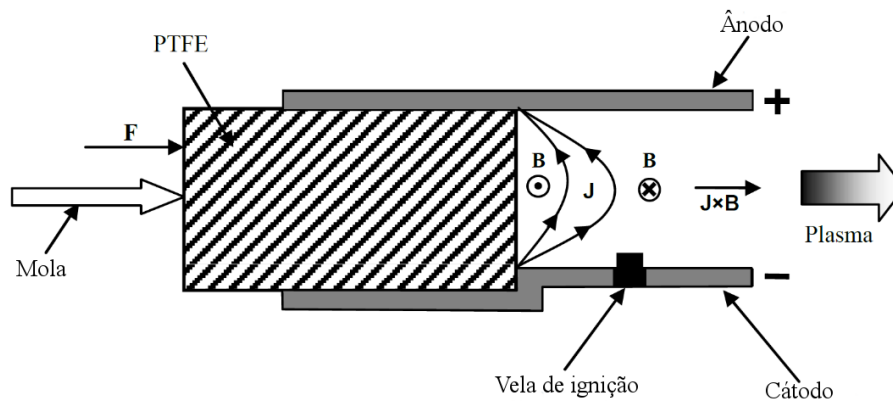
propileno e polietileno (BURTON; TURCHI, 1998), porém nenhum outro propelente mostrou desempenho melhor que o PTFE em termos de impulso específico e carbonização de superfície (MARQUES, 2009). Tentativas de combinar polímeros com metais também foram realizadas, contudo a temperatura de sinterização do PTFE ultrapassava a temperatura de fusão dos metais semeados, criando uma camada de metal derretido na superfície do PTFE, isto impedia a liberação de massa sublimada (BURTON; TURCHI, 1998). O teste de propelentes compostos por diferentes plásticos, em camadas alternantes, também foram realizadas, sem ganhos de desempenho.

1.3.5 Geometrias

O princípio de funcionamento do PPT pode ser replicado em diferentes arranjos geométricos, cada qual com suas peculiaridades. Nesta seção são apresentadas algumas geometrias.

A Figura 1.4 mostra o arranjo de PPT mais comum, o retangular. Esse arranjo foi usado para explicar o funcionamento básico do PPT. A única parte móvel no arranjo é a mola que mantém o propelente sempre na mesma posição (MARQUES, 2009).

Figura 1.4 - Geometria retangular.

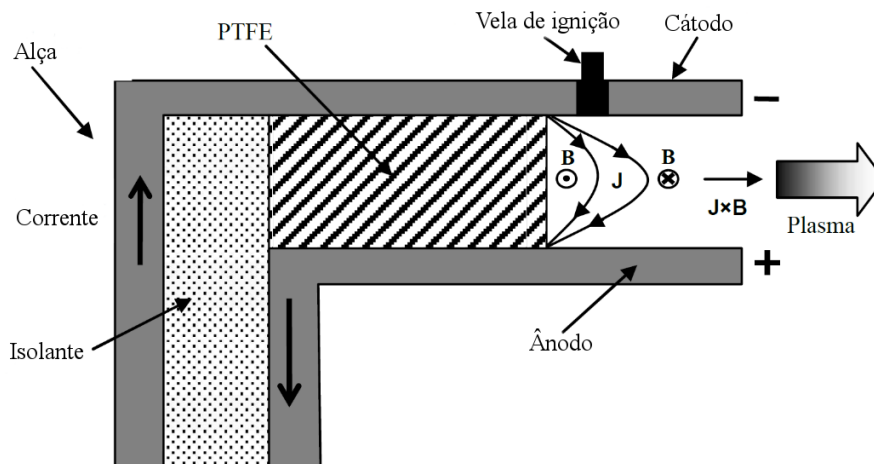


Fonte: Adaptado de Marques (2009).

A Figura 1.5 apresenta outra configuração retangular, desta vez sem a mola de alimentação. Neste arranjo, conforme o propelente é consumido, sua superfície se afasta da saída do PPT e da vela de ignição. Isto pode gerar variações de empuxo e dificultar a ignição da descarga principal. A possível vantagem desta configuração é

a ausência de partes móveis e a possibilidade da corrente do cátodo repelir a corrente no plasma, aumentando o forção eletromagnética que acelera o plasma (MARQUES, 2009).

Figura 1.5 - Geometria retangular sem mola.

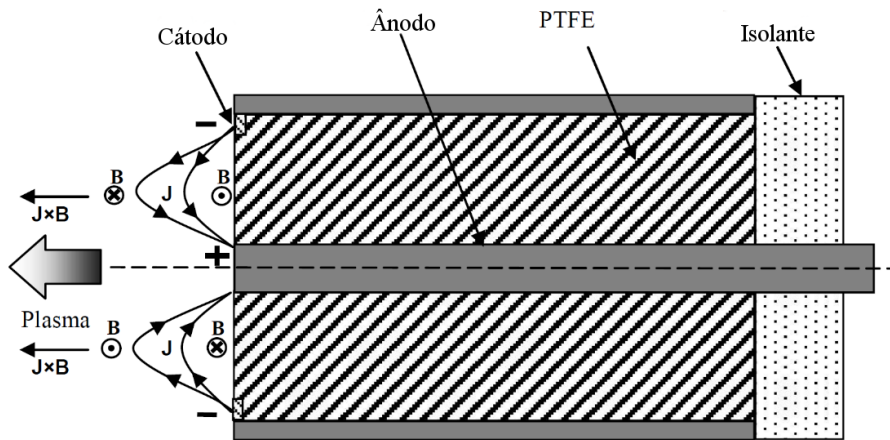


Fonte: Adaptado de Marques (2009).

A Figura 1.6 mostra a configuração coaxial do PPT. Neste esquema, não existe vela de ignição. A descarga principal ocorre mediante a aplicação de um campo elétrico forte o suficiente para romper a rigidez dielétrica do propelente. Para minimizar a tensão elétrica associada com este campo, minimiza-se a distância entre os eletrodos, conseqüentemente, diminuindo o tamanho do propulsor. Os PPTs deste tipo geralmente são micro propulsores (SELSTROM, 2010).

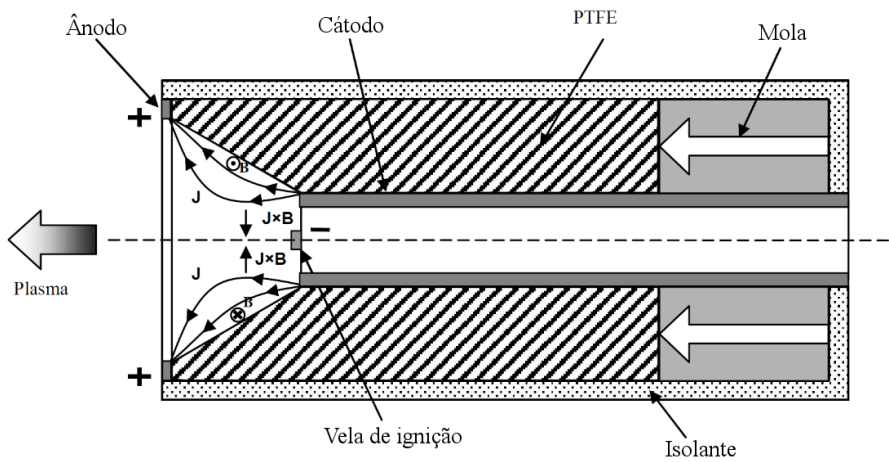
Na Figura 1.7 é apresentada outra configuração coaxial, desta vez com vela de ignição. É interessante notar que o propelente, além de oco, deve ter formato cônico e que o mesmo tende a manter o formato conforme é consumido (MARQUES, 2009).

Figura 1.6 - Geometria coaxial.



Fonte: Adaptado de Marques (2009).

Figura 1.7 - Geometria coaxial com propelente cônico e vela de ignição.



Fonte: Adaptado de Marques (2009).

1.3.6 Vantagens e Desvantagens

Entre as principais vantagens de propulsores elétricos do tipo PPT, estão (BURTON; TURCHI, 1998):

- Tempo para entrar em operação zero;

- Potência dissipada em espera zero;
- Inerte e seguro a falhas - quando não energizado não há torques ou forças;
- Boa escalabilidade;
- Usável em giro ou estabilização em três eixos de satélites;
- Todas as vantagens dos propelentes sólidos: sem tanques, linhas de alimentação, lacres, válvulas, consumo de propelente fácil de medir, compatível com vácuo, criogenia e gravidade zero, não-corrosivo, não-tóxico, não afetado por rápidas mudanças de temperatura, não afetado por variações de aceleração abruptas;
- Operação compatível com sistemas de controle baseados em lógica digital;
- Nível de empuxo variável;
- Desempenho adequado para uso em manutenção de posição e controle de atitude;
- Capaz de operar em um grande intervalo de temperaturas ambiente;
- Capacidade de controle do vetor de empuxo;
- Simplicidade e baixo custo;
- Operação imediata, mesmo depois de muitos anos de armazenamento;
- Controle fino de empuxo, possibilitando controle fino de atitude e posição.

Apesar das inúmeras vantagens dos propulsores do tipo PPT frente aos propulsores químicos, e mesmo frente a outros tipos de propulsores elétricos, existem desvantagens associados ao PPT que limitam muito seu escopo de possíveis aplicações espaciais. Entre estas desvantagens podemos citar (BURTON; TURCHI, 1998):

- Baixo empuxo;
- Baixa eficiência de propulsão (em torno de 10%);
- Difícil de ser modelado.

O desenvolvimento do PPT ao longo da história aconteceu de maneira experimental, com muito êxito. O fato de ter sido o primeiro tipo de propulsor elétrico no espaço, já em 1964, evidencia este fato. Desde sua concepção original, em 1962, com os projetos de Antropov e Khrabrov (BURTON; TURCHI, 1998), até hoje, os experimentos são a principal base da análise e desenvolvimento do PPT.

Entre as principais medições que se fazem no PPT, as medidas de velocidade são as que mais revelam a complexidade do processo de aceleração que ocorre no propulsor (BURTON; TURCHI, 1998). Devido a presença de campos elétricos e magnéticos, em um meio com gases neutros e ionizados em diferentes estados de carga, a distribuição de velocidades no PPT é bastante ampla. Medidas de velocidade no propulsor LES-6 indicaram velocidades de 30-35 km/s para espécies dupla e triplamente ionizadas de carbono (C^{++} , C^{+++}) e flúor (F^{++} , F^{+++}) (BURTON; TURCHI, 1998)(GUMAN et al., 1970).

Medidas de velocidade no propulsor russo MIPD-3, também indicaram velocidade de exaustão de íons de 25-35 km/s e velocidades de gases neutros de 15 km/s, sendo a velocidade média dos gases neutros e ionizados de 11 km/s. As técnicas usadas para medição de velocidade foram: espectroscopia de emissão, interferometria laser, sonda de langmuir e sondas magnéticas (POPOV et al., 1997).

Estudos das características de pluma foram realizados por Li et al. (2003). A contaminação de um prato de alumínio, a 1100 mm do propulsor, indicou que o elemento dominante era o carbono e que a proporção de elementos contaminantes não era alterada com a distância ao propulsor. Fotografias de diversos instantes da descarga elétrica e do efeito da inclinação do motor na pluma também foram realizadas.

Medições da evolução espacial e temporal do campo magnético, ao longo da descarga elétrica, foram realizadas por Berkery e Choueiri (2003). A partir destas medições foi calculada a distribuição de corrente elétrica na descarga, que se revelou bastante complexa nos momentos iniciais da mesma.

Diversas leis empíricas para o cálculo do *impulse bit* foram desenvolvidas para o PPT, levando-se em consideração a parte eletromagnética e a parte gás-dinâmica do impulso. Contudo, esses modelos são simplificações pobres do universo de fenômenos físicos que ocorrem no PPT, entre os quais podemos destacar (BURTON; TURCHI, 1998):

- (1) Transferência de calor;

- (2) Fenômenos eletromagnéticos (geração e distribuição de campos elétricos, magnéticos e de corrente elétrica);
- (3) Reações químicas (mudança de fase, despolimerização e ionização);
- (4) Mecânica de fluidos (escoamento de fluido com variação de propriedades de ponto-a-ponto).

A complexidade de fenômenos ocorrendo em um propulsor PPT é a principal responsável pela falta de modelos teóricos mais completos sobre o funcionamento do mesmo. Sem estes modelos, que descrevam em detalhes o processo de geração e aceleração do plasma, qualquer tentativa de otimização e escalonamento do propulsor fica restrita aos limitados modelos experimentais desenvolvidos até o momento (TURCHI; MIKELLIDES, 1995).

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo alguns conceitos básicos necessários para o entendimento do problema de simulação do PPT e sua posterior solução serão descritos.

2.1 Ablação

Um dos fenômenos de maior importância no funcionamento de um PPT é a ablação do propelente e conseqüente geração de plasma. Sendo assim, o entendimento deste mecanismo é de extrema importância para a criação de modelos analíticos e a simulação numérica deste processo (BURTON; TURCHI, 1998).

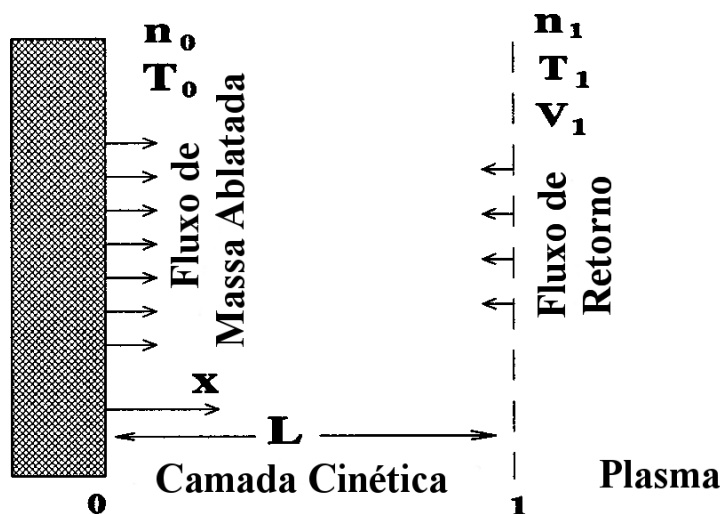
Além do propulsor de plasma pulsado, a ablação controlada por descarga elétrica tem um interesse geral em diversas aplicações, tais como fusíveis elétricos, disjuntores, raio X suave e fontes ultravioletas extremas (BURTON; TURCHI, 1998), (KEIDAR et al., 2001), (KEIDAR et al., 2001), (ZAGHLOUL, 2004). Nestes dispositivos, a energia de descarga é principalmente dissipada pela ablação do material de parede, que então forma o componente principal do plasma de descarga. O vapor ablado aumenta a pressão dentro da cavidade ou capilar e o plasma é expelido pela saída.

Anteriormente, a maioria dos modelos de plasma de descarga controlada empregou a relação de Langmuir (LANGMUIR, 1913), que é limitada ao caso de ablação de material em vácuo. Para as condições de um propulsor de plasma pulsado, essa abordagem também foi usada recentemente (KEIDAR et al., 2000), (MIKELLIDES, 1999). No entanto, o processo de ablação controlada por descarga deve ser descrito levando-se em conta o fato de que, na cavidade do propulsor o vapor não se expande em vácuo, mas sim em um volume de descarga.

2.1.1 Camada cinética

Anisimov (1968) desenvolveu um modelo de ablação para o caso da vaporização de um metal no vácuo. Este modelo considera que a expansão do vapor acontece através de uma onda de rarefação e relaciona as condições de contorno da superfície do metal com as condições na onda de expansão rarefeita. O modelo analítico é unidimensional e considera que entre vizinhança da superfície sendo vaporizada e a massa de plasma, há uma região de poucos livres-caminhos-médios de comprimento onde as condições termodinâmicas estão substancialmente fora do equilíbrio. Esta região é chamada de camada cinética e uma representação da mesma é dada na Figura 2.1.

Figura 2.1 - Representação esquemática da camada cinética.



Fonte: Adaptada de Keidar et al. (2001)

A equação de Boltzmann deve ser resolvida para determinar a estrutura desta camada e os valores das variáveis hidrodinâmicas dos dois lados desta descontinuidade. Porém, a equação de Boltzmann não admite solução analítica. A situação assemelha-se, neste caso, a uma forte onda de choque (ou melhor, uma "forte onda de rarefação"). Portanto, o método mais apropriado é o que consiste na solução da equação cinética sugerida por Mott-Smith (GOMBOSI, 1994) em seu estudo da estrutura de ondas de choque. A característica especial deste método é uma aproximação da função de distribuição de velocidades dentro da região de descontinuidade pela soma das funções de distribuição antes e depois da descontinuidade com coeficientes dependentes das coordenadas. Tal aproximação pode ser aceitável se a região que contém a mudança da função de distribuição for estreita o suficiente (ANISIMOV, 1968).

A função de distribuição de velocidades dentro da camada cinética é aproximada através da Equação 2.1 (ANISIMOV, 1968).

$$f(x, \mathbf{v}) = \alpha(x)f_0(\mathbf{v}) + [1 - \alpha(x)]f_1(\mathbf{v}) \quad (2.1)$$

onde

$$f_0(\mathbf{v}) = \begin{cases} n_0 \left(\frac{m}{2\pi k T_0} \right)^{\frac{3}{2}} \exp\left(-\frac{m(v_x^2 + v_y^2 + v_z^2)}{2kT_0}\right), & v_x > 0 \\ \beta f_1(\mathbf{v}) & v_x < 0, \end{cases}$$

$$f_1(\mathbf{v}) = n_1 \left(\frac{m}{2\pi k T_1} \right)^{\frac{3}{2}} \exp\left(-m \frac{(v_x - V_1)^2 + v_y^2 + v_z^2}{2kT_1}\right)$$

$\alpha(x)$ é a função desconhecida que satisfaz as condições $\alpha(0) = 1$ e $\alpha(\infty) = 0$, T_0 é a temperatura da superfície sofrendo ablação, k é a constante de Boltzmann, m é a massa atômica (considerada constante entre os pontos 0 e 1), n_0 é a densidade numérica de vapor saturado nessa temperatura, n_1 é a densidade numérica no ponto 1, T_1 é a temperatura no ponto 1, e V_1 é a velocidade no ponto 1. A expressão para $f_0(\mathbf{v})$ leva em consideração que os átomos vaporizados têm distribuição de velocidades maxwelliana em uma temperatura igual a temperatura da superfície. Além disso, supõe-se que a função de distribuição de velocidades para as partículas que retornam ($V_1 < 0$) é $\beta f_0(\mathbf{v})$, onde β é o coeficiente de proporcionalidade (ANISIMOV, 1968).

Para achar a relação entre os parâmetros da superfície ablacionada¹ e o plasma (fronteiras 0 e 1 na Figura 2.1), faz-se uso das equações de conservação da massa, momentum e energia através da camada cinética. Integrando a Equação 2.1 em relação as velocidades e igualando-se o resultado nas fronteiras 0 e 1, tem-se as seguintes equações de conservação (ANISIMOV, 1968):

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} v_x f(0, \mathbf{v}) dv_x dv_y dv_z = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} v_x f(1, \mathbf{v}) dv_x dv_y dv_z \quad (2.2)$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} v_x^2 f(0, \mathbf{v}) dv_x dv_y dv_z = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} v_x^2 f(1, \mathbf{v}) dv_x dv_y dv_z \quad (2.3)$$

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} v_x v^2 f(0, \mathbf{v}) dv_x dv_y dv_z = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} v_x v^2 f(1, \mathbf{v}) dv_x dv_y dv_z \quad (2.4)$$

Sendo as Equações 2.2, 2.3 e 2.4, a conservação da massa, momentum e energia, res-

¹(REZENDE, 1995)

pectivamente. Computando as integrais acima tem-se o seguinte resultado (KEIDAR et al., 2001):

$$\frac{n_0}{2(\pi d_0)^{0.5}} = n_1 V_1 + \beta \frac{n_1}{2(\pi d_1)^{0.5}} \{\exp(-\alpha^2) - \alpha \pi^{0.5} \operatorname{erfc}(\alpha)\} \quad (2.5)$$

$$\frac{n_0}{4d_0} = \frac{n_1}{2d_1} \{(1 + 2\alpha^2) - \beta[(0.5 + \alpha^2) \operatorname{erfc}(\alpha) - \alpha \exp(-\alpha^2)/\pi^{0.5}]\} \quad (2.6)$$

$$\frac{n_0}{(\pi d_0)^{1.5}} = \frac{n_1}{\pi(d_1)^{1.5}} \left[\alpha(\alpha^2 + 2.5) - \beta/2 \{ (2.5 + \alpha^2) \alpha \operatorname{erfc}(\alpha) - \frac{(2 + \alpha^2) \exp(-\alpha^2)}{\pi^{0.5}} \} \right] \quad (2.7)$$

Onde $d_0 = m/2kT_0$, $d_1 = m/2kT_1$, $\alpha = V_1/\sqrt{2kT_1/m}$, $\operatorname{erfc}(\alpha) = 1 - \operatorname{erf}(\alpha)$, $\operatorname{erf}(\alpha)$ é a função erro, T_0 é temperatura da superfície, e n_0 é a densidade de equilíbrio. Neste sistema de equações, a velocidade na borda da camada de cinética (V_1 na fronteira 1) é um parâmetro livre. Este parâmetro afeta fortemente a densidade numérica e a temperatura na fronteira 1 (n_1 e T_1) Keidar et al. (2001).

O principal resultado do trabalho de Anisimov (1968) é o cálculo do fluxo máximo de átomos retornados para a superfície, que foi encontrado em cerca de 18% do fluxo de átomos vaporizados. Este resultado foi obtido sob a suposição de que a velocidade do fluxo atômico é igual à velocidade do som no limite externo da camada cinética. Em muitas situações físicas, no entanto, a expansão do vapor não é pela velocidade do som, pois há um plasma denso no volume de descarga. Beilis (1996) analisou a vaporização de metais em plasmas de descarga no caso de uma mancha no catodo de arco a vácuo. Ele concluiu que os parâmetros no limite exterior da camada cinética estão próximos de seus valores de equilíbrio e que a velocidade no limite externo da camada cinética é muito menor que a velocidade do som. Em ambas as análises mencionadas acima, nenhuma informação é fornecida sobre a mudança da função de distribuição de velocidade da partícula de um estado de não-equilíbrio para um estado de equilíbrio dentro da camada cinética (KEIDAR et al., 2001).

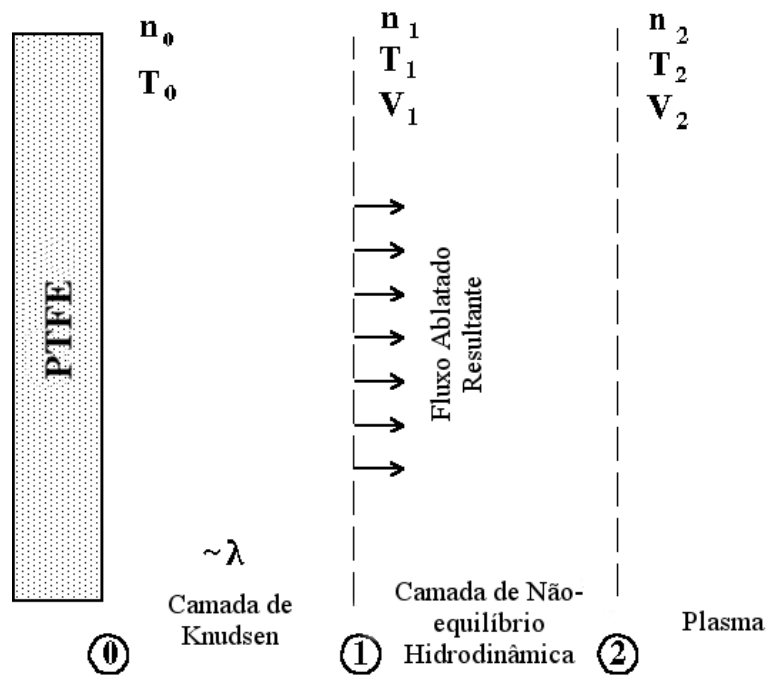
2.1.2 Camada hidrodinâmica

Keidar et al. (2001) estenderam este modelo de camada única para um modelo de duas camadas, como indicado na Figura 2.2. Neste modelo, tem-se duas regiões distintas entre a massa de plasma e a superfície sendo ablacionada. A primeira região

é camada cinética já mencionada acima. A segunda região considerada é a camada hidrodinâmica, dominada por colisões e fora do equilíbrio termodinâmico, ou seja, a temperatura das partículas pesadas e dos elétrons diferem. Ao fim desta região (fronteira 2) é assumido que seja alcançado o equilíbrio termodinâmico entre todas as espécies (íons, elétrons e partículas neutras).

A ideia deste novo modelo é estabelecer uma transição entre o fim da camada cinética e o início do plasma quase-neutro, onde a temperatura dos íons, elétrons e partículas neutras é a mesma. Para relacionar os parâmetros entre as fronteiras 1 e 2 da Figura 2.2, faz-se uso das equações de conservação de massa e momentum contínuas, já que se trata de uma região hidrodinâmica (KEIDAR et al., 2001).

Figura 2.2 - Representação esquemática da camada cinética e hidrodinâmica.



Fonte: Adaptada de Keidar et al. (2001)

A equação de conservação da massa, unidimensional, para partículas pesadas (íons, neutras) é dada por (KEIDAR et al., 2004):

$$\frac{d(nV)}{dx} = 0 \quad (2.8)$$

onde n é densidade numérica do plasma and V é a velocidade. A equação de conservação do momentum, unidimensional, é dada por (KEIDAR et al., 2004):

$$M(nV)\frac{dV}{dx} + \frac{d(nkT)}{dx} = 0 \quad (2.9)$$

onde M é a massa atômica e T é a temperatura do plasma. Assumindo um plasma fracamente ionizado na camada hidrodinâmica, a integração das equações de conservação da massa e momentum leva as seguintes relações entre os parâmetros nas fronteiras 1 e 2 (KEIDAR et al., 2004):

$$\begin{aligned} n_1 V_1 &= n_2 V_2 \\ n_1 k T_1 + m n_1 V_1^2 &= n_2 k T_2 + m n_2 V_2^2 \end{aligned} \quad (2.10)$$

Combinando estas duas equações, chega-se a seguinte expressão para a velocidade na fronteira 1 (KEIDAR et al., 2004) (ZAGHLOUL, 2004):

$$\alpha^2 = \frac{V_1^2}{(2kT_1/m)} = \frac{1}{2} \frac{\left(\frac{n_2 T_2}{n_1 T_1} - 1\right)}{\left(1 - \frac{n_1}{n_2}\right)} \quad (2.11)$$

O termo do lado esquerdo da Equação 2.11 é o parâmetro livre $\alpha = V_1/\sqrt{2kT_1/m}$ citado anteriormente. Com esta equação torna-se possível calcular a taxa de ablação, que é proporcional a $n_1 V_1$.

Este sistema de equações é fechado com a fórmula da pressão de vapor de equilíbrio, que para o PTFE é dada por (KEIDAR et al., 2001) (MIKELLIDES, 1999):

$$P_0 = P_c \exp(-T_c/T_0) = n_0 k T_0 \quad (2.12)$$

onde P_0 é a pressão de equilíbrio do PTFE, $P_c = 1,84e15 \text{ Nm}^{-2}$ e $T_c = 20815 \text{ K}$.

2.1.3 Correções e conjunto de equações modificadas

Zaghloul (2004) melhorou o modelo de duas camadas de Keidar et al. (2001), levando em consideração fenômenos físicos negligenciados, que impactam significativamente

o resultado da análise com este modelo, bem como entendimento do fenômeno sob consideração.

As expressões derivadas no conjunto de Equações 2.5-2.7 são válidas apenas para um vapor monoatômico (gás). Na derivação dessas expressões, a conservação do fluxo de massa, fluxo de momento e fluxo de energia translacional foram assumidas e estabelecidas através da camada de Knudsen. No entanto, para um gás poliatômico como o vapor de PTFE, deve-se adicionar à energia translacional, a contribuição de energia devido aos graus de liberdade internos, aplicando-se a conservação do fluxo de energia em seguida. A contribuição dos graus internos de liberdade por unidade de massa, $U_{idf}(T)$, pode ser expressa como (ZAGHLOUL, 2004):

$$U_{idf}(T) = \frac{(5 - 3\gamma) kT}{2(\gamma - 1) m} \quad (2.13)$$

sendo γ a razão dos calores específicos, m é massa molecular média, T a temperatura e k a constante de Boltzmann. Adicionando $U_{idf}(T)$ à energia translacional, a equação para conservação do fluxo de energia é escrita como (ZAGHLOUL, 2004):

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} v_x (U_{idf}(T) + v^2) f(x, \mathbf{v}) dv_x dv_y dv_z = \text{fluxo de energia} \quad (2.14)$$

Outra importante fonte de erros nas equações é a suposição que a massa das partículas na superfície do PTFE, na borda da camada cinética e no plasma seja sempre a mesma (ZAGHLOUL, 2004). Na prática, no entanto, podem haver variações na massa das partículas entre os pontos 0, 1 e 2 representados na Figura 2.2. A massa das partículas nestes pontos são designadas m_0 , m_1 e m_2 , respectivamente.

Levando-se em consideração os fatores descritos acima, as integrais de conservação 2.2, 2.3 e 2.4 são recalculadas, obtendo-se assim (ZAGHLOUL, 2004):

$$\frac{m_0 n_0}{2(\pi d_0)^{0.5}} = m_1 n_1 V_1 + \beta \frac{m_1 n_1}{2(\pi d_1)^{0.5}} \{ \exp(-\alpha^2) - \alpha \pi^{0.5} \text{erfc}(\alpha) \} \quad (2.15)$$

$$\frac{m_0 n_0}{4d_0} = \frac{m_1 n_1}{2d_1} \{ (1 + 2\alpha^2) - \beta [(0.5 + \alpha^2) \text{erfc}(\alpha) - \alpha \exp(-\alpha^2) / \pi^{0.5}] \} \quad (2.16)$$

$$\begin{aligned} \frac{m_0 n_0}{(\pi d_0)^{1.5}} = & \left\{ \frac{4(\gamma - 1)}{(\gamma + 1)} \right\} \frac{m_1 n_1}{\pi (d_1)^{1.5}} \left[\alpha \left(\alpha^2 + \frac{\gamma}{\gamma - 1} \right) - (\beta/2) \right. \\ & \left. \times \left\{ \left(\frac{\gamma}{\gamma - 1} + \alpha^2 \right) \alpha \operatorname{erfc}(\alpha) - \left(\frac{\gamma + 1}{2(\gamma - 1)} + \alpha^2 \right) \exp(-\alpha^2)/\pi^{0.5} \right\} \right] \end{aligned} \quad (2.17)$$

Para o caso de um gás monoatômico, onde $\gamma = 5/3$ e $m_0 = m_1$, $U_{idf}(T)$ é igual a zero e as Equações 2.15-2.17 reduzem-se para o conjunto de Equações 2.5-2.7. Estas equações podem ser reescritas em uma forma mais comumente encontrada na literatura (KNIGHT, 1979):

$$\frac{m_0 T_1}{m_1 T_0} = \left[\left(1 + \pi \left(\frac{\gamma - 1}{\gamma + 1} \frac{\alpha}{2} \right)^2 \right)^{1/2} - \pi^{1/2} \frac{\gamma - 1}{\gamma + 1} \frac{\alpha}{2} \right]^2 \quad (2.18)$$

$$\begin{aligned} \frac{\rho_1}{\rho_0} = \frac{m_1 n_1}{m_0 n_0} = & \left(\frac{m_1 T_0}{m_0 T_1} \right)^{1/2} \left[\left(\alpha^2 + \frac{1}{2} \right) e^{\alpha^2} \operatorname{erfc}(\alpha) - \frac{\alpha}{\pi^{1/2}} \right] \\ & + \frac{m_1 T_0}{2m_0 T_1} [1 - \pi^{1/2} \alpha e^{\alpha^2} \operatorname{erfc}(\alpha)] \end{aligned} \quad (2.19)$$

$$\beta = \left[(2\alpha^2 + 1) - \alpha \pi^{1/2} \left(\frac{m_1 T_0}{m_0 T_1} \right)^{1/2} \right] e^{\alpha^2} \frac{m_0 n_0}{m_1 n_1} \left(\frac{m_1 T_0}{m_0 T_1} \right)^{1/2} \quad (2.20)$$

A Equação 2.18 permite calcular a temperatura T_1 em função da temperatura T_0 e do parâmetro α . Com a temperatura T_1 calculada pode-se usar T_0 e α na Equação 2.19 para se obter a densidade $\rho_1 = m_1 n_1$. O coeficiente β também pode ser calculado desta maneira (ZAGHLOUL, 2004) (KNIGHT, 1979).

2.1.3.1 Combinação das camadas cinética e hidrodinâmica

Como consequência deste novo modelo com correções, a Equação 2.11, decorrente da combinação da camada cinética e hidrodinâmica, é modificada para (ZAGHLOUL, 2004):

$$\alpha^2 = \frac{V_1^2}{(2kT_1/m)} = \frac{1}{2} \frac{\left(\frac{n_2 T_2}{n_1 T_1} - 1\right)}{\left(1 - \frac{m_1 n_1}{m_2 n_2}\right)} = \frac{1}{2} \frac{\left(\frac{P_2}{P_1} - 1\right)}{\left(1 - \frac{\rho_1}{\rho_2}\right)} \quad (2.21)$$

Esta equação modificada permite, novamente, calcular a taxa de ablação como função dos parâmetros n_2 , T_2 e T_0 . Mas, desta vez, é possível usar diferentes massas de partícula nos pontos 1 e 2 (ZAGHLOUL, 2004).

A Equação 2.21 terá uma solução real se $P_2 \geq P_1$ e $(\rho_2 = m_2 n_2) > (\rho_1 = m_1 n_1)$ ou $P_1 \geq P_2$ e $(\rho_1 = m_1 n_1) > (\rho_2 = m_2 n_2)$, contudo, e porque não houve consideração de evaporação difusa no modelo sob consideração, somente a segunda possibilidade é fisicamente viável, já que o escoamento ablativo não será possível para $P_1 \leq P_2$. Com as equações de Anisimov (Eq 2.5-2.7) pode-se demonstrar que $n_0 T_0 \geq n_1 T_1$. Portanto, uma condição necessária para obter uma solução fisicamente aceitável pode ser derivada como (ZAGHLOUL, 2004):

$$n_2 \leq n_{crit} = \frac{n_0 T_0}{T_2} = \frac{P_c}{k T_2} \exp(-T_c/T_0) \quad (2.22)$$

2.2 Simulação do PPT

Historicamente, o desenvolvimento científico aconteceu apoiado em dois pilares principais: experimentos puros e teoria pura (ANDERSON, 1995). Os experimentos servem como ponto de partida para formulação de leis e teorias que, por sua vez, permitem explicar fenômenos diversos através de um mesmo conjunto básico de regras, além de serem capazes de prever novos fenômenos. Estas previsões são, então, confirmadas ou refutadas por novos experimentos e observações, fechando o ciclo do método científico.

A partir do surgimento e, principalmente, da massificação dos computadores digitais, uma nova abordagem surgiu como uma terceira via ao desenvolvimento científico e tecnológico: a simulação numérica. Esta nova abordagem não substituiu de maneira nenhuma a experimentação e análise teórica dos fenômenos físicos. Contudo, ela provê uma nova ferramenta para a pesquisa científica e uma nova ferramenta de projeto para a engenharia (ANDERSON, 1995).

A simulação numérica nada mais é do que a utilização de algoritmos e métodos matemáticos para a resolução de equações diferenciais que descrevem o comportamento

de sistemas físicos (ANDERSON, 1995).

Com a simulação numérica computadorizada se faz possível a realização de experimentos numéricos. Ou seja, uma vez que existe um programa que resolva as equações que representam o modelo físico considerado, pode-se usá-lo para avaliar o comportamento do sistema da mesma forma que se faria em um experimento físico (ANDERSON, 1995).

No caso específico do PPT, onde uma grande quantidade de fenômenos físicos distintos acontece, a simulação por computador se faz extremamente necessária. As leis experimentais desenvolvidas para o PPT e as representações em forma de circuitos agrupados, não capturam adequadamente os processos físicos que acontecem neste tipo de propulsor (ANDERSON, 1995).

Portanto, se faz necessário o desenvolvimento de uma modelagem teórica que incorpore estes processos. Por exemplo, a distribuição espaço-temporal da corrente elétrica, e subsequente geração do campo magnético são processos que exigem a formulação, no mínimo, ao nível da magneto-hidrodinâmica (MHD) (BURTON; TURCHI, 1998).

Para análise deste tipo de formulação teórica se faz necessária a simulação deste conjunto de equações (BURTON; TURCHI, 1998). Avanços neste sentido já foram realizados com o auxílio do código MACH2.

2.2.1 MACH2

O código MACH2 foi desenvolvido pelo Centro de Computação e Teoria de Plasma do Laboratório de Pesquisa da Força Aérea Americana. É um código descendente do código MOQUI, escrito por J. Brackbill no Laboratório Nacional de Los Alamos (LANL, na sigla em inglês) (MIKELLIDES, 1999).

É um código MHD, dependente do tempo, bidimensional, com capacidade de cálculo em geometrias complexas com simetria cilíndrica e simetria paralela a um plano. Simula um único fluido, mas tem capacidade para associar-lhe múltiplas temperaturas (íons e elétrons). É um código com formulação contínua e, portanto, incapaz de lidar com rarefação apropriadamente (MIKELLIDES, 1999).

Tem sido usado em diversas simulações complexas envolvendo problemas de hidrodinâmica e plasma, incluindo, toroides compactos, propulsores magneto-plasmodinâmicos, propulsores de ablação a laser (MOELLER; CHANG, 2007), entre

outros.

É capaz de simular uma ampla gama de propulsores, tanto com campos externos aplicados como com campos auto induzidos, em regime permanente ou pulsado. Resultados de simulação com MACH2 guiaram modelagens analíticas que permitiram, entre outros, a obtenção de relações de escalonamento para o comportamento do propulsor como função da corrente de entrada, campo externo aplicado, tipo de propelente e geometria do propulsor (TURCHI; MIKELLIDES, 1995).

Turchi e Mikellides (1995) usaram o código MACH2 como uma ferramenta de modelagem teórica do PPT. Isto permitiu a análise e otimização de operação do propulsor, através da modificação da corrente elétrica de entrada e modificações geométricas. As simulações alcançaram resultados muitos satisfatórios, apesar das limitações do código.

Apesar do sucesso obtido na simulação do PPT, o código MACH2 foi desenvolvido em um laboratório militar, portanto, seu uso é extremamente restrito e controlado. As únicas informações disponíveis sobre o código são suas características gerais e algumas equações que são resolvidas pelo software, mas sem entrar em detalhes de implementação das equações, métodos de resolução e acoplamento dos diversos processos físicos que são simulados.

Sendo assim, o desenvolvimento de um código aberto, capaz de simular o PPT seria de grande valia para a análise, pesquisa e aperfeiçoamento deste propulsor. Também há a possibilidade de simulação de outros tipos de propulsores e fenômenos de plasma, dependendo da complexidade e da base numérica na qual o programa é criado.

2.3 Equações Governantes da Mecânica dos Fluidos

Fluidos, que denotam líquidos, gases e plasmas, são substâncias que não mudam permanentemente sob estresse (força por unidade de área). Enquanto um sólido resiste a um cisalhamento aplicado ou a uma tensão tangencial por deformação, um fluido não resiste e uma tensão de cisalhamento aplicada a um fluido o coloca em movimento (MOUKALLED *et al.*, 2016).

As equações governantes da mecânica de fluidos são derivadas a partir do princípio da conservação. O princípio da conservação afirma que, para um sistema isolado, certas quantidades mensuráveis físicas são conservadas em uma região local. Este princípio de conservação ou lei de conservação é um axioma que não pode ser com-

provado matematicamente, mas pode ser expresso por uma relação matemática. Leis deste tipo governam várias quantidades físicas tais como massa, momento e energia. (MOUKALLED et al., 2016).

As equações diferenciais governantes que devem ser resolvidas para a descrição de um fluido são dadas por (GREENSHIELDS et al., 2008):

- Conservação da massa:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot [\rho \mathbf{u}] = 0 \quad (2.23)$$

- Conservação do momentum:

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot [\mathbf{u}(\rho \mathbf{u})] + \nabla p + \nabla \cdot \mathbf{T} = \mathbf{0} \quad (2.24)$$

- Conservação da energia total:

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot [\mathbf{u}(\rho E)] + \nabla \cdot [\mathbf{u}p] + \nabla \cdot (\mathbf{T} \cdot \mathbf{u}) + \nabla \cdot \mathbf{j} = 0 \quad (2.25)$$

onde ρ é a densidade de massa, \mathbf{u} é a velocidade do fluído, p é pressão; a densidade de energia total $E = e + |\mathbf{u}^2|/2$ sendo e a energia interna específica, \mathbf{j} é o fluxo de calor difusivo, \mathbf{T} é o tensor de estresse viscoso, definido como positivo na compressão.

No caso de um escoamento invíscido, $\mathbf{T}=\mathbf{0}$ e $\mathbf{j}=\mathbf{0}$, e as Equações de 2.23-2.25 reduzem-se para as equações de Euler. Para escoamentos viscosos, o tensor de estresse pode ser representado através da lei de Newton (Fluidos Newtonianos) (GREENSHIELDS et al., 2008):

$$\mathbf{T} = -2\mu dev(\mathbf{D}) \quad (2.26)$$

onde μ é viscosidade dinâmica, o tensor de deformação do gradiente $\mathbf{D} = \frac{1}{2} [\nabla \mathbf{u} + (\nabla \mathbf{u})^T]$ e seu componente deviatórico $dev(\mathbf{D}) = \mathbf{D} - (\frac{1}{3})tr(\mathbf{D})\mathbf{I}$, onde \mathbf{I} é o tensor unitário. O fluxo de calor difusivo pode ser representado com a lei de Fourier (GREENSHIELDS et al., 2008):

$$\mathbf{j} = -k\nabla T \quad (2.27)$$

onde T é a temperatura e k é a condutividade térmica. As equações de Navier-Stokes são produzidas através da substituição das relações constitutivas da viscosidade e

condução de calor, as Equações 2.26 e 2.27, respectivamente, nas equações governantes.

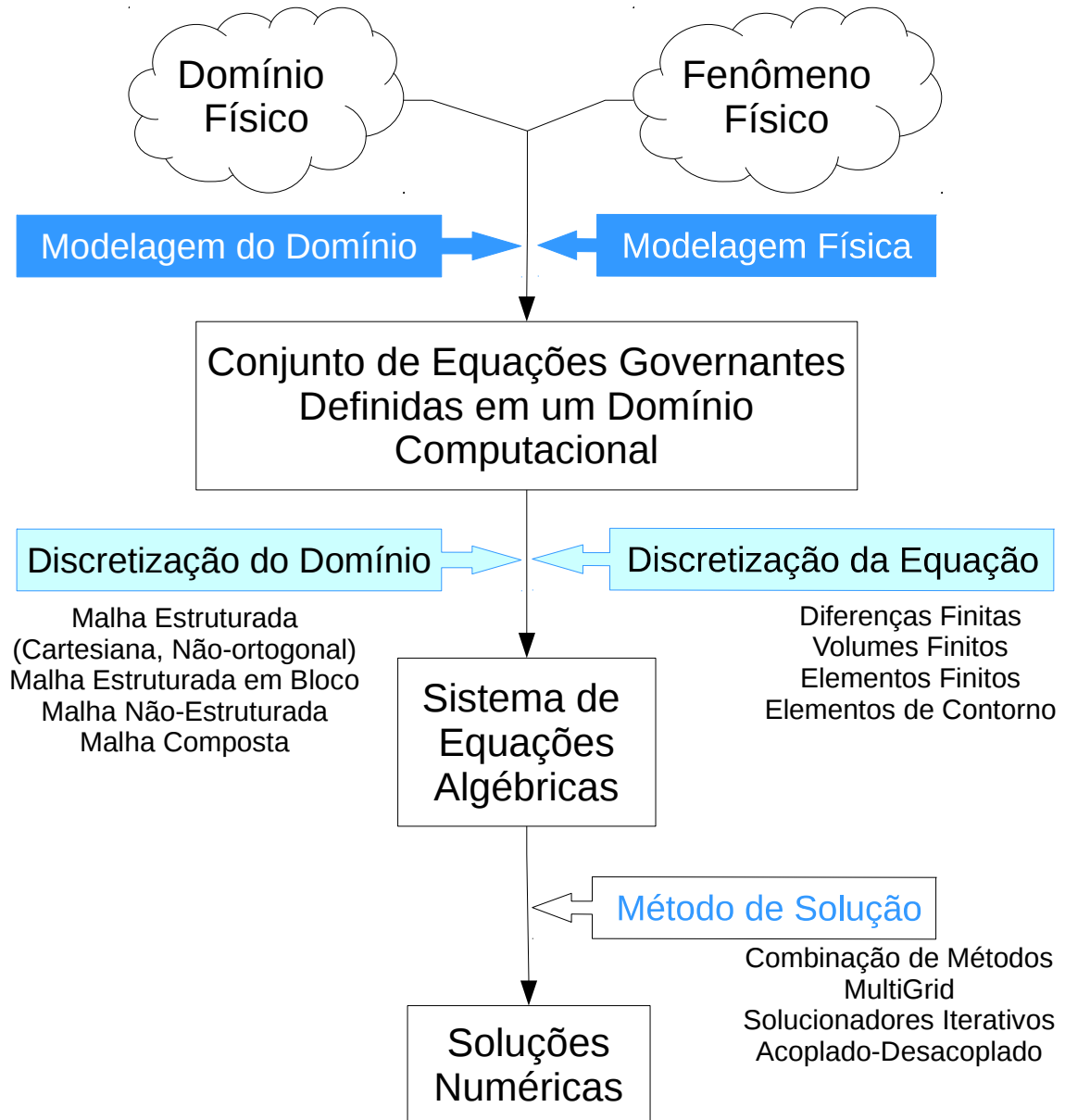
Neste trabalho considera-se um gás caloricamente perfeito para o qual $p = \rho RT$ e $e = c_v T = (\gamma - 1)RT$, onde R é constante do gás e $\gamma = c_p/c_v$ é a razão dos calores específicos a pressão e volume constantes, c_p e c_v , respectivamente (GREENSHIELDS et al., 2008) (MOUKALLED et al., 2016).

2.4 O Processo de Discretização

A solução numérica de uma equação diferencial parcial consiste em encontrar os valores da variável dependente Ψ em pontos específicos sobre o domínio de interesse, a partir dos quais sua distribuição sobre este domínio pode ser aproximada. Esses pontos são chamados de elementos ou nós de malha, e são o resultado da discretização da geometria original em um conjunto de formas geométricas não sobrepostas. Os nós ou elementos resultantes são geralmente posicionados em centroides de células ou em vértices, dependendo do procedimento de discretização adotado. Em todos os métodos, o foco está em substituir a solução exata contínua da equação diferencial parcial por valores discretos. A distribuição de Ψ é, portanto, discretizada (MOUKALLED et al., 2016).

Este processo de conversão da equação governante em um conjunto de equações algébricas sobre os valores discretos de Ψ é conhecido como discretização e os métodos específicos empregados para realizar essa conversão são chamados de métodos de discretização. Os valores discretos de Ψ são normalmente calculados resolvendo-se um conjunto de equações algébricas que relacionam os valores dos elementos de malha vizinhos entre si; estas equações discretizadas ou algébricas são derivadas da equação de conservação que governa Ψ . Uma vez que os valores de Ψ são calculados, os dados são processados para extrair qualquer informação necessária. Os vários estágios do processo de discretização estão ilustrados sucintamente na Figura 2.3 (MOUKALLED et al., 2016).

Figura 2.3 - Resumo do processo de discretização.



Fonte: Adaptada de Moukalled et al. (2016)

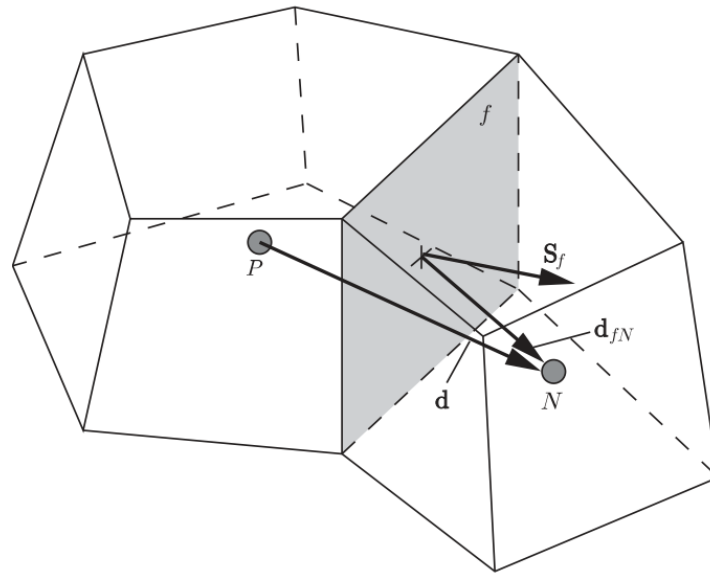
2.4.1 Método dos volumes finitos

No Método dos Volumes Finitos (FVM, na sigla em inglês), o domínio da solução é primeiro subdividido em um número finito de volumes ou células de controle contíguos, formando a malha sobre o domínio. Este método pode ser utilizado em malhas de células poliédricas com um número arbitrário de faces, cada uma com um número arbitrário de vértices (BONART, 2012) (MOUKALLED et al., 2016). Em geral, não há alinhamento da malha com o sistema de coordenadas e o número de células vizinhas pode variar de célula para célula. A única restrição geral que pode ser feita sobre conectividade de célula é que uma face de célula é interna e intercepta apenas duas células ou compreende parte de um limite externo (fronteira do domínio) e pertence apenas a uma única célula. Ao apresentar o esquema numérico, devemos dispensar notações baseadas em valores nodais como ' j ' e ' k ', em vez disso, apresentar expressões genéricas baseadas em valores interpolados de célula e face (GREENSHIELDS et al., 2008).

A cada face é atribuída uma célula proprietária e uma célula vizinha. O vetor da área da face \mathbf{S}_f é um vetor normal à superfície da face apontando para fora da célula proprietária, cuja magnitude é a da área da face, como mostrado na Figura 2.4. Neste sistema, conhecido como colocado, todas as variáveis dependentes e propriedades do material são armazenadas no centro da célula, por exemplo P na Figura 2.4. O vetor \mathbf{d} conecta o centroide da célula proprietária P ao da célula vizinha N e o vetor \mathbf{d}_{fN} conecta o centro da face ao centroide da célula N (BONART, 2012) (MOUKALLED et al., 2016).

A aplicação do método dos volumes finitos começa expressando-se as equações diferenciais que se deseja resolver dentro de uma integral sobre um volume de célula, representado por V e assumido fixo no espaço. Os termos de divergência e gradiente são então convertidos em integrais de área sobre a superfície da célula, representada por S , usando uma forma generalizada do Teorema de Gauss. Depois, as integrais são aproximadas usando fórmulas de quadratura adequadas. A integração requer fluxos nas faces das células e, como os valores das variáveis são calculados apenas no centroide de cada célula, alguma forma de interpolação deve ser usada para expressar valores das variáveis nas faces de cada célula (BONART, 2012) (MOUKALLED et al., 2016).

Figura 2.4 - Discretização de Volume Finito.



Fonte: Greenshields et al. (2008)

Para poliedros com um número arbitrário de faces, é desejável que a interpolação para uma determinada face seja apenas entre as células proprietárias e vizinhas daquela face, caso contrário, se tornam excessivamente complexos. Como resultado, obtém-se uma equação algébrica para cada volume de controle. Finalmente, a matriz resultante é resolvida diretamente ou iterativamente (BONART, 2012) (MOUKALLED et al., 2016).

2.4.1.1 Termos convectivos

Os termos convectivos são termos das equações governantes que representam o transporte de propriedades do fluido através do movimento macroscópico do próprio fluido. São representados matematicamente como $\nabla \cdot [\mathbf{u}\Psi]$, sendo que \mathbf{u} é velocidade do fluido e Ψ é um propriedade qualquer sendo transportada. Cada um destes termos é integrado sobre um volume de controle e linearizado como segue (GREENSHIELDS et al., 2008):

$$\int_V \nabla \cdot [\mathbf{u}\Psi] dV = \int_S d\mathbf{S} \cdot [\mathbf{u}\Psi] \approx \sum_f [\mathbf{S}_f \cdot \mathbf{u}_f] \Psi_f = \sum_f \phi_f \Psi_f \quad (2.28)$$

onde Σ_f significa soma sobre todas as faces e $\phi_f = \mathbf{S}_f \cdot \mathbf{u}_f$ é o fluxo volumétrico, i.e. o volume de fluido escoando através da face por segundo. O método mais simples para calcular \mathbf{u}_f e Ψ_f é: interpolação linear de \mathbf{u} das células vizinhas (diferença central); interpolação de Ψ de acordo com um dos muitos esquemas que usualmente usam algum grau de *upwinding* para estabilizar a solução. A direção de *upwind* é baseada na velocidade da escoamento e assim é caracterizada pelo sinal de ϕ_f . A interpolação linear de Ψ é calculada usando o coeficiente ponderado $w_f = |\mathbf{S}_f \cdot \mathbf{d}_{fN}| / |\mathbf{S}_f \cdot \mathbf{d}|$, de acordo com (GREENSHIELDS et al., 2008):

$$\Psi_f = \omega_f \Psi_P + (1 - \omega_f) \Psi_N$$

2.4.1.2 Termos gradientes

Os termos gradientes presentes nas equações governantes incluem ∇p na Equação 2.24. Tais termos são usualmente integrados sobre um volume de controle e discretizados como segue (GREENSHIELDS et al., 2008):

$$\int_V \nabla \Psi dV = \int_S d\mathbf{S} \Psi \approx \sum_f \mathbf{S}_f \Psi_f \quad (2.29)$$

Para escoamentos incompressíveis, Ψ_f é tipicamente calculado através de interpolação linear.

2.4.1.3 Termos laplacianos

A discretização do termo Laplaciano com coeficiente de difusão Γ para malhas poliédricas é dada como (GREENSHIELDS et al., 2008):

$$\int_V \nabla \cdot (\Gamma \nabla \Psi) dV = \int_S d\mathbf{S} \cdot (\Gamma \nabla \Psi) \approx \sum_f \Gamma_f \mathbf{S}_f \cdot (\nabla \Psi)_f \quad (2.30)$$

Usualmente, Γ_f é interpolado linearmente a partir dos valores no centro da célula poliédrica. Para o caso geral que uma face não é ortogonal (i.e. \mathbf{S}_f não é paralelo a \mathbf{d}), o cálculo de $\mathbf{S}_f \cdot (\nabla \Psi)_f$ é dividido em um componente ortogonal em termos de valores de células vizinhas e proprietárias, e um componente não ortogonal em termos de um gradiente completo, calculado no centro da célula e ele mesmo interpolado para a face: (GREENSHIELDS et al., 2008).

$$\mathbf{S}_f \cdot (\nabla \Psi)_f = \underbrace{A(\Psi_N - \Psi_P)}_{\text{ortogonal}} + \underbrace{\mathbf{a} \cdot (\nabla \Psi)_f}_{\text{não ortogonal}} \quad (2.31)$$

onde $A = |\mathbf{S}_f|^2 / (\mathbf{S}_f \cdot \mathbf{d})$ e $\mathbf{a} = \mathbf{S}_f - Ad$.

2.4.1.4 Condições de contorno

As duas principais condições de contorno existentes são: a condição de Dirichlet, que especifica um valor constante de uma variável em algum contorno da geometria; e a condição de Neumann, que especifica um gradiente constante em algum domínio (GREENSHIELDS et al., 2008).

Essas condições de contorno são aplicadas da seguinte forma. Para uma condição de Dirichlet, um valor fixo Ψ_b é especificado no contorno. A discretização do termo convectivo requer o valor de Ψ em todas as faces, neste caso Ψ_b pode ser diretamente substituído em tais faces de contorno. A discretização de um termo Laplaciano requer o gradiente normal de Ψ em cada face, que é avaliado em uma face de contorno pela diferenciação Ψ_b e Ψ_i , onde i denota a célula adjacente à face limite (GREENSHIELDS et al., 2008).

Para uma condição de Neumann, um gradiente normal fixo $(\mathbf{n} \cdot \nabla \Psi)_b$ é especificado no contorno. Este pode ser diretamente substituído na face de contorno para a discretização do termo Laplaciano. Para o termo convectivo, um valor da face de contorno deve ser calculado através da extrapolação de Ψ_i usando o gradiente normal (GREENSHIELDS et al., 2008).

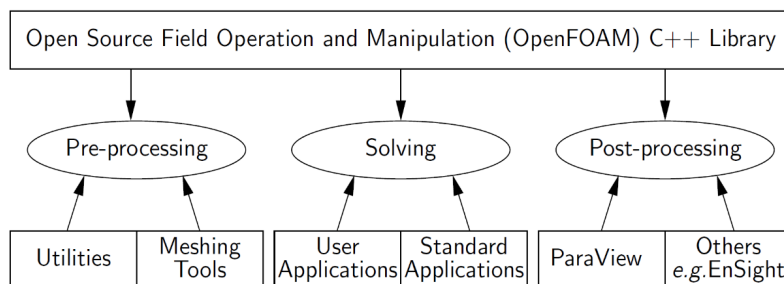
2.5 OpenFOAM - Open Field Operation and Manipulation

OpenFOAM é uma biblioteca C++ de código aberto criada para resolução de problemas envolvendo a dinâmica de fluidos computacional (CFD); tendo capacidades, porém, de resolver equações diferenciais parciais (PDE) gerais e realizar a simulação de diversos problemas multifísicos. OpenFOAM usa o método numérico dos volumes finitos (FVM, na sigla em inglês) para resolver sistemas de equações diferenciais parciais atribuídas a qualquer malha não estruturada tridimensional de células poligonais (GUIDE, 2017).

É uma biblioteca que permite facilmente a criação de executáveis, conhecidos como aplicações. As aplicações podem ser de dois tipos: códigos, que são projetados para resolverem um problema específico em mecânica contínua; e utilidades, que são projetadas para realizar tarefas que envolvem manipulação de dados, como conversão de formatos de arquivos e pré-processamento. A estrutura geral da biblioteca OpenFOAM é apresentada na Figura 2.5 (GUIDE, 2017).

Todos os códigos desenvolvidos dentro do OpenFOAM são, por padrão, tridimensionais, mas podem ser usados para problemas unidimensionais ou bidimensionais pela aplicação de condições particulares nos contornos situados no plano da direção sem interesse (GUIDE, 2017).

Figura 2.5 - Estrutura da biblioteca OpenFOAM.



Fonte: Guide (2017).

Diversos programas de pré e pós-processamento estão incluídos no ambiente OpenFOAM. Estas utilidades são, elas mesmas, escritas usando a biblioteca OpenFOAM, isto garante a consistência entre os dados manipulados (GUIDE, 2017).

Dentro da biblioteca padrão do OpenFOAM encontram-se uma vasta gama de códigos já implementados, para os mais variados problemas físicos, entre os quais (GUIDE, 2017):

- escoamentos compressíveis;
- escoamentos incompressíveis;
- escoamentos multifásicos;
- Simulação numérica direta (DNS) e simulação de grandes vórtices (LES);
- Combustão;
- Transferência de calor;
- escoamentos com partículas;
- Dinâmica molecular;
- Simulação numérica de Monte Carlo (DSMC);
- Eletrostática e Magnetostática;
- Análise de estresse linear em sólidos;
- Finanças.

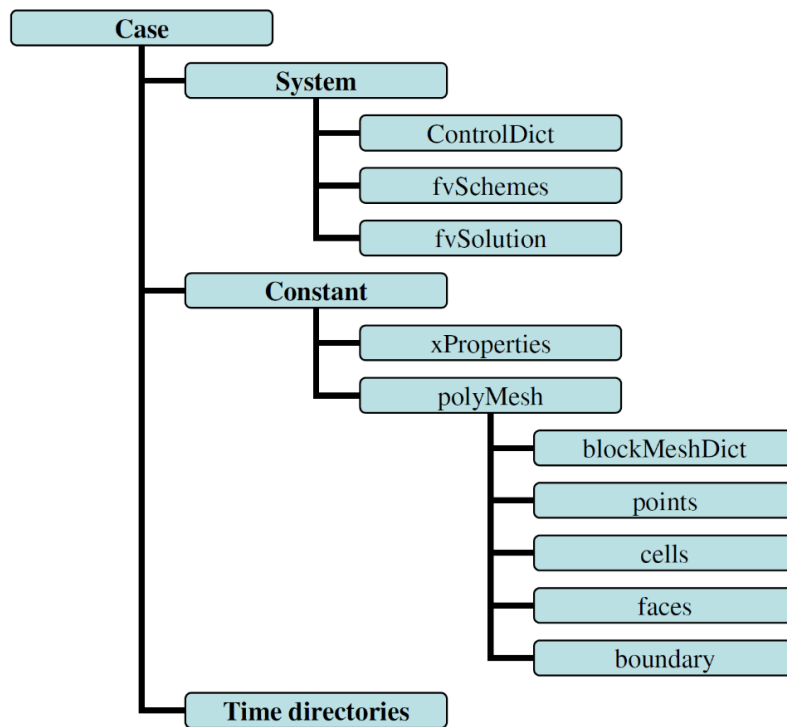
Para realizar uma simulação no OpenFOAM são necessários dois conjuntos de arquivos. O primeiro conjunto é composto pelos arquivos de configuração do caso a ser rodado, ou seja, são arquivos utilizados para o pré-processamento da simulação, tais como: descrição da geometria, condições de contorno, condições iniciais, passo de tempo, duração da simulação e etc (GUIDE, 2017) (HUANG, 2010).

O segundo conjunto de arquivos são os códigos fonte do simulador. Estes arquivos contêm os modelos matemáticos e algoritmos de solução deste modelo escritos em linguagem C++. Depois de compilados, os arquivos geram os códigos, ou seja, geram as aplicações que irão resolver o caso selecionado (GUIDE, 2017) (HUANG, 2010).

2.5.1 Estrutura de diretório da simulação

A estrutura básica de organização de um caso no OpenFOAM, com os diretórios e arquivos básicos para rodar uma simulação, são apresentados na Figura 2.6.

Figura 2.6 - Estrutura de diretório de um caso OpenFOAM.



Fonte: Huang (2010).

O diretório raiz *Case*, geralmente tem o nome do caso que se deseja rodar. Os demais diretórios devem ter sempre o mesmo nome, independente do caso (GUIDE, 2017) (HUANG, 2010).

O diretório *System* contém os arquivos de controle da simulação (*ControlDict*), controle da discretização das equações (*fvSchemes*) e controle dos solucionadores lineares (*fvSolution*). Além destes, também podem existir dentro deste diretório outros arquivos de controle como, por exemplo, controle de inicialização de variáveis (*setFieldDict*), para inicializações não uniformes das variáveis (GUIDE, 2017) (HUANG, 2010).

O diretório *Constant* é usado para descrever a geometria, através dos arquivos no subdiretório *polyMesh*, e armazenar as constantes e propriedades físicas usadas na simulação, dentro dos arquivos cujos nomes terminam em *Properties* (GUIDE, 2017) (HUANG, 2010).

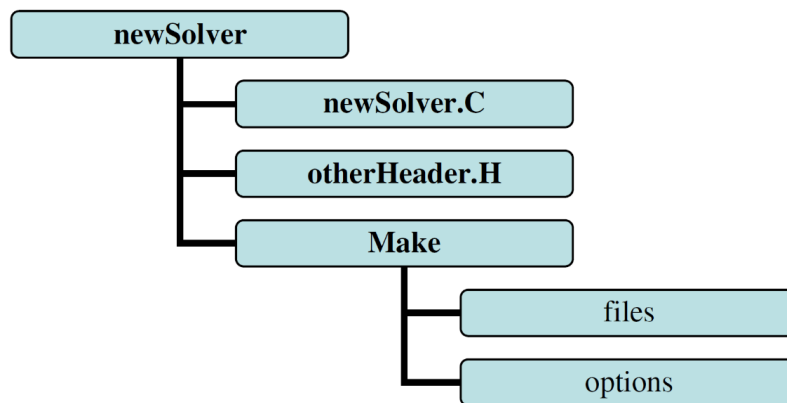
Os diretórios *Time* são criados conforme a simulação avança no tempo, sendo que cada diretório recebe o nome do tempo de simulação em que foi criado. Dentro desses diretórios são armazenados arquivos que correspondem aos valores das variáveis de campo calculadas naquele instante. O único diretório de tempo que deve existir antes da simulação começar é o diretório com nome θ , responsável por armazenar as condições iniciais das variáveis (GUIDE, 2017) (HUANG, 2010).

2.5.2 Estrutura de diretórios do código de simulação

Além de ser usado como um pacote de simulação comum, OpenFOAM também permite a criação de novos códigos para equações diferenciais parciais. A Figura 2.7 ilustra a organização típica de um diretório para um novo código (GREENSHIELDS, 2015).

O nome de diretório *newSolver* representa o nome do novo código implementado, ou um que já exista por padrão no OpenFOAM. Dentro deste, temos o arquivo fonte principal *newSolver.C*, é neste arquivo que são implementadas as equações e algoritmos a serem resolvidos (GREENSHIELDS, 2015).

Figura 2.7 - Estrutura de um código OpenFOAM.



Fonte: Huang (2010).

O arquivo *otherHeader.H* é usado para armazenar estruturas de código auxiliares ao código principal, por exemplo, para declarações de variáveis. É importante notar que o arquivo *otherHeader.H* é incluído no arquivo *newSolver.C* no momento da compilação do código, ou seja, os códigos são combinados em um só para criação da nova aplicação (GREENSHIELDS, 2015).

O subdiretório *Make* armazena os arquivos de configuração de compilação *files* e *options*. O arquivo *files* armazena o nome do código que será criado. No arquivo *options* são armazenados os nomes de outros arquivos com extensão *.H* para serem incluídos na compilação, sendo estes arquivos da biblioteca OpenFOAM (GREENSHIELDS, 2015).

2.5.3 Vantagens

Entre as principais vantagens da biblioteca OpenFOAM, podemos citar (JASAK et al., 2007) (GREENSHIELDS, 2015):

- Está sob desenvolvimento ativo;
- Tem capacidades comparáveis a softwares pagos;
- Comunidade extensa dentro da indústria, academia e laboratórios de pesquisa;
- Ampla documentação;
- Código totalmente aberto e gratuito;
- É desenvolvido em camadas isoladas;
- Capacidades de paralelização já implementadas e de simples utilização;
- Faz checagem dimensional nas equações a serem resolvidas;
- Fácil implementação de novas equações.

Uma das características que mais chamam a atenção na biblioteca OpenFOAM, é a forma como equações são introduzidas no código fonte. Ao contrário da maioria dos códigos abertos disponíveis, que exigem que o programador discretize as equações manualmente antes de resolvê-las, o código de um código OpenFOAM aceita a linguagem natural das equações. Por exemplo, a Equação 2.32 representa a energia cinética turbulenta, esta equação é escrita no código C++ do OpenFOAM conforme mostrado no trecho de código da Figura 2.8 (JASAK et al., 2007) (GREENSHIELDS, 2015).

$$\frac{\partial y}{\partial t} + \nabla \cdot (uk) - \nabla \cdot [(\nu + \nu_t) \nabla k] = \nu_t \left[\frac{1}{2} (\nabla u + \nabla u^T) \right]^2 - \frac{\epsilon_0}{k_0} k \quad (2.32)$$

Figura 2.8 - Equação no código C++ da biblioteca OpenFOAM.

```
solve
(
  fvm::ddt(k)
  + fvm::div(phi, k)
  - fvm::laplacian(nu() + nut, k)
  == nut*magSqr(symm(fvc::grad(U)))
  - fvm::Sp(epsilon/k, k)
);
```

Fonte: [Jasak et al. \(2007\)](#).

Essa facilidade em implementar novas equações diminui muito a chance de erros durante a programação, além de permitir a rápida modificação do código.

Diversos trabalhos envolvendo eletromagnetismo e simulação MHD já foram desenvolvidos utilizando-se a biblioteca OpenFOAM. [Huang \(2010\)](#) implementou duas formulações diferentes (A-V e A-J) para cálculo de campo magnetostático em materiais como cobre, aço linear e ímãs permanentes. Além disso, realizou cálculos de força magnética utilizando o Método de Força de Lorentz e o Método do Tensor de Estresse de Maxwell.

[Roghair et al. \(2013\)](#) implementou um modelo eletro-hidrodinâmico para simulação de interface fluido-fluido sob a presença de um campo elétrico. A Lei de Gauss e a equação de transporte de carga foram incorporadas no modelo de Volume de Fluido (VOF), já existente por padrão na distribuição OpenFOAM. A interface entre dois fluidos dielétricos e a interface entre um fluido condutor e outro dielétrico foram simuladas, com boa concordância para com as relações teóricas.

[Ferroni \(2012\)](#) utilizou a biblioteca OpenFOAM para desenvolver um código de escoamento MHD. O programa desenvolvido se mostrou eficaz em simular escoamento de metais líquidos, usados na refrigeração de reatores de fusão nuclear. Um software pago com capacidade de cálculo MHD, foi testado e falhou nesta simulação, devido as condições impostas ao escoamento.

[Sass-tisovskaya \(2009\)](#) desenvolveu uma aplicação baseada em OpenFOAM para simulação de solda com arco de plasma. A parte eletromagnética foi testada separadamente para validação. A ferramenta de simulação completa foi avaliada usando um problema de soldagem TIG.

2.5.4 *RhoCentralFoam*

O *rhoCentralFoam* é um código compressível baseado em densidade para fluxos viscosos de alta velocidade, conta com os esquemas *central-upwind* de Kurganov e Tadmor (GUIDE, 2017), não escalonados, criado por Greenshields et al. (2008). Uma breve descrição do algoritmo de solução do código é apresentada na seção 3.3. Para mais detalhes sobre como o código funciona, o leitor pode consultar Greenshields et al. (2008).

Nos fluidos compressíveis, as propriedades não são apenas transportadas pelo fluxo, mas também pela propagação de ondas. Isso requer a construção de interpolações de fluxo que levam em conta que o transporte das quantidades físicas consideradas podem ocorrer em qualquer direção, ou seja, a solução das equações deve ser estabilizada levando-se em consideração não apenas o transporte de propriedades através de difusão e convecção, mas também por meio de ondas de choque (GREENSHIELDS et al., 2008).

Como as interpolações são de valores de células vizinhas para uma dada face, os métodos de Kurganov e Tadmor (2000) (KT) e Kurganov et al. (2001) (KNP) podem ser aplicados (GREENSHIELDS et al., 2008).

No código *rhoCentralFoam* a preocupação principal é com a precisão espacial dos esquemas de discretização sob investigação. As derivadas de tempo são discretizadas por um esquema implícito simples de Euler, em vez de métodos mais elaborados; não há, no entanto, dificuldade em incorporar tais métodos para integração temporal (GREENSHIELDS et al., 2008).

2.5.4.1 Tratamento dos termos convectivos

Os termos convectivos das equações de conservação listadas na Seção 2.3 e aplicáveis ao código *rhoCentralFoam*, são da forma $\nabla \cdot [\mathbf{u}\rho]$, $\nabla \cdot [\mathbf{u}(\rho\mathbf{u})]$, $\nabla \cdot [\mathbf{u}(\rho E)]$ e $\nabla \cdot [\mathbf{u}p]$. A forma como eles são tratados é muito importante para esquemas do tipo central (GREENSHIELDS et al., 2008).

A interpolação para uma dada face é feita apenas com valores de células vizinhas, logo, pode-se aplicar os métodos KT e KNP em sua forma original para sistemas multidimensionais usando a chamada reconstrução “dimensão por dimensão” (KURGANOV et al., 2001) (KURGANOV; TADMOR, 2000) (também conhecida como reconstrução “face a face” em uma estrutura poliédrica), ao invés de algumas formas genuinamente multidimensionais de reconstrução (GREENSHIELDS et al., 2008).

O procedimento de interpolação é dividido em duas direções, representando o escoamento para dentro e fora da célula proprietária da face em questão. Essas direções são denotadas $f+$, coincidindo com a direção $+\mathbf{S}_f$, e $f-$, coincidindo com $-\mathbf{S}_f$. A discretização é feita da seguinte maneira (GREENSHIELDS et al., 2008):

$$\sum_f \phi_f \Psi_f = \sum_f [\alpha \phi_{f+} \Psi_{f+} + (1 - \alpha) \phi_{f-} \Psi_{f-} + \omega_f (\Psi_{f-} - \Psi_{f+})] \quad (2.33)$$

Os primeiros dois termos do lado direito da Equação 2.33 são os cálculos de fluxo nas direções $f+$ e $f-$, respectivamente. O terceiro termo é somente requerido quando o termo de convecção é parte da derivada substancial, e.g. $\nabla \cdot [\mathbf{u}(\rho\mathbf{u})]$ na Equação 2.24, para o qual $\partial(\rho\mathbf{u})/\partial t$ completa a derivada substantiva de $\rho\mathbf{u}$. É um termo de difusão adicional usando o fluxo volumétrico w_f baseado na velocidade máximas de propagação de qualquer descontinuidade que possa existir em uma face entre valores interpolados nas direções $f+$ e $f-$ (GREENSHIELDS et al., 2008).

No método KT, as contribuições de $f+$ e $f-$ são ponderadas igualmente, ou seja, o coeficiente é $\alpha = 0,5$, assim sua descrição como um esquema central. O método KNP calcula α baseado nas velocidade locais unilaterais de propagação. A ponderação é então enviesada na direção *upwind*, assim esses esquemas são denominados *upwind* centrais. Fluxos volumétricos associados com as velocidade locais de propagação podem ser calculados como segue, notando que eles são ambos definidos aqui como positivos nas suas respectivas direções $f+$ e $f-$ (GREENSHIELDS et al., 2008):

$$\psi_{f+} = \max(c_{f+} |\mathbf{S}_f| + \phi_{f+}, c_{f-} |\mathbf{S}_f| + \phi_{f-}, 0) \quad (2.34)$$

$$\psi_{f-} = \max(c_{f+} |\mathbf{S}_f| - \phi_{f+}, c_{f-} |\mathbf{S}_f| - \phi_{f-}, 0) \quad (2.35)$$

Aqui $c_f = \sqrt{\gamma RT_{f\pm}}$ são as velocidades do som na face, para dentro e para fora da célula proprietária. O fator de ponderação é (GREENSHIELDS et al., 2008):

$$\alpha = \begin{cases} \frac{1}{2} & \text{para o método KT} \\ \frac{\psi_{f+}}{\psi_{f+} + \psi_{f-}} & \text{para o método KNP} \end{cases} \quad (2.36)$$

O fluxo volumétrico difusivo é calculado de acordo com (GREENSHIELDS et al., 2008):

$$\omega_f = \begin{cases} \alpha \max(\Psi_{f+}, \Psi_{f-}) & \text{para o método KT} \\ \alpha(1 - \alpha)(\Psi_{f+} + \Psi_{f-}) & \text{para o método KNP} \end{cases} \quad (2.37)$$

O método envolve as interpolações de face $f+$ e $f-$ de um certo número de variáveis (T , ρ , etc.) a partir de valores nos centros de células vizinhas. O procedimento de interpolação usa um limitador para chavear entre os esquemas de baixo e alta-ordem baseados na função de limitação de fluxo $\beta(r)$, onde r representa a razão de sucessivos gradientes da variável interpolada, restringido para $r \geq 0$. Em uma malha poliédrica, r pode ser descrito como segue para a direção $f+$ (GREENSHIELDS et al., 2008):

$$r = 2 \frac{\mathbf{d} \cdot (\nabla \Psi)_P}{(\nabla_{\mathbf{d}} \Psi)_f} - 1 \quad (2.38)$$

onde $(\nabla \Psi)_P$ é o gradiente completo calculado na célula proprietária P como descrito na Seção 2.4.1.2 com interpolação linear e $(\nabla_{\mathbf{d}} \Psi)_f = \Psi_N - \Psi_P$ pe o componente do gradiente normal a face, escalado por $|\mathbf{d}|$.

As interpolações $f+$ e $f-$ são baseadas na limitação das interpolações *upwind* de primeira ordem e linear de segunda ordem padrões. Os limitantes escolhidos são de diminuição de variação total (TVD, na sigla em inglês) e simétricos, para os quais $\beta(r)/r = \beta(1/r)$, conhecidos como Minmod [28] e van Leer [29], cuja as funções limitantes são $\beta(r) = \max[0, \min(1, r)]$ e $\beta(r) = (r + |r|)/(1 + r)$, respectivamente. Então, a interpolação $f+$ de Ψ , por exemplo, é simplesmente calculado como (GREENSHIELDS et al., 2008):

$$\Psi_{f+} = (1 - g_{f+})\Psi_P + g_{f+}\Psi_N \quad (2.39)$$

onde $g_{f+} = \beta(1 - \omega_f)$. É evidente que $\beta = 0$ leva a interpolação *upwind* e $\beta = 1$ leva a interpolação linear; Outro ponto a se notar é que $0 \leq \beta \leq 2$, tal que $\beta = 2$ corresponde a interpolação *downwind* (GREENSHIELDS et al., 2008).

2.5.4.2 Tratamento dos termos gradientes

Os termos gradientes foram descritos na Seção 2.4.1.2, os esquemas KT e KNP, contudo, dividem o procedimento de interpolação nas direções $f+$ e $f-$ de acordo

com (GREENSHIELDS et al., 2008):

$$\sum_f \mathbf{S}_f \Psi_f = \sum_f [\alpha \mathbf{S}_f \Psi_{f+} + (1 - \alpha) \mathbf{S}_f \Psi_{f-}] \quad (2.40)$$

as interpolações de $f+$ e $f-$ usam o limitador descrito anteriormente.

2.5.4.3 Tratamento dos termos laplacianos

Os termos Laplacianos não sofrem alteração devido aos esquemas KN e KNP, sendo discretizados como descrito na Seção 2.4.1.3.

2.6 Desenvolvimento do Simulador

Sabendo-se da falta de uma ferramenta de simulação aberta, de fácil acesso e adequada para a simulação de PPT, propõe-se o desenvolvimento de um código, baseado na biblioteca OpenFOAM, que permita a simulação de aspectos básicos de um propulsor PPT retangular sem vela de ignição.

Várias características indicam que a biblioteca OpenFOAM é adequada para o desenvolvimento do código, entre as quais (GUIDE, 2017):

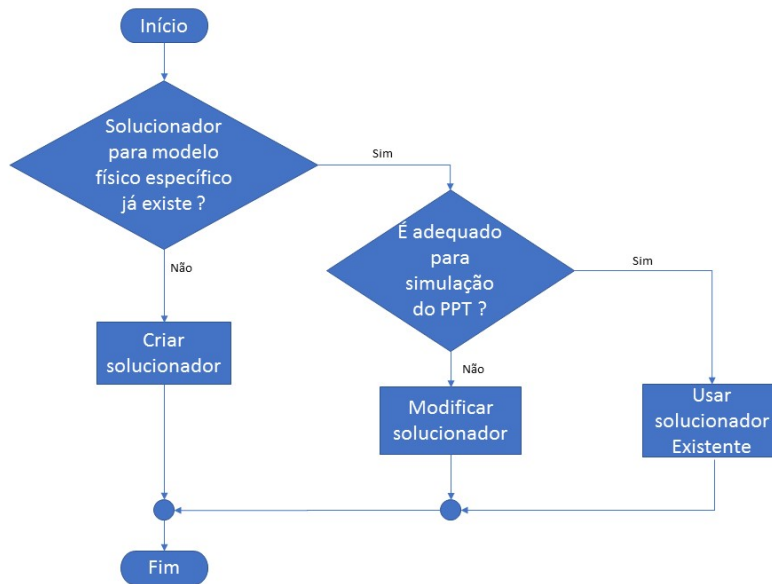
- Biblioteca aberta, gratuita e robusta;
- Escrita em C++;
- Documentação ampla sobre a biblioteca;

Não será simulada a vela de ignição devido à complexidade de funcionamento da mesma, além do fato de ter pouco influência no desempenho do propulsor em si. Além disso, a geometria retangular foi escolhida devido a simplicidade de análise da mesma.

Devido ao princípio de funcionamento do PPT, pode-se destacar quatro processos básicos que ocorrem no mesmo: a propagação de corrente elétrica, na descarga elétrica; a geração de campo magnético, devido a corrente elétrica circulante; a transferência de calor, entre o a descarga elétrica e o propelente; e escoamento do fluido sob a ação de forças de corpo, devido ao gradiente de pressão e a Força de Lorentz.

Como já existem diversos códigos disponíveis no OpenFOAM, o desenvolvimento de cada código se dará conforme fluxograma exposto na Figura 2.9.

Figura 2.9 - Passos para o desenvolvimento de um código.



Fonte: Próprio Autor.

Cada um dos processos físicos acontecendo no PPT será resolvido separadamente, através de um código próprio, com exceção da geração do campo magnético. Depois que cada um dos processos for simulado separadamente, haverá a integração de cada código correspondente ao simulador completo do PPT. Este método de desenvolvimento modular permite maior segurança e agilidade na criação do simulador.

Uma vez que a simulação completa esteja rodando adequadamente, será realizada a alteração da energia do capacitor de descarga. O código completo será validado através comparação dos resultados numéricos obtidos com dados experimentais, disponíveis na literatura, de PPTs retangulares.

3 METODOLOGIA

Para o desenvolvimento do simulador completo do PPT, os processos físicos acontecendo no mesmo foram divididos em três domínios principais:

- A descarga elétrica que aquece e sublima o PTFE;
- A propagação de calor através da barra de PTFE;
- O escoamento do fluido dentro da cavidade do PPT.

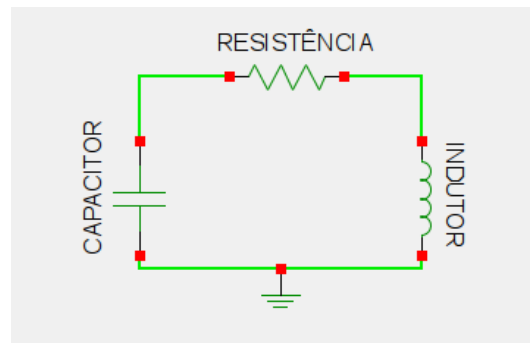
Para cada um destes domínios foi desenvolvido e/ou adaptado um código no OpenFOAM. Para que os códigos de cada processo físico em questão fossem integrados, condições de contorno especiais foram desenvolvidas, a fim de permitir a interação entre os diferentes domínios.

A versão do OpenFOAM utilizada foi a 4.1 e a elaboração de cada um dos códigos é apresentada nas seções seguintes.

3.1 Código *RLC_Foam*

Como um passo inicial para simular a descarga elétrica sobre a superfície do PTFE, foi criado o código numérico *RLC_Foam*. Este código simula um circuito série composto por um resistência, um indutor e um capacitor. O esquema elétrico do circuito é apresentado na Figura 3.1.

Figura 3.1 - Circuito simulado pelo código *RLC_Foam*.



Fonte: Próprio Autor.

Este circuito é descrito por uma equação diferencial ordinária de segunda ordem, dada pela Equação 3.1.

$$L \frac{\partial^2 Q}{\partial t^2} + R \frac{\partial Q}{\partial t} + \frac{Q}{C} = 0 \quad (3.1)$$

3.1.1 Malha e Condições de contorno

Como o código resolve apenas uma equação diferencial ordinária dependente do tempo, a malha usada é composta por um única célula, com condições de contorno *empty* em todas as faces. Esta é uma exigência do OpenFOAM, pois o mesmo utiliza o Método dos Volumes Finitos (FVM, na sigla em inglês) para resolução de equações e, assim sendo, há necessidade de criação de um volume com o qual as variáveis estejam associadas.

3.1.2 Condições iniciais

A variável para o qual se resolve é a carga Q do capacitor. Portanto, as condições iniciais utilizadas no problema são a carga inicial do capacitor $Q(t = 0) = 2,720294 \text{ mC}$ e a derivada da carga, ou seja, a corrente $I(t = 0) = Q'(t = 0) = 0$.

Este valor de carga inicial foi calculado utilizando-se a Equação 3.2, que representa a energia armazenada no capacitor em função da carga, para que o mesmo armazene uma energia de 1,85 J.

$$E = \frac{Q^2}{2C} \quad (3.2)$$

3.1.3 Propriedades físicas

Os parâmetros utilizados para a simulação do circuito RLC são dados na Tabela 3.1. Estes valores podem ser alterados através das entradas apropriadas no dicionário de configuração *transportProperties*.

Tabela 3.1 - Constantes físicas usadas no código *RLC_Foam*.

Indutância L	34 nH
Resistência R	30 mΩ
Capacitância C	2 μF

Fonte: Turchi e Mikellides (1995)

3.2 Código *HeatPropagation*

Para simulação da propagação do calor na barra de PTFE do propulsor, o código *HeatPropagation* foi desenvolvido. A fonte de calor que aquece o PTFE é proveniente do efeito Joule da descarga elétrica sobre a superfície do PTFE.

Para modelar este fenômeno a Equação 3.3, conhecida como equação do calor, é usada. Onde α é difusividade térmica.

$$\frac{\partial T}{\partial t} - \alpha \nabla^2 T = 0 \quad (3.3)$$

3.2.1 Malha

Como o processo físico que aquece o PTFE dura apenas alguns micro-segundos, a porção da barra de PTFE que é aquecida se torna ínfima. Sendo assim, para possibilitar a visualização da variação de temperatura do PTFE, foi usada uma malha extremamente fina na direção do eixo X, com um grande número de células nesta direção. A Tabela 3.2 resume as informações geométricas desta malha.

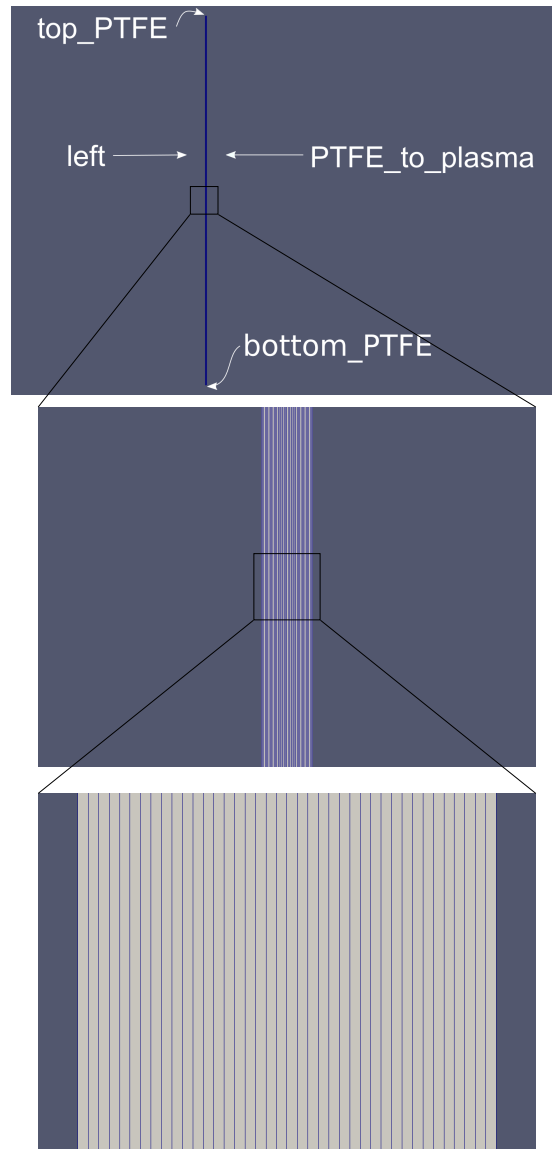
Tabela 3.2 - Geometria da malha para o código *HeatPropagation*.

Dimensão	Tamanho (cm)	Número de células	Tamanho da célula (cm)
Comprimento (eixo X)	0,01	40	0,00025
Altura (eixo Y)	3	15	0,2
Largura (eixo Z)	1	1	1

Fonte: Próprio Autor

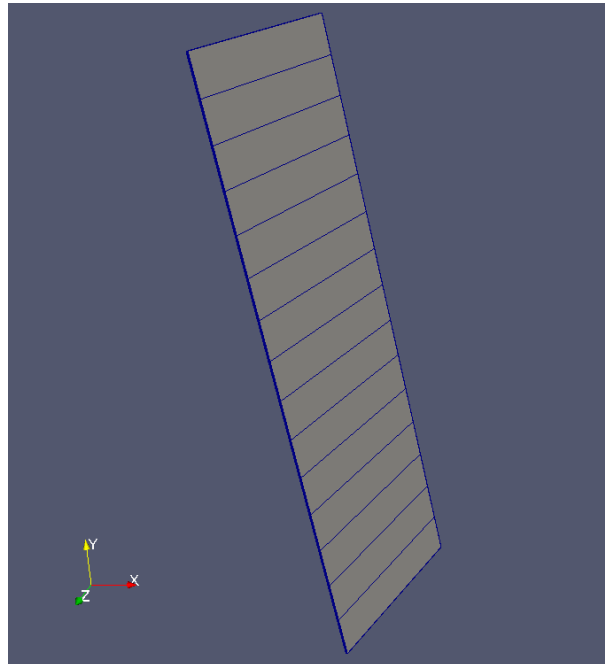
A Figura 3.2 mostra a malha usada para simular a barra de PTFE. Nesta figura também são mostrados os nomes das fronteiras usadas. Para possibilitar a visualização das células que compõem a malha de simulação do PTFE são mostrados dois níveis de zoom sobre a malha.

Figura 3.2 - Detalhe da malha usada para simulação do aquecimento no PTFE.



Fonte: Próprio Autor.

Figura 3.3 - Malha para simulação do aquecimento na barra de PTFE vista em 3D.



Fonte: Próprio Autor.

A Figura 3.3 permite visualizar a malha em 3D. O acoplamento entre a malha do fluido e a malha do sólido exige que o mesmo número de células estejam presentes na fronteira de interligação das malhas. Ou seja, o número de células na direção do eixo Y (vertical) da malha do PTFE, deve ser igual ao número de células na direção do eixo Y da malha do fluido, retratada na Figura 3.6 da Página 58.

3.2.2 Condições de contorno

As condições de contorno para a barra de PTFE são apresentadas na Tabela 3.3. É assumido que todas as fronteiras do PTFE são adiabáticas, menos a fronteira *PTFE_to_plasma*. Neste contorno específico, o domínio de simulação do PTFE (sólido) é interligado ao domínio de simulação do plasma (fluido), sendo necessária uma condição de contorno que interligue estes domínios e suas respectivas equações. Esta condição de contorno especial foi desenvolvida e nomeada *Ablation*, sendo descrita mais adiante neste capítulo.

Tabela 3.3 - Condições de contorno para a Temperatura T (K) na malha do PTFE.

Contorno	T
top_PTFE	zeroGradient
left	zeroGradient
bottom_PTFE	zeroGradient
PTFE_to_plasma	Ablation
frontAndBack	empty

Fonte: Próprio Autor

3.2.3 Condição inicial

A condição de temperatura ambiente foi aplicada a toda a barra de PTFE, do mesmo modo que fora aplicada ao fluido no código *rhoCentralFoam*.

Tabela 3.4 - Condição inicial para a temperatura na malha do PTFE.

Campo	Valor Inicial
Temperatura (T)	uniform 300

Fonte: Próprio Autor

3.2.4 Propriedades Físicas

Para que a simulação de propagação de calor possa ser realizada, além das condições iniciais e de contorno, são necessárias as propriedades termodinâmicas do PTFE. A Tabela 3.5 sumariza os dados relevantes para simulação.

Tabela 3.5 - Propriedades do PTFE sólido.

Propriedade	Valor Inicial
α	$1,28e-7 \text{ m}^2/s$
c_p	1250 J/Kg/K
ρ	2150 Kg/m^3

Fonte: Adaptado de Mikellides (1999).

3.3 Código *rhoCentralFoam*

O código *rhoCentralFoam* foi desenvolvido para resolver problemas de escoamentos compressíveis e viscosos. Ele é baseado em esquemas semi-discretos, não-escalonados (BONART, 2012). Este código padrão da biblioteca OpenFOAM foi escolhido como base do simulador por sua capacidade de resolver com precisão possíveis ondas de choque, que podem aparecer em escoamentos trans-sônicos.

Este código resolve cada uma das equações governantes do escoamento separadamente. O primeiro passo de resolução é o cálculo da densidade a partir da equação de continuidade, usando a velocidade calculada no passo de tempo anterior. A Equação 2.23, apresentada novamente aqui, é usada neste passo.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot [\rho \mathbf{u}] = 0 \quad (3.4)$$

A seguir, a equação do momentum é resolvida em dois passos distintos. O motivo desta resolução em duas partes é a preocupação de se evitar que o procedimento de solução seja completamente explícito, ou seja, que todos os novos valores no passo de tempo atual sejam calculados a partir das condições de difusão, convecção e contorno no passo de tempo anterior. Sabe-se que esta abordagem fornece um baixo custo computacional por passo de tempo, contudo, ela limita severamente o tamanho deste passo, que deve ficar cada vez menor conforme o termo de difusão domina a equação.

Primeiramente, a densidade de momentum invíscida ($\hat{\mathbf{u}}$) é calculada. Na Equação 3.5, o primeiro termo do lado esquerdo é a derivada de tempo, calculada levando-se em consideração somente os fluxos não-viscosos.

$$\left(\frac{\partial \hat{\mathbf{u}}}{\partial t} \right)_I + \nabla \cdot (\mathbf{u} \hat{\mathbf{u}}) + \nabla p = 0 \quad (3.5)$$

A relação entre densidade de momentum e velocidade é dada por $\hat{\mathbf{u}} = \rho \mathbf{u}$, com esta relação pode-se achar um novo valor para a velocidade com os valores calculados de $\hat{\mathbf{u}}$ e ρ . As forças viscosas são levadas em consideração no cálculo do momentum através da equação de correção difusiva para \mathbf{u} .

$$\left(\frac{\partial(\rho\mathbf{u})}{\partial t}\right)_V - \nabla \cdot (\mu\nabla\mathbf{u}) - \nabla \cdot \mathbf{T}_{exp} = 0 \quad (3.6)$$

Onde os termos do tensor de estresse viscoso contendo o acoplamento inter-componente são tratados explicitamente, $\mathbf{T}_{exp} = \mu[(\nabla\mathbf{u})^T - \frac{2}{3}tr(\nabla\mathbf{u})\mathbf{I}]$.

A solução da equação de energia procede de forma similar. Primeiro, a densidade de energia total ($\hat{E} = \rho E$) é resolvida através da Equação 3.7. O fluxo de calor difusivo é ignorado neste passo.

$$\left(\frac{\partial\hat{E}}{\partial t}\right)_I + \nabla \cdot (\mathbf{u}(\hat{E} + p)) + \nabla \cdot (\mathbf{T} \cdot \mathbf{u}) = 0 \quad (3.7)$$

Uma vez achado o valor de \hat{E} , a temperatura é calculada através da Equação 3.8, usando os valores já calculados de \hat{E} , ρ e \mathbf{u} . c_v é o calor específico a volume constante.

$$T = \frac{1}{c_v} \left(\frac{\hat{E}}{\rho} - \frac{|\mathbf{u}|^2}{2} \right) \quad (3.8)$$

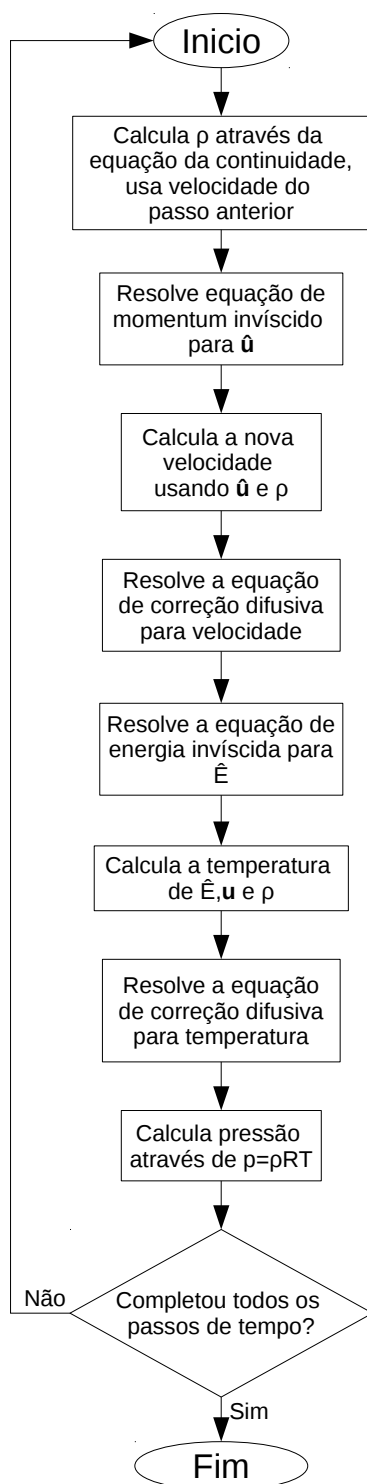
Finalmente, a equação de correção de difusão para T , Equação 3.9, é resolvida, incluindo-se o fluxo de calor difusivo.

$$\left(\frac{\partial(\rho c_v T)}{\partial t}\right)_V - \nabla \cdot (k\nabla T) = 0 \quad (3.9)$$

A viscosidade (μ) e a condutividade térmica (k) são funções da temperatura e, portanto, são atualizadas dentro de cada iteração do código, permanecendo constante até a próxima iteração. A pressão (p), é calculada no final de cada iteração. O Fluxograma 3.4 ilustra os passo do algoritmo de solução deste código.

No artigo de Greenshields et al. (2008), o código *rhoCentralFoam* foi validado usando-se dados experimentais e soluções analíticas para quatro casos de teste, entre eles: *The supersonic jet problem* de Ladenburg e *Hypersonic flow over a 25°-55° biconic* de Holden e Wadhams . A acurácia observada do código é boa e permite fazer predições de escoamento satisfatórias.

Figura 3.4 - Fluxograma de execução do código *rhoCentralFoam*.



Fonte: Adaptado de Bonart (2012).

3.3.1 Malha

A malha usada para simulação do escoamento é baseada na geometria do PPT retangular LES-6. Este modelo de PPT tem dimensões de comprimento, altura e profundidade iguais a 6cm, 3cm e 1cm, respectivamente.

A geometria retangular foi escolhida para a simulação devido à sua simplicidade e facilidade de criação. A malha apresentada no Figura 3.5 foi gerada com o programa utilitário *blockMesh*, presente dentro da biblioteca OpenFOAM. O arquivo de configuração deste utilitário encontra-se no Apêndice A.

Devido a restrições de poder computacional disponível para as simulações, simplificações na malha foram feitas para permitir o desenvolvimento e teste do código em tempo hábil. Entre estas simplificações pode-se citar as duas principais: alteração do domínio de simulação e definição do tamanho das células.

O domínio de simulação inicialmente pensado era igual a geometria do PPT LES-6, com as dimensões já supracitadas mas, devido ao peso da computação, estas dimensões foram alteradas para o comprimento, altura e largura apresentadas na Tabela 3.6.

O tamanho das células foi selecionado de modo a permitir a resolução de eventuais detalhes, presentes na dinâmica do escoamento, e a velocidade de computação do código de simulação. As dimensões de células são apresentadas na Tabela 3.6.

Cada uma das fronteiras do domínio de simulação recebe um nome para aplicação das condições de contorno. Na Figuras 3.5 e 3.6 as nomes de cada uma das fronteiras do problema são apresentados.

Tabela 3.6 - Geometria da malha para o código *rhoCentralFoam*.

Dimensão	Tamanho (cm)	Número de células	Tamanho da célula (cm)
Comprimento (eixo X)	1,99	50	0,0398
Altura (eixo Y)	3	15	0,2
Largura (eixo Z)	1	1	1

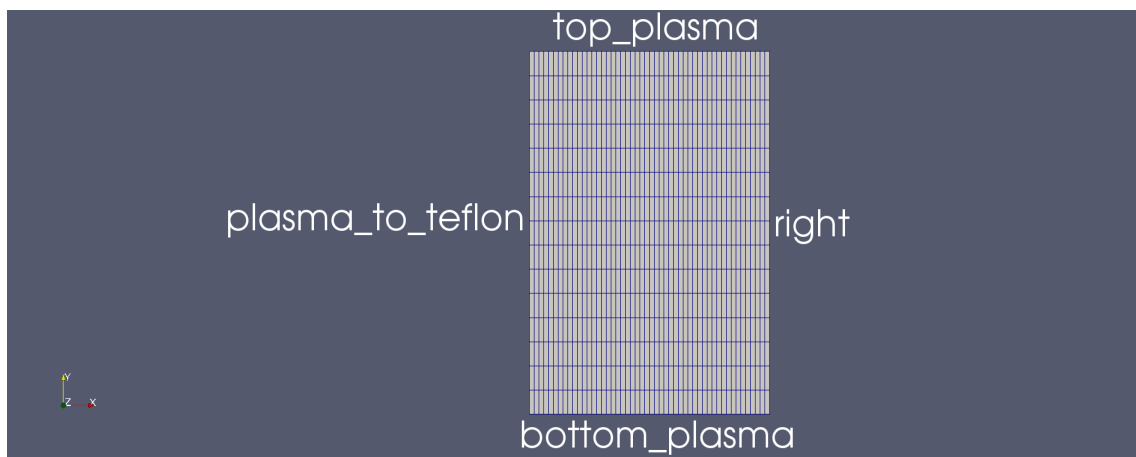
Fonte: Próprio Autor

Na Figura 3.5 é apresentada a malha de simulação para o código *rhoCentralFoam*,

vista de perfil. O escoamento acontece da esquerda para a direita, no sentido positivo do eixo X.

No problema sob estudo, a convecção é o meio de transporte das propriedades do fluido mais dominante. Logo, espera-se que as maiores variações em tais propriedades (gradientes), ocorram na direção do vetor velocidade do fluido. Sabendo-se disto, a malha de simulação foi pensada para ter maior resolução espacial na direção do escoamento (eixo X), com 50 células nesta direção. Em contraste, a direção transversal ao escoamento (eixo Y) tem apenas 15 células.

Figura 3.5 - Malha para simulação do escoamento, vista no plano YX.

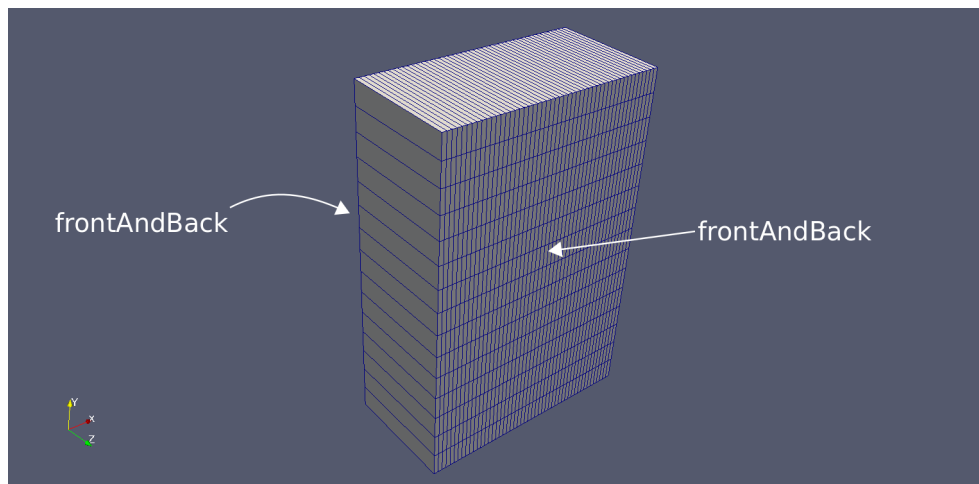


Fonte: Próprio Autor.

Para diminuição do tempo computacional, a simulação do escoamento é considerada um problema bidimensional. Contudo, a biblioteca OpenFOAM exige a definição de uma malha tridimensional para todos os problemas simulados. A aplicação de condições de contorno especiais, nas fronteiras apropriadas, permite desconsiderar um ou mais eixos nos cálculos da simulação, transformando-a, de fato, em uma simulação bi-, uni- ou zero-dimensional (GUIDE, 2017).

Para melhor compreensão da geometria, a Figura 3.6 mostra a malha em vista 3D.

Figura 3.6 - Malha para simulação do escoamento vista em 3D.



Fonte: Próprio Autor.

3.3.2 Condições de contorno

Cada variável independente das equações governantes do escoamento, deve-se estabelecer as condições de contorno apropriadas em cada fronteira da geometria em questão. As condições de contorno que são aplicadas às fronteiras do domínio de simulação (malha da geometria) são apresentadas nas Tabelas 3.7-3.9. Para todas as variáveis, a fronteira *frontAndBack* é definida como *empty*. Uma simulação em 2D exige esta condição, como citado anteriormente na Seção 3.1.1.

Para a velocidade \mathbf{U} , a condição *slip* foi escolhida para as fronteiras *top_plasma* e *bottom_plasma*. Esta condição de contorno permite que a velocidade tangencial à parede do propulsor seja diferente de zero, ou seja, permite que haja escorregamento do fluido sobre a parede. Isto é adequado para simulações em baixa pressão, como é o caso deste estudo. Para a fronteira *plasma_to_PTFE* a velocidade é fixada em $\vec{0}$ (grandeza vetorial) uniforme em todo este contorno, já que o escoamento parte de velocidade zero.

Tabela 3.7 - Condições de contorno para velocidade U (m/s).

Contorno	U
top_plasma	slip
right	zeroGradient
bottom_plasma	slip
plasma_to_PTFE	fixedValue uniform (0 0 0)
frontAndBack	empty

Fonte: Próprio Autor

Para a pressão \mathbf{p} , a condição de *zeroGradient* foi para todas as fronteiras. Isto permite que a pressão varie livremente em todos os pontos do domínio, sem que haja qualquer fronteira que atue como uma fixação de pressão.

Tabela 3.8 - Condições de contorno para pressão p (Pa).

Contorno	p
top_plasma	zeroGradient
right	zeroGradient
bottom_plasma	zeroGradient
plasma_to_PTFE	zeroGradient
frontAndBack	empty

Fonte: Próprio Autor

Para a temperatura T , a condição *zeroGradient* foi aplicada as fronteiras *top_plasma*, *bottom_plasma* e *right*. Esta condição representa a imposição de parede adiabática para as fronteiras do propulsor. A fronteira *fixedValue* tem o valor fixo de 300K, contudo, uma condição de contorno especial foi criada e imposta a esta fronteira, de modo que esta configuração de fronteira é irrelevante. A condição de contorno desenvolvida é baseada na condição de Neumann, mas com chaveamento do valor do gradiente de acordo com a temperatura da superfície.

Tabela 3.9 - Condições de contorno para temperatura T (K).

Contorno	T
top_plasma	zeroGradient
right	zeroGradient
bottom_plasma	zeroGradient
plasma_to_PTFE	fixedValue uniform 300
frontAndBack	empty

Fonte: Próprio Autor

3.3.3 Condições iniciais

Para que a simulação possa ser computada é necessário que se estabeleçam as condições iniciais de cada uma das variáveis do problemas. Na tabela 3.10 são listadas as condições iniciais impostas.

No instante inicial considera-se que a cavidade do propulsor está preenchida com um gás em repouso, com massa molar igual ao do monômero de PTFE, ou seja, 100 g/mol. A temperatura é considerada como sendo a temperatura ambiente.

O código *rhoCentralFoam* trabalha no domínio contínuo do problema de escoamento de fluidos, sendo assim, é preciso garantir que as condições iniciais permitam a operação e convergência do código. Neste sentido, a condição inicial mais importante para determinar a regime do escoamento é a pressão. Quanto mais baixa a pressão, mais fora do domínio contínuo o sistema está. Por outro lado, o PPT trabalha, tipicamente, em condições de pressão extremamente baixas ($\sim 10^{-4}$ Pa).

Portanto, a fim de permitir uma análise qualitativa do escoamento na cavidade do propulsor, sem violar a restrição de domínio contínuo, a pressão de 100 Pa foi determinada, através de testes de convergência do código, como um valor adequado para a pressão inicial do fluido. Abaixo deste nível de pressão, os resultados da simulação apresentam valores inconsistentes devido à mudança de regime do escoamento. Por exemplo, as velocidades do fluido são hipersônicas em pontos do escoamento onde a temperatura não converge.

A significativa diferença entre a pressão de operação do PPT e a pressão inicial de simulação, pode causar variações no padrão de escoamento do fluido na cavidade do PPT, principalmente na velocidade de saída e densidade de saída. Contudo, não há interferência no cálculo de massa ablacionada, *impulse-bit* e vazão mássica, pois estes parâmetros são calculados com a temperatura de superfície, densidade do PTFE e as equações de conservação aplicadas à camada de ablação.

Tabela 3.10 - Condições iniciais para velocidade, pressão e temperatura.

Campo	Valor Inicial
Velocidade (U)	uniform (0 0 0)
Pressão (p)	uniform 100
Temperatura (T)	uniform 300

Fonte: Próprio Autor

3.3.4 Propriedades físicas

Para configuração das propriedades do fluido de simulação usam-se dicionários de configuração. Estes são arquivos de configuração que se encontram no diretório *constant*, diretório este que está dentro da pasta raiz do caso de simulação em questão. Isto é retratado na figura 2.6. Os dicionários usados pelo código *rhoCentralFoam* especificam as propriedades termodinâmicas (*thermophysicalProperties*) e de turbulência (*turbulenceProperties*) do problema. Eles definem o modelo termofísico que será construído para o caso de simulação. Um modelo termofísico é construído no OpenFOAM como um sistema p-T de pressão e temperatura do qual outras propriedades são computadas.

O arquivo mais importante de configuração usado neste estudo é o *thermophysicalProperties*, listado no apêndice A. Neste arquivo são listados o modelo termofísico (*thermoType*) que será construído e as propriedades de composição do fluido (*mixture*). A entrada de dicionário *thermoType* é obrigatória e especifica o pacote de modelagem termofísica que é usado na simulação. A abordagem de codificação do OpenFOAM monta pacotes de modelagem termofísica começando com a equação de estado e, em seguida, adicionando mais camadas de modelagem termofísica que herdam propriedades da camada anterior. As entradas do sub-dicionário (palavras-chave) *thermoType* refletem as várias camadas de modelagem e a estrutura subjacente na qual elas são combinadas.

A Tabela 3.11 contém os dados de configuração do modelo usado na simulação.

Tabela 3.11 - Tabela com os dados do campo *thermoType*.

type	hePsiThermo
mixture	pureMixture
transport	const
thermo	hConst
equationOfState	perfectGas
specie	specie
energy	sensibleInternalEnergy

Fonte: Próprio Autor

A entrada *hePsiThermo* especifica o modelo termofísico de composição química fixa e baseado em compressibilidade. Todos códigos compressíveis do OpenFOAM utilizam este modelo.

A entrada *mixture* especifica a composição da mistura. A opção normalmente usada para modelos termofísicos sem reações é *pureMixture*, que representa uma mistura com composição fixa. Quando *pureMixture* é especificado, os coeficientes da mistura são especificados em um sub-dicionário chamado *mixture*.

A entrada *transport* especifica a modelagem de transporte que, por sua vez, define a forma como são calculadas a viscosidade dinâmica μ , condutividade térmica κ e difusividade térmica α . Com o valor *const* para esta entrada, assume-se que μ e o número de Prandtl ($Pr = c_p\mu/\kappa$) são constantes e especificados por duas entradas, *mu* e *Pr*, respectivamente.

A entrada *thermo* define o modelo termodinâmicos, que busca calcular o calor específico e do qual outras propriedades são derivadas. O valor *hConst* assume que c_p , o calor específico a temperatura constante e H_f , o calor de fusão, são constantes. Estes valores são especificados pelas entradas *cp* e *Hf*, respectivamente.

A entrada *equationOfState* define a equação de estado para o fluido. A equação dos gases perfeitos foi escolhida devido ao fato de gases em baixa pressão, como é o caso neste estudo, se aproximarem muito bem de um gás ideal.

A entrada *specie* define a composição de cada constituinte na mistura. Porém, somente um valor em *mixture* é usado com *specie*, o peso molar da espécie, representado por *nMoles*.

A última entrada *energy* especifica a variável usada na solução do problema. No estudo em questão, a energia interna específica é usada como variável.

Na Tabela 3.12 os valores para simulação do gás de PTFE encontram-se listados. Os valores são baseados no monômero do tetrafluoretileno, cuja fórmula é C_2F_4 e tem peso molar igual a 100 g/mol.

Tabela 3.12 - Tabela com os dados do campo *mixture*.

specie	nMoles	1
	molWeight	100
thermodynamics	Cp	1004.5
	Hf	0
transport	mu	1.8e-05
	Pr	0.7

Fonte: Próprio Autor

Para simplificação dos processos físicos simulados e conseqüente diminuição do tempo de simulação, a turbulência foi desconsiderada neste trabalho. O dicionário correspondente, *turbulenceProperties*, especifica a simulação como laminar.

3.4 Condição de Contorno de Ablação

Conforme citado na seção 1.3.1, um dos fenômenos centrais do funcionamento do PPT é a ablação do propelente (PTFE) e aceleração do mesmo através de processos magneto-termodinâmicos. A ablação, por sua vez, é a remoção de material da superfície de um objeto por vaporização, lascamento ou outros processos erosivos.

No PPT, a ablação do propelente se dá através de uma descarga elétrica, que vaporiza uma fina camada da superfície do PTFE. A vaporização do PTFE começa, no entanto, apenas acima de uma determinada temperatura, chamada aqui de $T_{Ablation}$. Abaixo desta temperatura, a superfície do PTFE apenas aquece estando exposta a fonte de calor, como aconteceria com qualquer outro material.

Logo, para modelar o processo de ablação do PTFE, é necessário que exista uma condição de contorno que mude de comportamento de acordo com a temperatura da superfície na qual ela está aplicada. Este tipo de condição de contorno não existia nativamente no OpenFOAM, portanto, foi necessário o seu desenvolvimento.

O problema básico desta condição de contorno pode ser modelado nas seguintes etapas:

- 1) O circuito elétrico RLC gera uma descarga elétrica;
- 2) A corrente desta descarga passa através da superfície do PTFE;
- 3) Como existe resistência à passagem da corrente elétrica através da superfície, existe uma fonte de calor (Efeito Joule);
- 4) Esta fonte de calor aquece a superfície do PTFE, até a temperatura de ablação;
- 5) A partir desta temperatura, existe a ejeção de massa para o escoamento (plasma) e o conseqüente resfriamento da superfície do PTFE.

O primeiro passo, para implementação da condição de contorno, é o cálculo do calor gerado a partir do Efeito Joule da corrente de descarga. Para isso, supõe-se que toda a resistência do circuito RLC se encontra sobre a superfície exposta do PTFE e que toda corrente elétrica se espalha sobre esta superfície. Com isso, deduz-se as seguintes equações:

$$P = I^2 * R \quad (3.10)$$

onde P é potência gerado pela corrente I de descarga, e R é a resistência do circuito RLC, presente na Tabela 3.1.

$$Q = \frac{P}{AreaTotal} \quad (3.11)$$

onde Q é o calor gerado na superfície com área dada por $AreaTotal$, esta área corresponde à superfície exposta do PTFE e é dada pela geometria específica do propulsor. Para o PPT retangular considerado neste estudo, esta área equivale a $3cm^2$ (3cm x 1cm).

O aquecimento do PTFE, dado o calor Q , é modelado simplesmente por uma condição de Neumann ou *fixedGradient*, no linguajar do OpenFOAM. Esta condição é representada através da equação:

$$-k\nabla T = Q \quad (3.12)$$

onde ∇T é o gradiente da temperatura na superfície e k é condutividade térmica.

Acima de $T_{Ablation}$, a condição de contorno deixa de ser apenas o aquecimento da superfície e passa a contar também com o resfriamento causado pela vaporização do PTFE. A equação que modela este novo comportamento é dada por:

$$-k\nabla T + DeltaH * T_{Ablation} * M = Q \quad (3.13)$$

onde $DeltaH$ é o calor de vaporização efetivo, M é a massa ablada por unidade de tempo e $T_{Ablation}$ é a temperatura de ablação.

Então, basicamente, o que a condição de ablação deve fazer é monitorar a temperatura da superfície e chavear entre a Equação 3.12 e a Equação 3.13, de acordo com esta temperatura. Os valores de $DeltaH$ e k são lidos do sub-dicionário de configuração da condição de contorno de ablação. Os valores que variam, como I e M , são lidos de dentro do código de simulação principal. Os códigos-fonte de implementação desta condição de contorno encontra-se no Apêndice B.

3.5 Código *PPT_Foam*

O nome do simulador desenvolvido é *PPT_Foam* e sua estrutura principal é baseada no código *rhoCentralFoam*. O fluxograma mostrado na Figura 3.7 resume bem a ordem de execução do código *PPT_Foam*.

Todas as ações dentro do laço de repetição já foram analisadas anteriormente, com exceção de uma: atualização de campos associados a ablação.

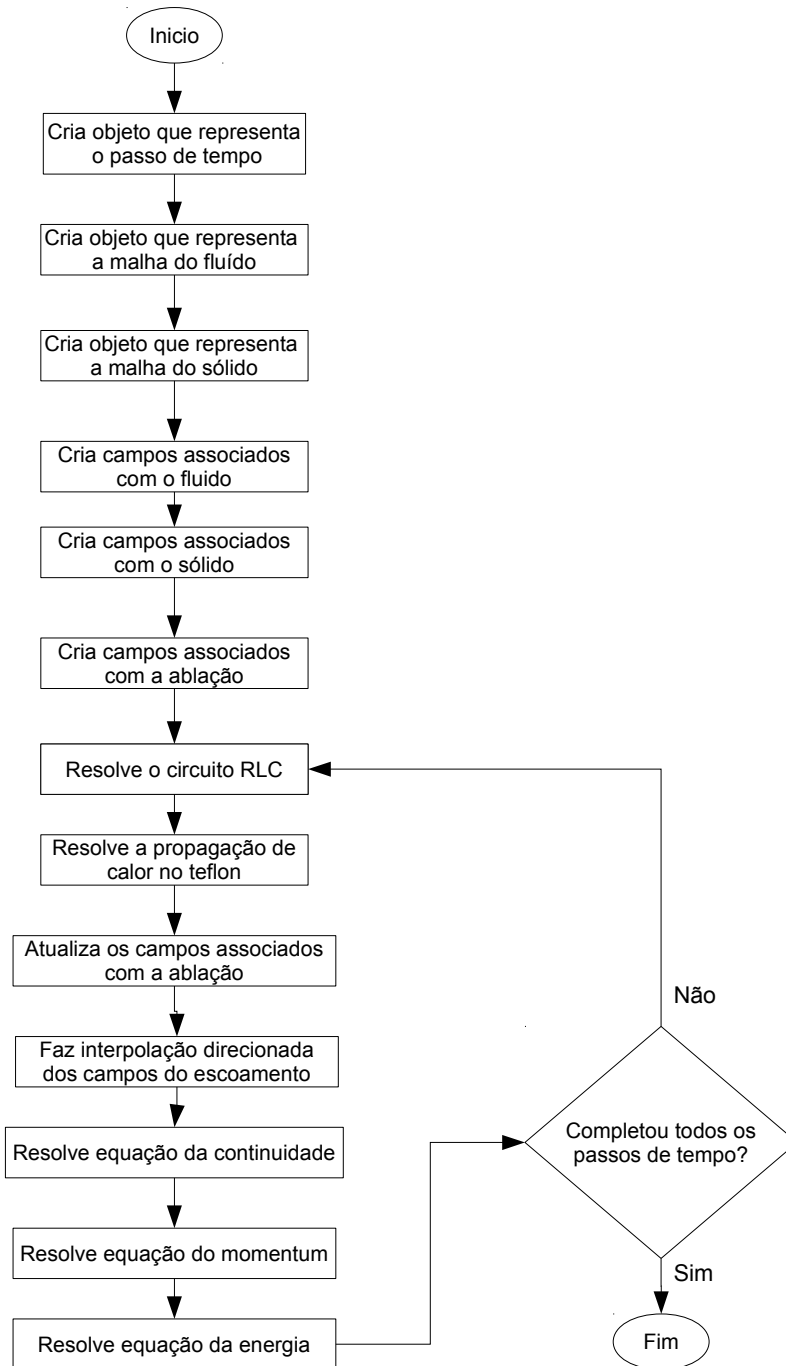
Esta parte do código é complementar à condição de contorno de ablação, pois usa a temperatura da superfície do PTFE (computada com a condição de contorno de ablação) para decidir se há ou não injeção de matéria no domínio do fluido.

Se a temperatura interna do PTFE (T_0) passar da $T_Ablation$, a ablação acontece e material é injetado no domínio do fluido. Para saber a quantidade de PTFE expelido da superfície, bem como sua temperatura e sua velocidade de ejeção usam-se as equações 2.18, 2.19 e 2.12, da seguinte maneira:

A pressão (P_0) é calculada a partir da temperatura de superfície (T_0). Com a pressão e temperatura calculadas pode-se descobrir, através da equação de gases perfeitos, a densidade (n_0). Ao fim deste processo, obtêm-se T_0 , P_0 e n_0 . Estabelecendo-se algum valor para α , é possível usar as equações 2.18, 2.19 e 2.12 para se descobrir T_1 , n_1 e a velocidade U_1 de injeção de PTFE na cavidade do propulsor. Para resolver este sistema de equações, um perfil para α é imposto. Neste estudo usou-se um α que varia de forma parabólica com o tempo e está centrado no começo do pulso elétrico.

O código *updateAblationFields* que implementa a funcionalidade descrita acima, bem como todos os códigos fonte do *PPT_Foam*, encontram-se no Apêndice C.

Figura 3.7 - Fluxograma de execução do código PPT_Foam.



Fonte: Próprio Autor.

4 RESULTADOS

Os resultados apresentados a seguir foram gerados para a energia inicial armazenada no capacitor de 1,85J. A simulação mostra o escoamento durante um intervalo de 100 μ s. Todas as simulações/compilações foram feitas em um laptop AVELL com processador Intel(R) Core(TM) i7-4700MQ CPU @ 2.40GHz, 8GB de memória RAM e 500GB de espaço em disco.

4.1 Escoamento na Cavidade do PPT

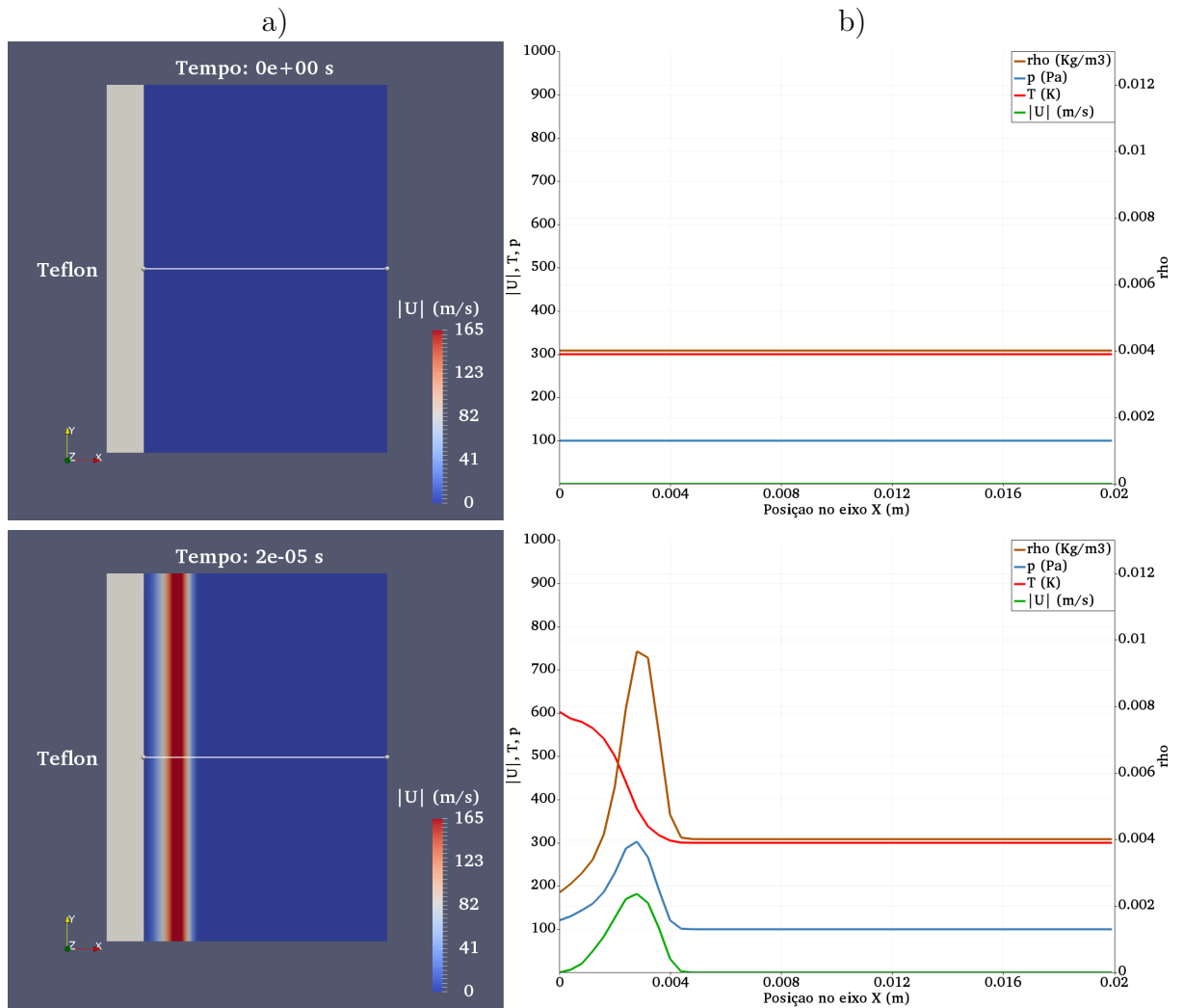
As Figuras 4.1-4.3 mostram o escoamento na cavidade do propulsor em diferentes instantes de tempo. Em cada instantâneo, o lado “a” retrata a magnitude da velocidade do fluido na cavidade do propulsor e uma linha de medição, enquanto o lado “b” mostra os perfis de densidade, pressão, temperatura e magnitude de velocidade. Estes perfis são gerados sobre a linha de medição horizontal que corta a cavidade do propulsor ao meio, mostrada no lado “a” de cada instantâneo.

Nas figuras é possível observar um pulso de alta densidade e pressão se movendo rapidamente para a direita, na direção do plano de saída do PPT. O aumento da densidade se dá pela ejeção de massa da superfície do PTFE ablacionado. É importante salientar que a escala de medida da densidade fica à direita dos gráficos de perfil das variáveis.

No pulso de ablação há o aumento da densidade, pressão e temperatura, em relação aos seus respectivos valores iniciais. No entanto, a temperatura aumenta em proporção consideravelmente menor que estas outras variáveis. Isto acontece porque grande parte da energia da massa que se desloca é cinética e não térmica. Além disso, como houve a injeção de massa no domínio, e não apenas a compressão de uma massa já existente, o aumento de temperatura diminui ainda mais pela lei do gases ideais.

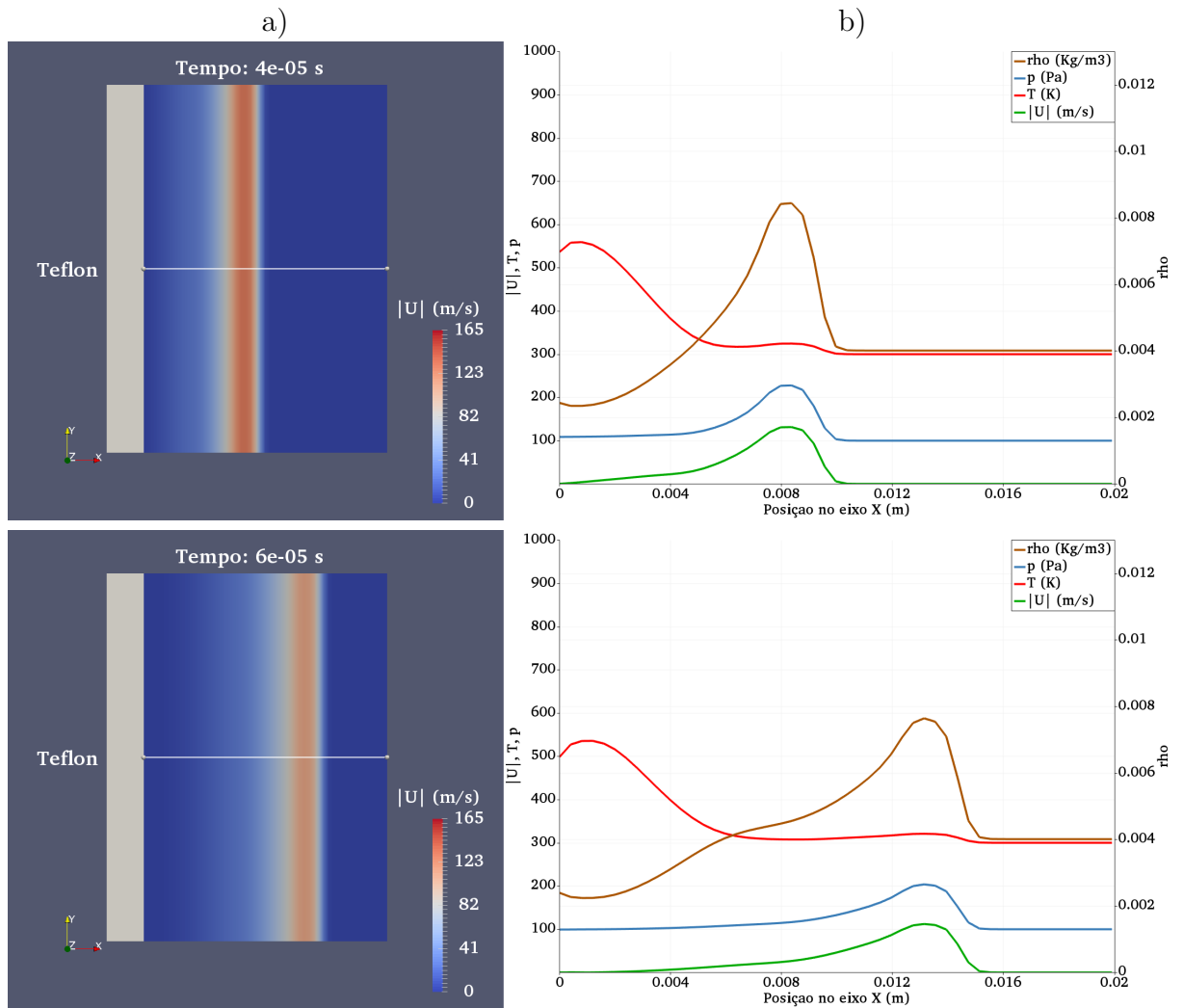
Outro ponto interessante é a presença de um bolsão de alta temperatura próximo à superfície recém ablacionada do PTFE, como retratado a partir da Figura 4.2b.

Figura 4.1 - Cavidade do propulsor e distribuição de densidade, pressão, temperatura e velocidade ao longo do eixo central, nos instantes de 0 e 20 μ s.



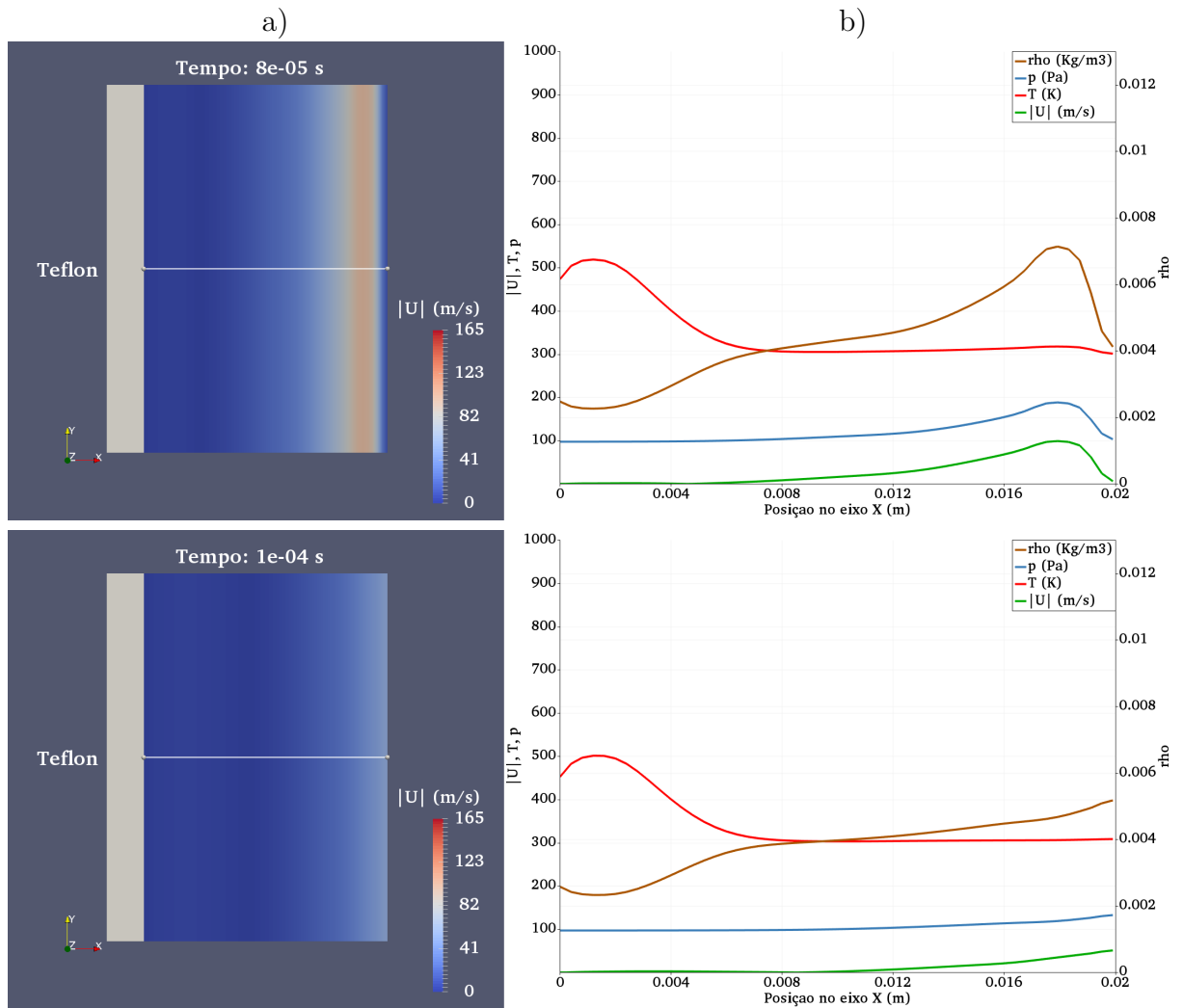
Fonte: Próprio Autor

Figura 4.2 - Cavidade do propulsor e distribuição de densidade, pressão, temperatura e velocidade ao longo do eixo central, nos instantes de 40 e 60 μs .



Fonte: Próprio Autor

Figura 4.3 - Cavidade do propulsor e distribuição de densidade, pressão, temperatura e velocidade ao longo do eixo central, nos instantes de 80 e 100 μ s.

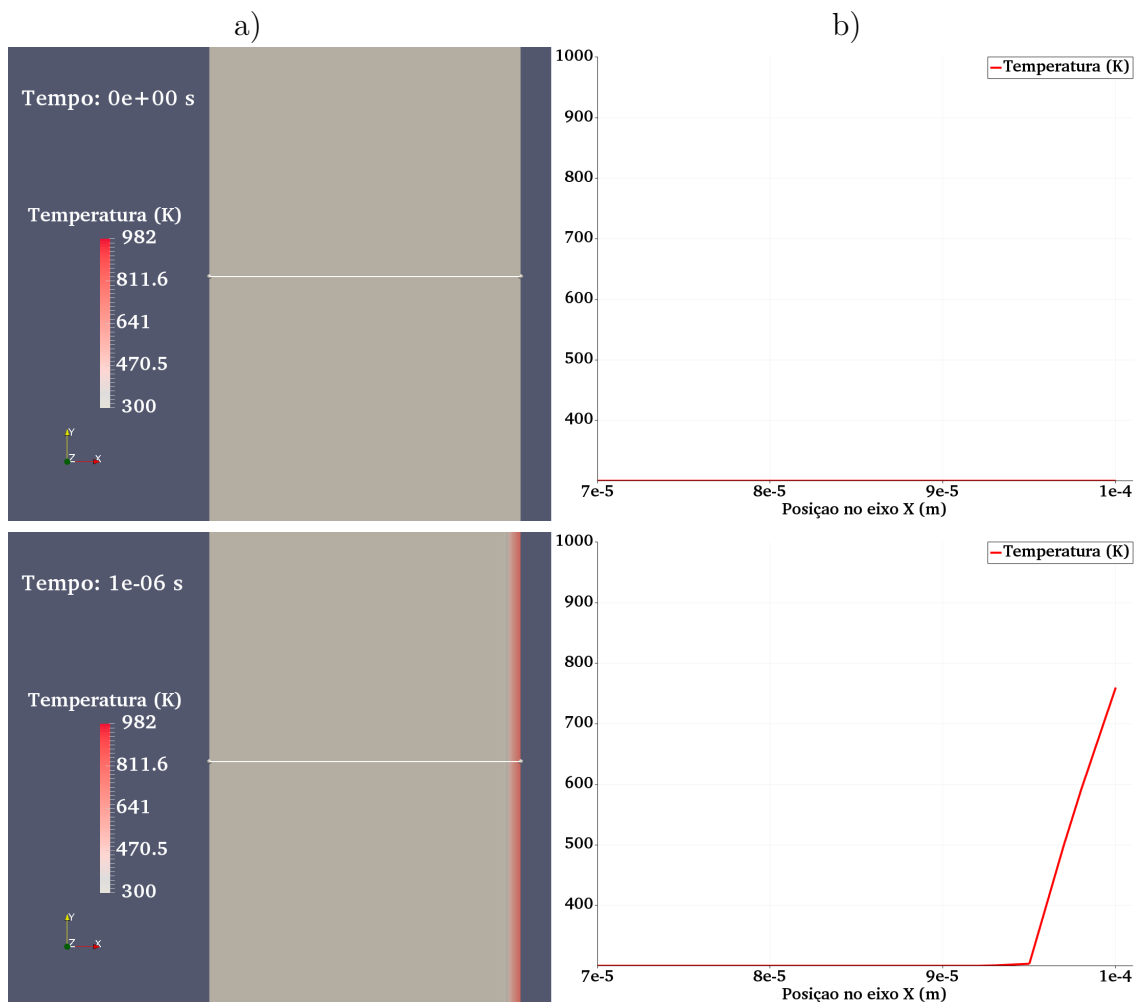


Fonte: Próprio Autor

4.2 Propagação de Calor no PTFE

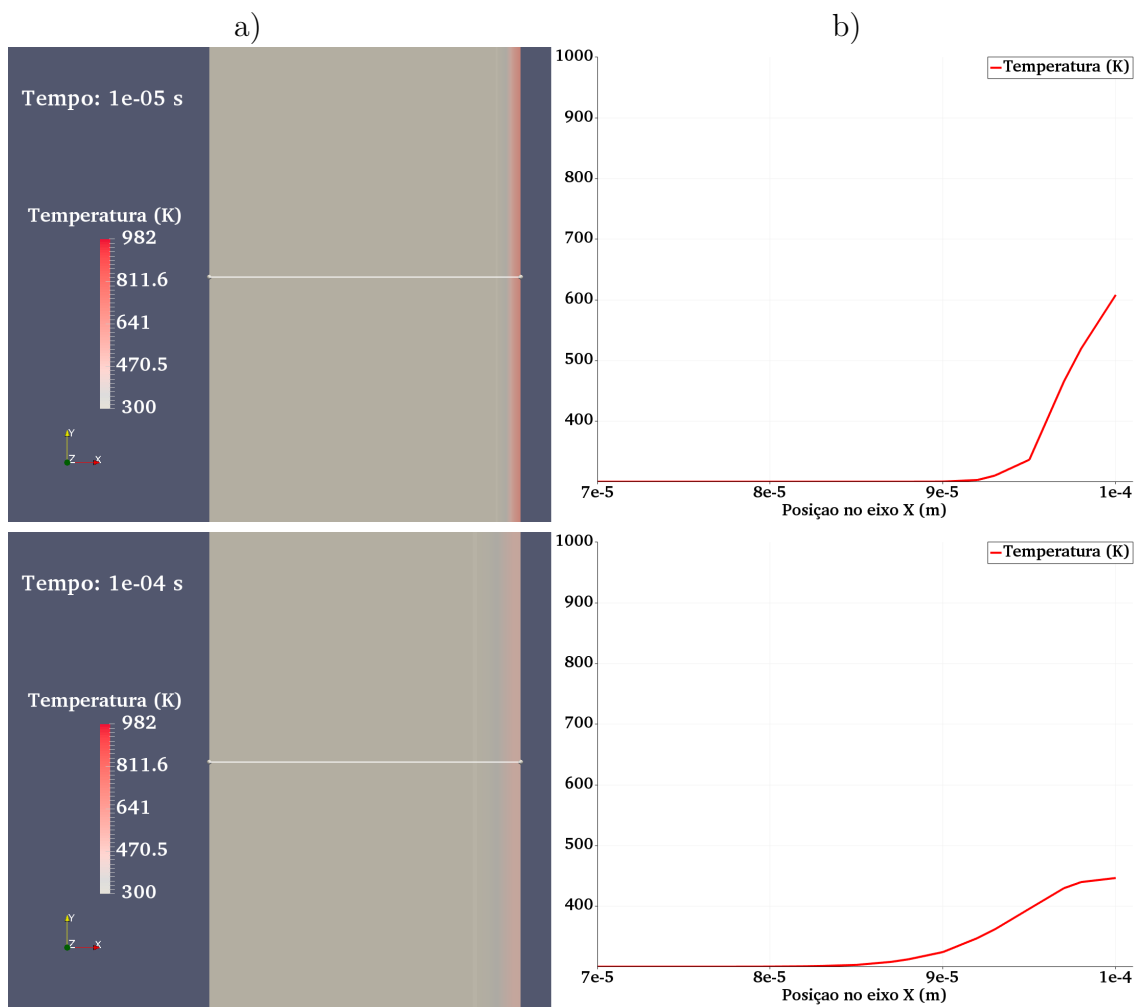
A Figura 4.4 mostra a distribuição de temperatura dentro da barra de PTFE em diferentes instantes de tempo. Os perfis de temperatura, do lado “b” dos instantâneos, são gerados sobre uma linha horizontal que corta a barra de PTFE ao meio, na direção do eixo X, semelhante a linha mostrada no lado “a” das Figuras 4.1-4.3. Pode-se verificar que a temperatura próxima à superfície do PTFE começa a diminuir logo após a ejeção do pulso de matéria ablacionada. Além disso, existe a propagação do calor para o interior da barra de PTFE.

Figura 4.4 - Barra de PTFE e distribuição de temperatura ao longo do eixo central, em diferentes instantes de tempo.



Fonte: Próprio Autor

Figura 4.5 - Barra de PTFE e distribuição de temperatura ao longo do eixo central, em diferentes instantes de tempo.

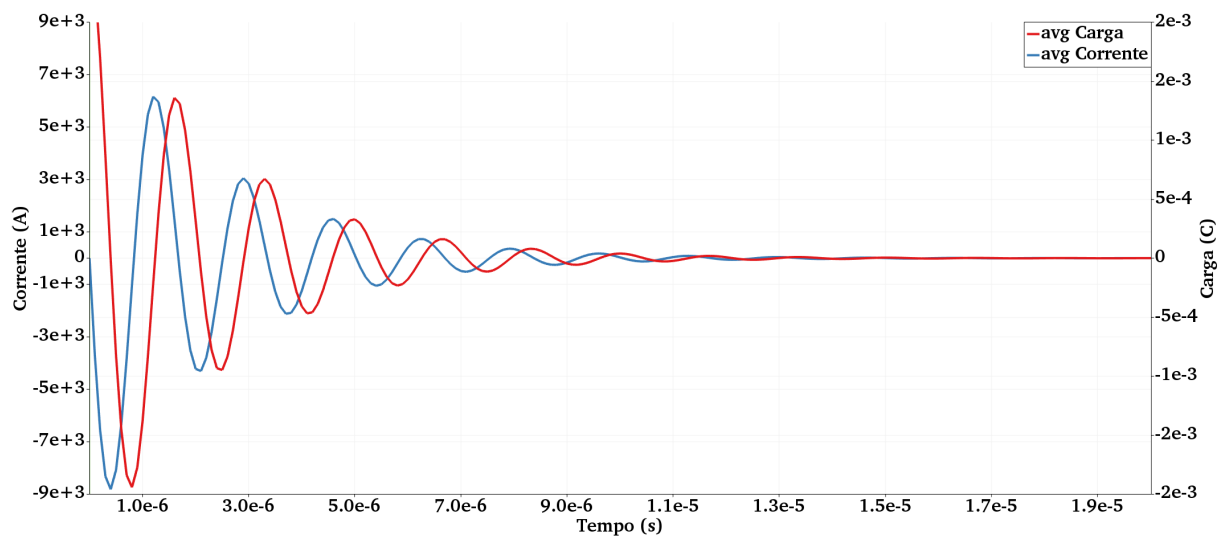


Fonte: Próprio Autor

4.3 Circuito RLC

Na Figura 4.6 as formas de onda, da corrente e carga no capacitor do circuito RLC, são plotadas sobrepostas em diferentes escalas. Nesta figura nota-se a defasagem de 90 graus entre a carga e a corrente, típica de um capacitor. É possível observar que o pulso de descarga tem comportamento subamortecido e que dura cerca de $10 \mu\text{s}$.

Figura 4.6 - Formas de onda de corrente e carga sobrepostas.

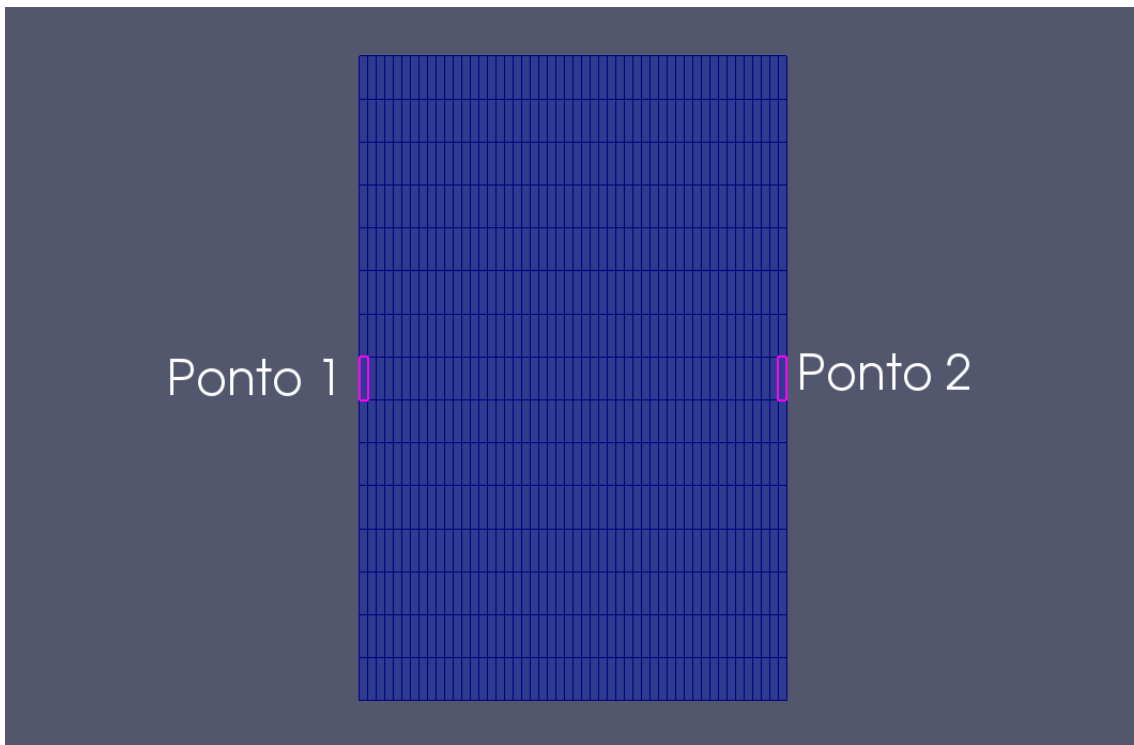


Fonte: Próprio Autor.

4.4 Propriedades da Camada de Ablação

O ponto de medição 0 corresponde ao interior da célula na superfície da barra de PTFE que sofre ablação, este ponto de medição é interno ao código, não sendo retratado em nenhuma figura. Os pontos 1 e 2 correspondem a entrada e saída da cavidade do PPT, respectivamente. Estes pontos estão representados na Figura 4.7.

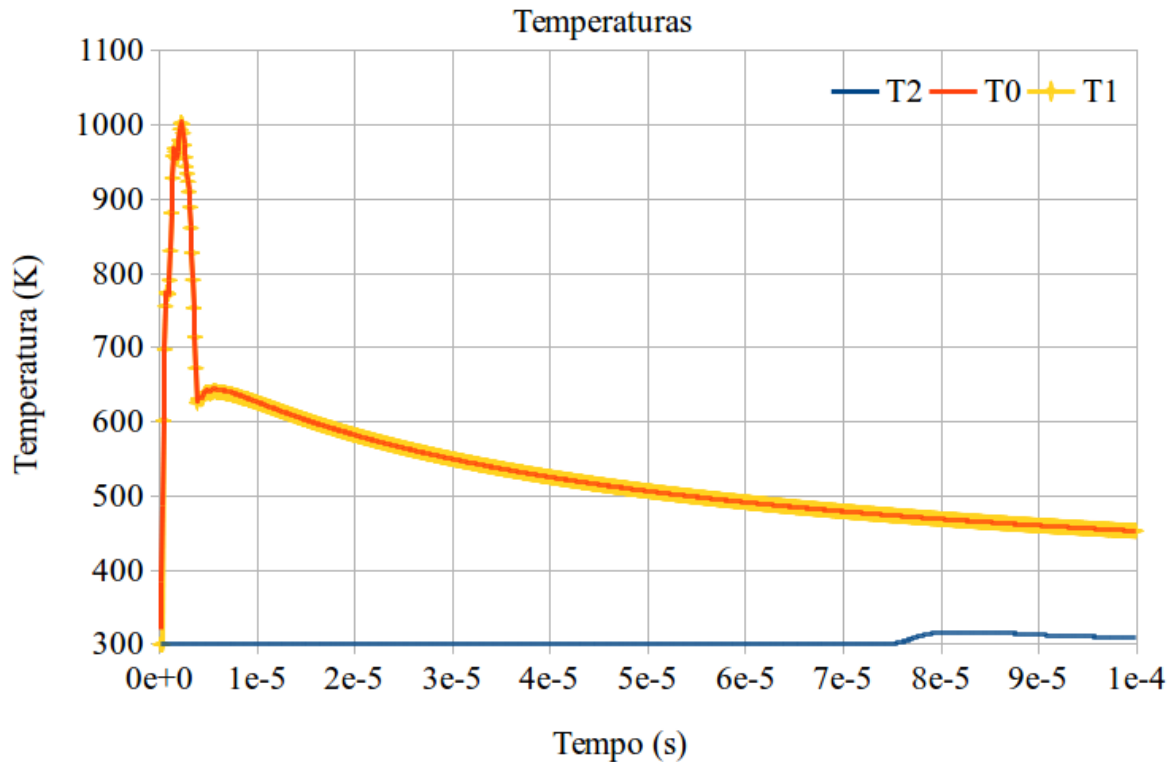
Figura 4.7 - Pontos de sondagem.



Fonte: Próprio Autor.

Na Figura 4.8 os perfis de temperatura são mostrados. No ponto 1, adjacente ao PTFE, a temperatura é praticamente igual àquela no interior da célula, ponto 0. Nota-se que existem pequenas ondulações nos picos dos gráficos de temperatura do ponto 0 e 1, isto é decorrência direta do fato de que o calor, que gera o aumento de temperatura, é proporcional ao quadrado da corrente. E a corrente, por sua vez, é senoidal. Ou seja, estas ondulações são a impressão do formato do pulso de corrente no perfil de temperatura da superfície do PTFE.

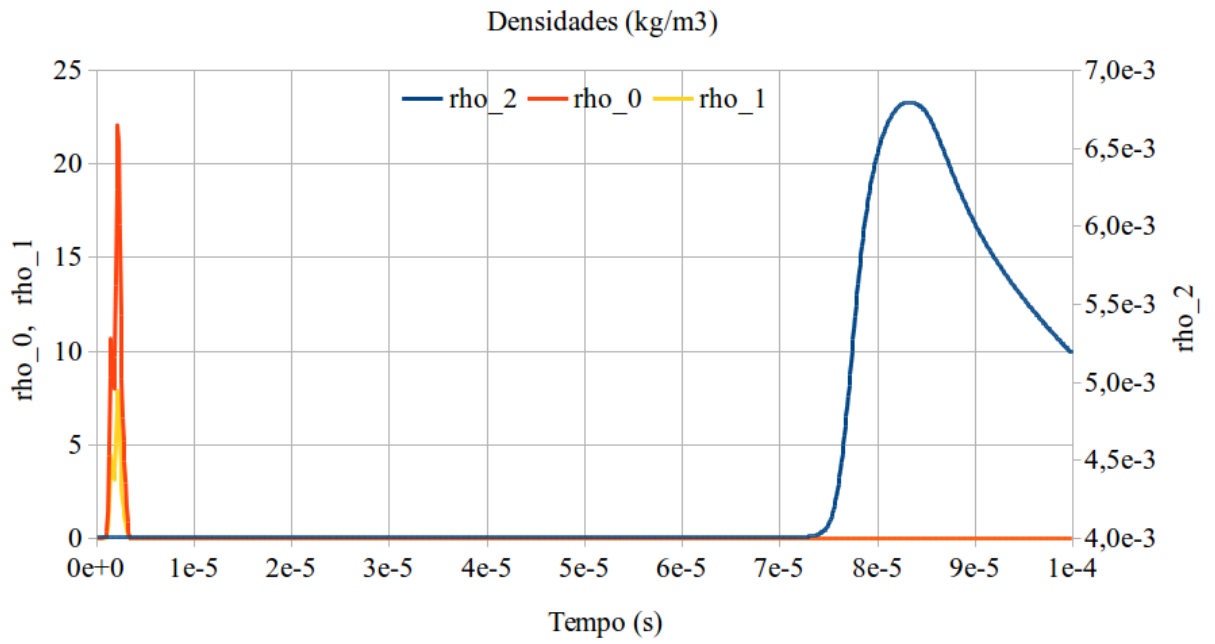
Figura 4.8 - Temperaturas nos pontos 0, 1 e 2.



Fonte: Próprio Autor

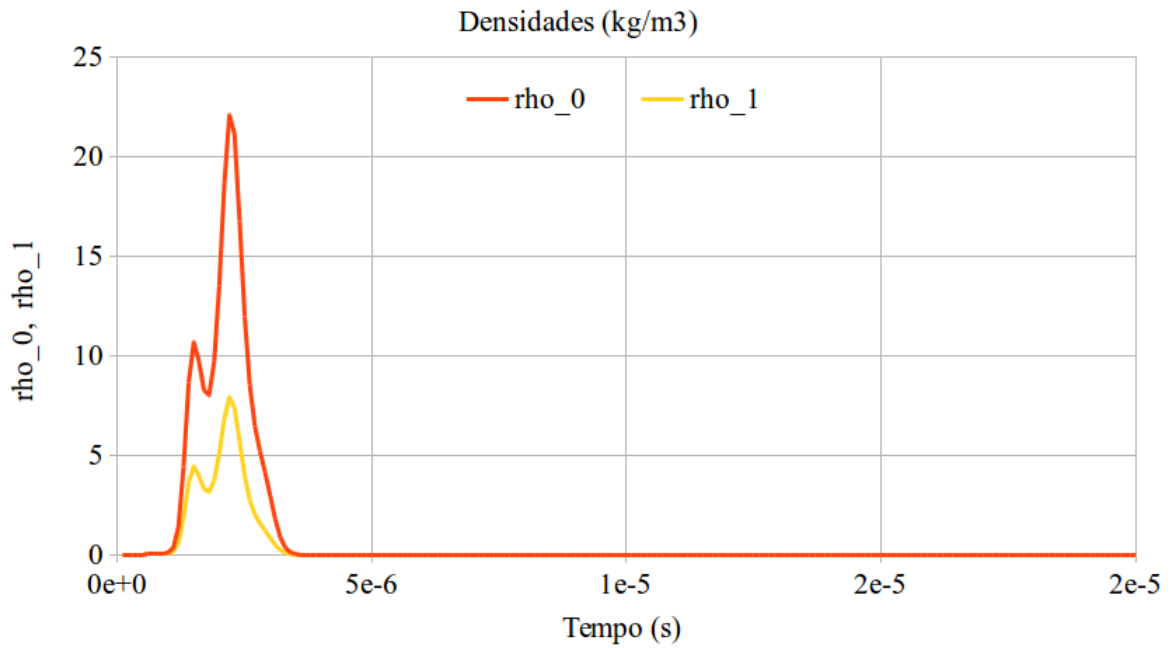
A Figura 4.9 mostra as densidade nos pontos 0, 1 e 2 em diferentes escalas. As densidades dos pontos 0 e 1 estão na escala da esquerda do gráfico, enquanto a densidade no ponto 2 está na escala da direita. Novamente o formato do pulso de corrente é vista nas densidades dos pontos 0 e 1. Nota-se que, apesar do densidade na saída do propulsor (ponto 2) ser 1000 vezes menor que na entrada, a duração do pulso da densidade de saída é bem maior, indicando a preservação da massa total no sistema. Na Figura 4.10 os formatos de onda das densidades nos pontos 0 e 1 são mostrados em detalhe para melhor compreensão.

Figura 4.9 - Densidades nos pontos 0, 1 e 2.



Fonte: Próprio Autor

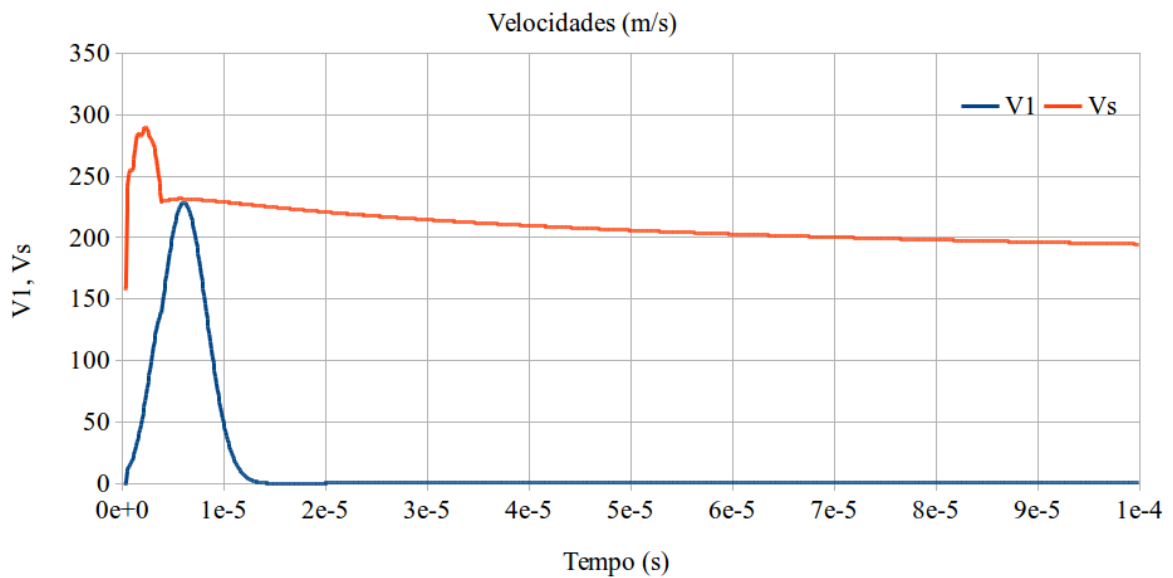
Figura 4.10 - Detalhe ampliado da Figura 4.9.



Fonte: Próprio Autor

Os gráficos da Figura 4.11 indicam a velocidade no ponto 1, adjacente a superfície do PTFE, e a velocidade do som também no ponto 1. A velocidade do som é calculada como uma função apenas da temperatura, considerando o fluido como um gás ideal. Percebe-se que, próximo ao tempo de $6 \mu s$ a condição de velocidade sônica é atingida, algo já previsto na literatura (KEIDAR et al., 2001)(ZAGHLOUL, 2004).

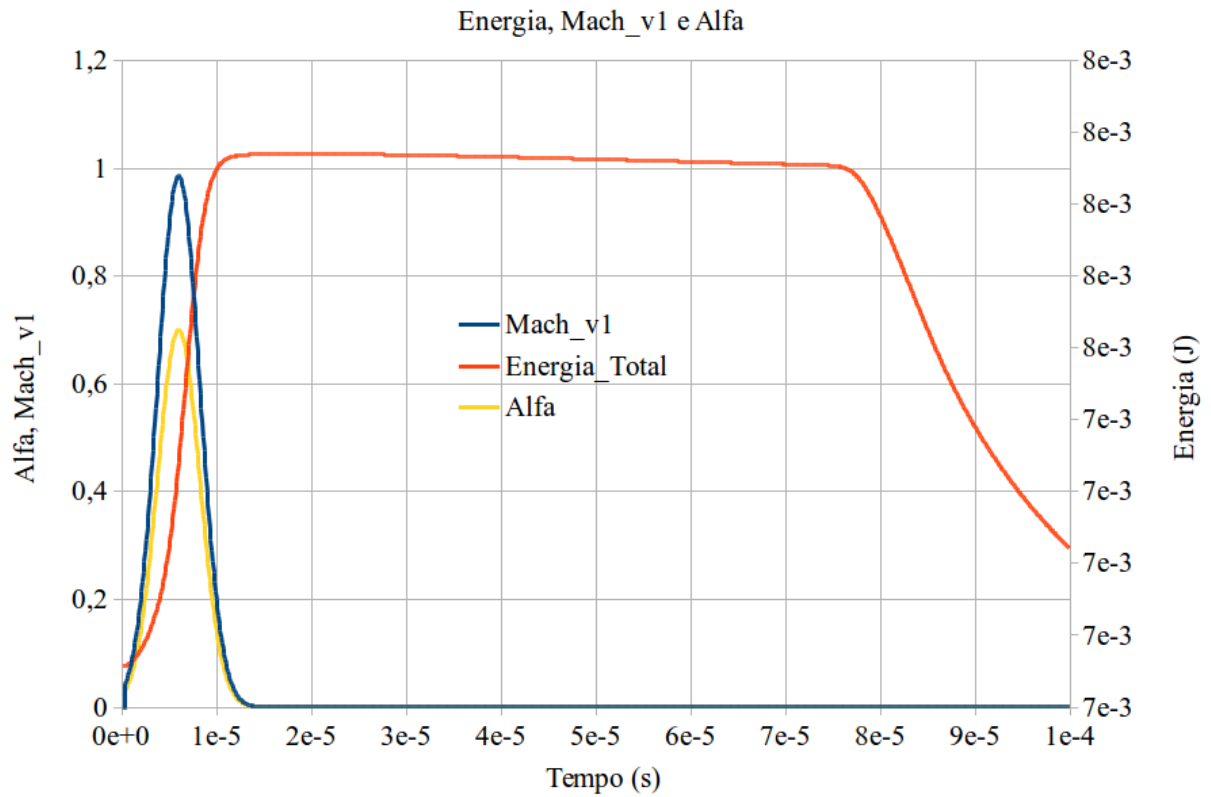
Figura 4.11 - Velocidade no ponto 1 (V1) e velocidade do som no meio (Vs).



Fonte: Próprio Autor

A Figura 4.12 retrata o número de Mach no ponto 1, a energia total na cavidade do propulsor e o parâmetro adimensional α . A velocidade do som é atingida pelo fluido, como indicado pelo número de Mach igual a 1, no instante $6 \mu s$. O formato do gráfico de energia é decorrência direta do fato de que a energia injetada pelo pulso na câmara do propulsor se matem até que o mesmo saia do propulsor. Como dito anteriormente na Seção 3.5, parâmetro α varia de maneira parabólica com o tempo e é imposto a fim de que seja possível resolver o sistema de equações que descrevem a ablação, a partir da teoria cinética.

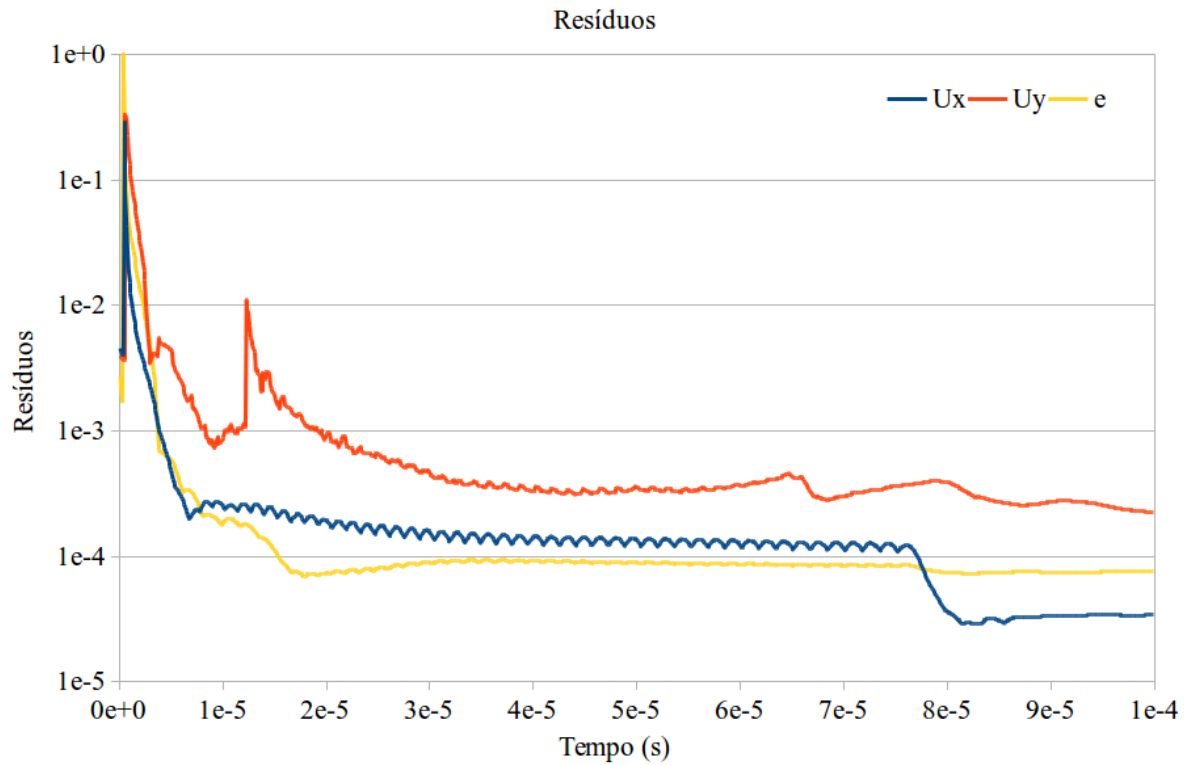
Figura 4.12 - Energia do fluido, Número de Mach e constante adimensional α .



Fonte: Próprio Autor

Na Figura 4.13 são plotados os resíduos das variáveis resolvidas de maneira implícita (iterativa) pelo código *rhoCentralFoam*. As outras variáveis são resolvidas explicitamente tendo, portanto, resíduo zero. Os três gráficos tendem a estabilização após uma fase de transição, mas de maneira oscilatória. Isto se deve, provavelmente, as variações abruptas das propriedades do fluido, causadas pelo pulso de matéria expelida e, também, a ausência de técnicas de estabilização e amortecimento no acoplamento entre os códigos.

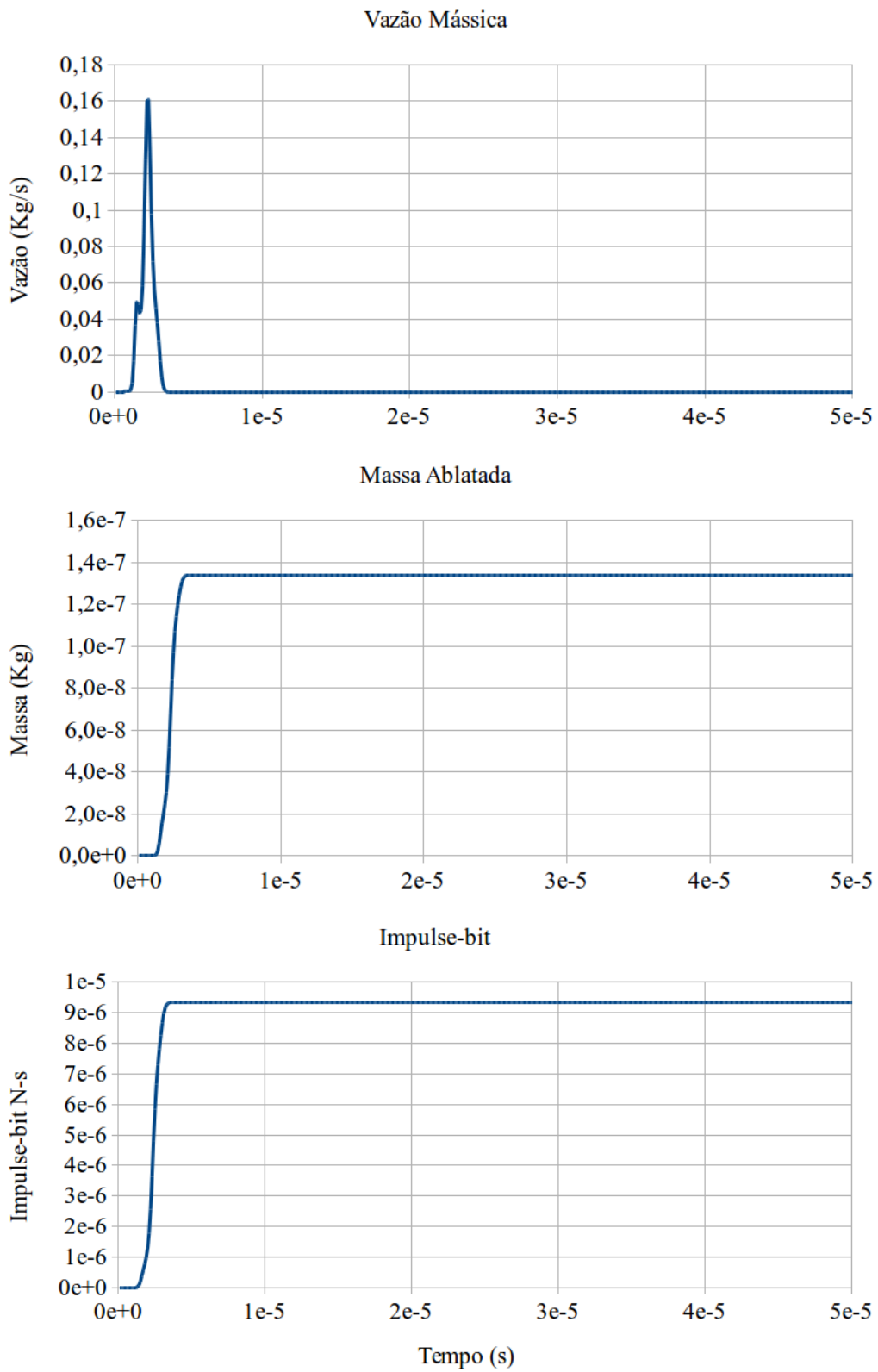
Figura 4.13 - Resíduos das variáveis de simulação.



Fonte: Próprio Autor

Por fim, a Figura 4.14 mostra os principais parâmetros usados na avaliação de propulsores do tipo PPT, a massa ablacionada, o *impulse-bit* e a vazão mássica. Novamente, observa-se no gráfico da vazão mássica a impressão do pulso de corrente.

Figura 4.14 - Gráficos das principais propriedades plotados em conjunto para melhor comparação.



Fonte: Próprio Autor

5 ANÁLISE DOS RESULTADOS

Nesta capítulo são discutidos os resultados obtidos e sua relação com os dados disponíveis na literatura. Os dados de comparação foram retirados, principalmente, do trabalho desenvolvido por Mikellides (1999) e Turchi e Mikellides (1995) ao estudar a aplicação do código MACH2 na simulação de propulsores PPT.

5.1 Corrente do Circuito RLC

Na Figura 5.1, a forma de onda da corrente de descarga calculada pelo código MACH2 é mostrada. Na simulação realizada por Turchi e Mikellides (1995) para obtenção desta forma de onda, o circuito elétrico equivalente utilizado é composto por dois outros circuitos elétricos.

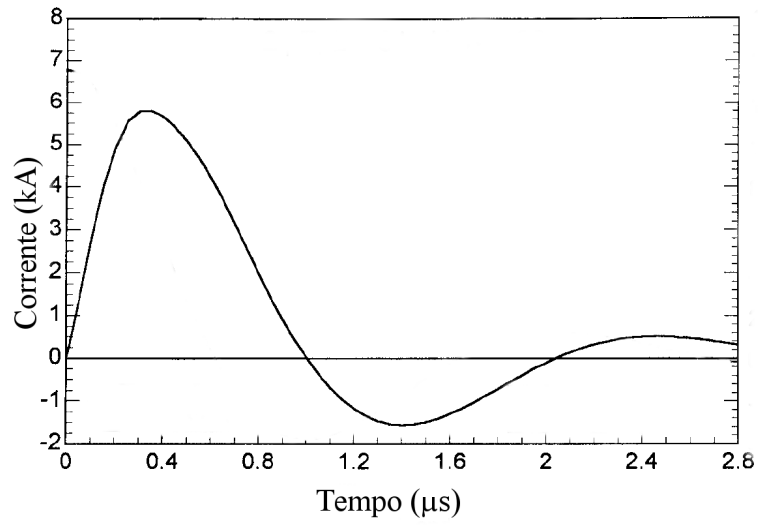
O primeiro é o circuito de descarga RLC, apresentado na Figura 3.1 e cujo os parâmetros são dados na Tabela 3.1. A resistência, indutância e capacitância deste circuito se mantêm fixas durante a descarga elétrica, ou seja, pode-se dizer que estes parâmetros são fixos no tempo.

O segundo circuito elétrico é o circuito do plasma de PTFE, este circuito modela o comportamento elétrico do plasma. Como as características do plasma variam com o tempo, os parâmetros elétricos do mesmo também variam. Esta variação do valores de resistência, indutância e capacitância do plasma geram um circuito elétrico que varia de comportamento no tempo.

O circuito elétrico equivalente, formado pelo circuito do plasma e o circuito RLC externo, tem seus parâmetros elétricos alterados continuamente. Como consequência direta disso, a forma de onda deste circuito, obtida por Turchi e Mikellides (1995) e apresentada na Figura 5.1, tem valores de pico e duração de descarga diferentes dos obtidos com o circuito RLC fixo, cuja forma de onda é dada na Figura 5.2.

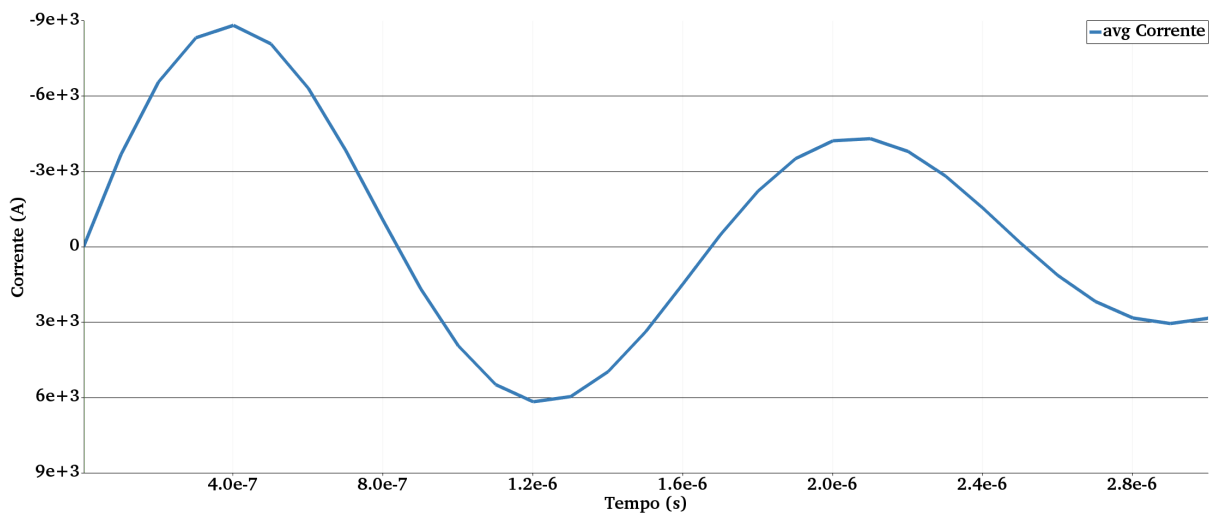
Como calor gerado na superfície é diretamente proporcional ao quadrado da corrente, um pulso de corrente com duração e pico máximo maiores implica em mais calor transferido para a superfície do propelente.

Figura 5.1 - Corrente calculada pelo MACH2.



Fonte: Adaptada de Turchi e Mikellides (1995)

Figura 5.2 - Formas de onda de corrente e carga sobrepostas.



Fonte: Próprio Autor.

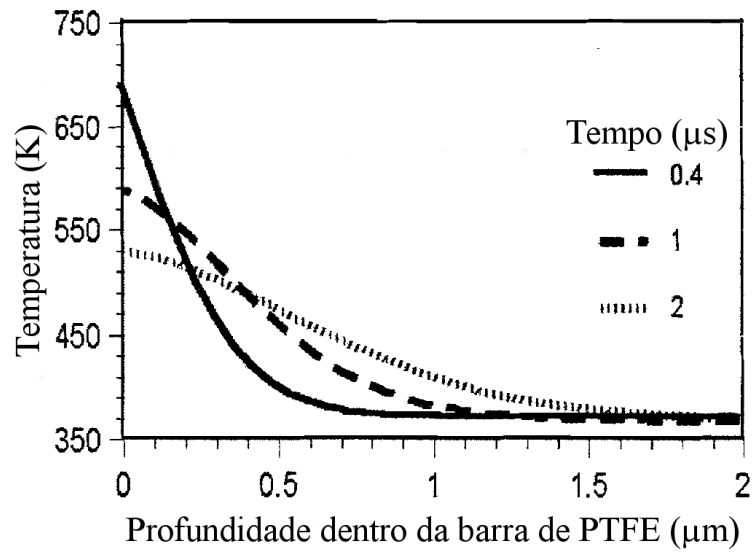
5.2 Distribuição de Temperatura na Barra de PTFE

Na Figura 5.3 o perfil de temperatura dentro da barra de PTFE, calculado pelo código MACH2, é apresentado. Nesta figura nota-se a propagação do calor para dentro do propelente e a conseqüente diminuição da temperatura de superfície do mesmo. É importante ressaltar que a temperatura de decomposição considerada por Mikellides (1999) em seu estudo é da ordem de 600 K, enquanto neste trabalho considera-se que a temperatura seja de 650 K. É possível admitir uma temperatura de decomposição diferente para o PTFE, pois o mesmo é uma mistura de polímeros com diferentes pontos de decomposição. Temperaturas de decomposição de até 700 K podem ser encontradas na literatura (LEI et al., 2015).

Além disso, as resoluções espacial e temporal da simulação realizada com o programa MACH2 são maiores do que as obtidas no presente estudo. Na Figura 5.3 o comprimento total sob análise é de $2 \mu m$ e o intervalo de tempo mínimo é $0,4 \mu s$. A resolução de tempo mínima obtida no presente trabalho com OpenFOAM é de $1 \mu s$. Nos instantâneos retratados na Figura 5.4, nos tempos de $1 \mu s$ e $2 \mu s$, percebe-se a variação abrupta do perfil de temperatura devido a resolução espacial insuficiente frente resolução temporal necessária, ou seja, o calor recebido pela superfície, em um curto intervalo de tempo, faz com que temperatura aumente rapidamente de um passo de tempo para o outro. Apesar das diferentes escalas de tempo, o perfil de distribuição de temperatura dentro do PTFE, calculada pelo código *rhoCentralFoam*, aproxima-se daquele obtido com o código MACH2.

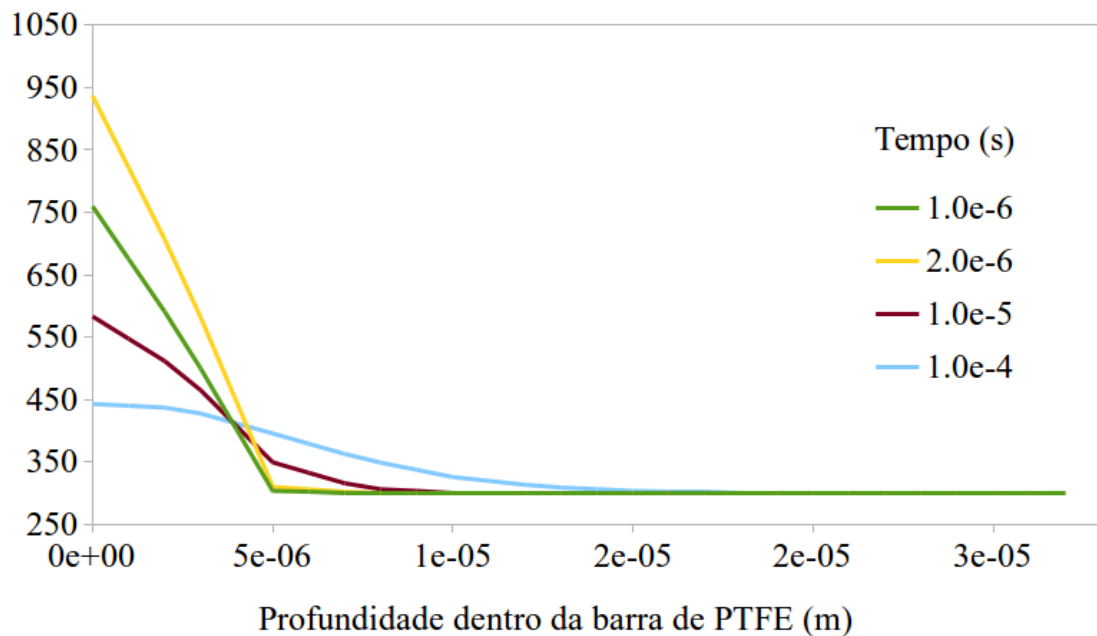
Comparando a Figura 5.3 com os perfis de temperatura apresentados na Figuras 5.4, nota-se o mesmo padrão de distribuição da temperatura com o tempo. Durante o instante inicial, a temperatura da superfície do PTFE ultrapassa consideravelmente sua temperatura de decomposição, decrescendo rapidamente, logo em seguida, devido a ablação da superfície. Também percebe-se pela Figura 5.4 que o pico máximo de temperatura na superfície do propelente é consideravelmente maior na Figura 5.3. Isto se deve ao maior calor recebido através da corrente de descarga.

Figura 5.3 - Perfil de temperatura dentro do PTFE, calculado pelo código MACH2.



Fonte: Adaptada de Mikellides (1999)

Figura 5.4 - Perfil de temperatura dentro do PTFE, calculado pelo código *rhoCentralFoam*.



Fonte: Próprio Autor

5.3 Massa Ablacionada e *Impulse-bit*

Na Tabela 5.2 os valores típicos para o PPT de modelo LES-6, na qual se baseia este trabalho, são apresentadas. Na Tabela 5.3 os valores obtidos no presente estudo são apresentados. Percebe-se, claramente, que o valor da massa ablacionada, massa ablacionada por energia e massa ablacionada por área, obtidas no presente trabalho, são maiores que os valores tabelados encontrados em [Burton e Turchi \(1998\)](#).

O calor recebido pela superfície do PTFE é diretamente proporcional ao quadrado da corrente (Efeito Joule), sendo assim, uma corrente com duração e amplitude maiores, como a obtida neste estudo e comparada com MACH2 na Tabela 5.1, gera uma quantidade de calor significativamente maior na superfície do PTFE. A consequência imediata desta maior fonte de calor, é que uma quantidade maior do PTFE atinge a temperatura de decomposição, desprendendo-se da barra de propelente e sendo ejetado do propulsor. Isto explica a maior massa ablacionada obtida no presente estudo.

Tabela 5.1 - Duração e amplitude das correntes obtidas com MACH2 e *rhoCentralFoam*.

	Duração da descarga	Amplitude máxima
MACH2	3 μs	6 kA
<i>rhoCentralFoam</i>	13 μs	9 kA

Fonte: Próprio Autor.

Tabela 5.2 - Tabela com dados experimentais.

Propulsor	E_0 , J	Tipo	I_{bit} , $\mu N - s$	I_{bit}/J , $\mu N - s/J$	Δm μg	$\Delta m/E_0$, $\mu g/J$	$\Delta m/\text{área}$, $\mu g/cm^2$
LES-6	1,85	Breech-Fed	26	14	8,88	4,8	3,3

Fonte: Adaptado de [Burton e Turchi \(1998\)](#)

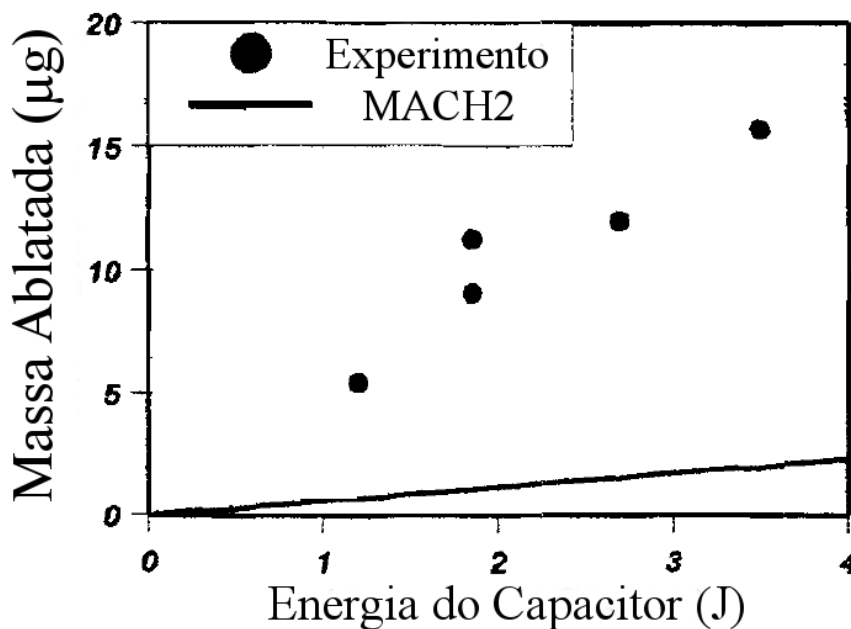
Tabela 5.3 - Tabela com dados simulados.

Propulsor	E_0 , J	Tipo	I_{bit} , $\mu N - s$	I_{bit}/J , $\mu N - s/J$	Δm μg	$\Delta m/E_0$, $\mu g/J$	$\Delta m/\text{área}$, $\mu g/cm^2$
LES-6	1,85	Breech-Fed	9,4	5,1	135	73	45

Fonte: Próprio Autor

Na Figuras 5.5 a massa ablatada, calculada pelo código MACH2, em função da energia do capacitor é mostrada. Nota-se que o código MACH2 neste caso subestima a quantidade de massa expelida, frente aos dados experimentais. Na Figura 5.6 a massa ablatada em função do tempo t é apresentada. Pode-se verificar que forma de onda da massa ablatada em função do tempo se assemelha, qualitativamente, àquela da Figura 5.9, a menos do pico por volta de $1 \mu s$, devido a consideração do fluxo de retorno de massa por parte do MACH2 (MIKELLIDES, 1999).

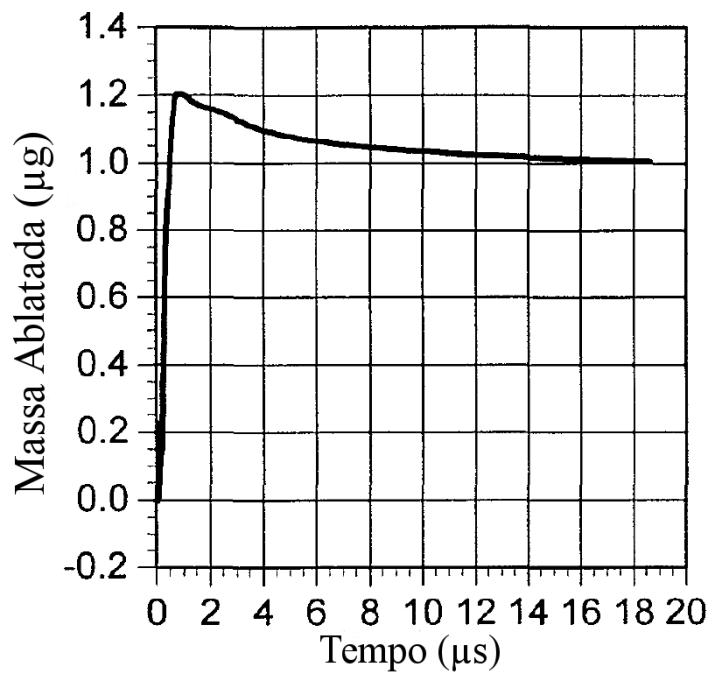
Figura 5.5 - Comparação entre a massa ablacionada experimental e predita pelo código MACH2.



Fonte: Adaptada de Mikellides (1999)

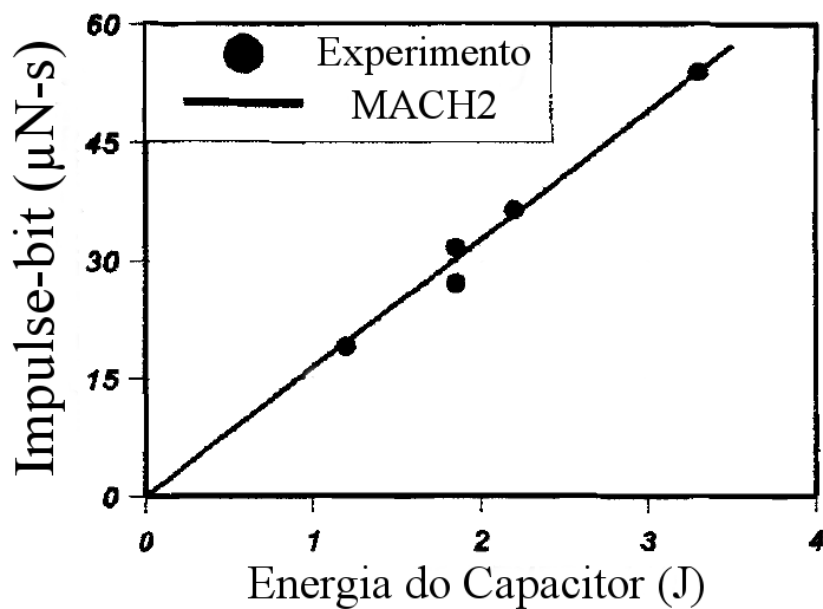
O *Impulse-bit* e *Impulse-bit* por energia da Tabela 5.2 são consideravelmente maiores que os valores obtidos através da simulação com OpenFOAM, apesar da massa expelida ser maior. Isto significa que velocidade de saída da massa nos experimentos é muito maior que a velocidade obtida neste estudo e apresentada na Figura 4.11. Este é um resultado esperado, uma vez que no presente estudo são desconsideradas quaisquer acelerações eletromagnéticas. Sabe-se que, apesar da baixa eficiência ($\sim 10\%$), a velocidade de saída do plasma no PPT é extremamente alta ($\sim 30 \text{ km/s}$) devida a aceleração eletromagnética (BURTON; TURCHI, 1998).

Figura 5.6 - Massa ablatada em função do tempo predita pelo código MACH2.



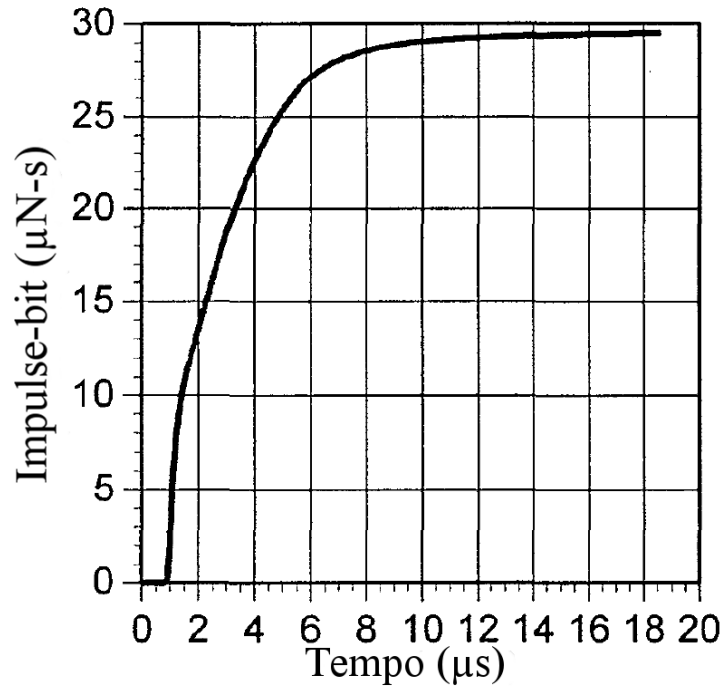
Fonte: Adaptada de Mikellides e Turchi (1996)

Figura 5.7 - Comparação entre o *impulse-bit* experimental e predito pelo código MACH2.



Fonte: Adaptada de Mikellides (1999)

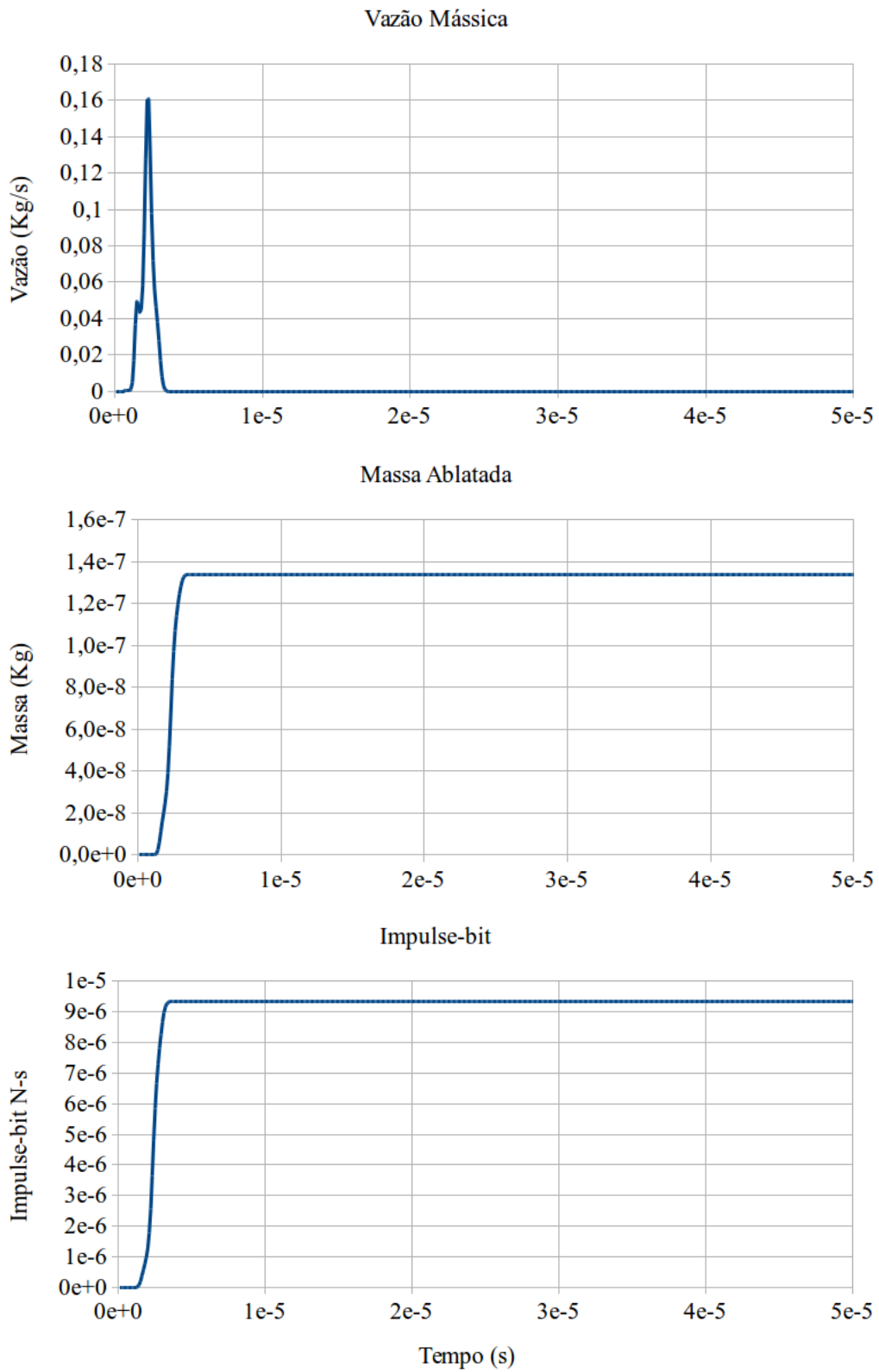
Figura 5.8 - *Impulse-bit* em função do tempo predito pelo código MACH2.



Fonte: Adaptada de Mikellides e Turchi (1996)

Apesar disso, a forma de onda do *impulse-bit* encontrada neste trabalho e mostrada na Figura 5.9, está de acordo, qualitativamente, com a calculada pelo código MACH2 e apresentada na Figura 5.8. Na Figura 5.7 é apresentado o *impulse-bit* em função da energia do capacitor.

Figura 5.9 - Gráficos das principais propriedades.



Fonte: Próprio Autor

6 CONCLUSÃO

Neste trabalho foi desenvolvido um código de simulação numérica, baseado em OpenFOAM, capaz de captar o processo termodinâmico básico do funcionamento de um propulsor de plasma pulsado. Foram feitas comparações com dados experimentais existentes na literatura, bem como dados de simulações baseadas no código MACH2.

O presente trabalho desconsiderou qualquer tipo de aceleração eletromagnética do propelente, obtendo-se assim um valor subestimado do *impulse-bit*. A massa ablacionada obtida, superestimou os valores experimentais encontrados na literatura. Como principal motivo desta discrepância, pode-se citar a amplitude e duração do pulso de corrente de descarga, que age como elemento de aquecimento do propelente.

Apesar de não considerar os aspectos eletromagnéticos, o código desenvolvido foi capaz de estimar qualitativamente a massa ablacionada e o *impulse-bit*, além de permitir uma análise da velocidade, temperatura de superfície, corrente de descarga, densidades e vazão mássica. O simulador desenvolvido é capaz de capturar os fenômenos termodinâmicos básicos do PPT, dando um passo inicial na simulação deste propulsor. Contudo, muitos aperfeiçoamentos ainda são necessários para permitir uma análise completa de todos os fenômenos ocorrendo no PPT e, conseqüentemente, seu uso no projeto de propulsores de plasma pulsado.

Entre as melhorias possíveis pode-se citar o cálculo das forças eletromagnéticas, reações químicas que podem ocorrer dentro do propulsor, modelos mais avançados do fenômeno da ablação e a implementação de métodos numéricos híbridos, capazes de capturar a pluma de exaustão do propulsor.

O problema da simulação básica do funcionamento de um PPT é bastante multidisciplinar, portanto bastante complexo. Existem diversos processos físicos ocorrendo dentro do PPT e o fato deste propulsor sempre trabalhar em regime transiente torna toda a modelagem mais difícil, pois simplificações que poderiam ser feitas em regime permanente não são possíveis.

Além dos processos físicos ocorrendo no PPT há, ainda, o problema de traduzir as equações governantes destes processos para o domínio computacional. Existem diversas técnicas computacionais para resolução de equações físicas. Cada conjunto de equações apresenta métodos mais ou menos adequados de resolução computacional. O problema de dinâmica de fluidos computacional faz uso intenso do método do volumes finitos (FVM, na sigla em inglês), devido à natureza conservativa destas

equações. Uma vez tendo o método computacional adequado, é necessário escolher entre os diversos *frameworks* e bibliotecas disponíveis para, só então, começar a implementação dos códigos de simulação dos diversos fenômenos ocorrendo no PPT.

A biblioteca OpenFOAM tem mais de 20 anos de existência e mais de 1 milhão de linhas de código ao todo. O aprendizado desta poderosa ferramenta demandou um grande esforço e tempo. O simulador desenvolvido no contexto desta biblioteca tem diversas limitações, contudo, consegue reproduzir alguns fenômenos observados no PPT. Este é um primeiro passo rumo ao desenvolvimento de um simulador aprimorado, capaz de modelar comportamentos mais complexos e de forma mais robusta.

REFERÊNCIAS BIBLIOGRÁFICAS

- ANDERSON, J. **Computational fluid dynamics**. New York: McGraw-Hill, 1995. 547 p. ISBN 9780070016859. 23, 24
- ANISIMOV, S. I. Vaporization of metal absorbing laser radiation. **Soviet Physics JETP**, v. 27, n. 1, p. 182–183, 1968. ISSN 1063-7761. 15, 16, 17, 18
- BEILIS, I. I. Theoretical modeling of cathode spot phenomena. In: BOXMAN, R. L.; SANDERS, D. M.; MARTIN, P. J. (Ed.). **Handbook of vacuum arc science & technology**. Park Ridge: William Andrew, 1996. p. 208–256. 18
- BERKERY, J.; CHOUEIRI, E. Characterization of current sheet evolution in a pulsed electromagnetic accelerator. In: INTERNATIONAL ELECTRIC PROPULSION CONFERENCE, 28., 2003, Toulouse, France. **Proceedings...** Toulouse, 2003. p. 1–12. 13
- BONART, H. **Implementation and validation of a solver for direct numerical simulations of turbulent reacting flows in OpenFOAM**. 73 p. Tese (Doutorado em Combustão) — Karlsruhe Institute of Technology, Karlsruhe, 2012. 29, 30, 53, 55
- BURTON, R. L.; TURCHI, P. J. Pulsed plasma thruster. **Journal of Propulsion and Power**, v. 14, n. 5, p. 716–735, sep 1998. ISSN 0748-4658. 1, 3, 4, 5, 8, 9, 11, 12, 13, 15, 24, 87, 88
- CASSADY, R.; HOSKINS, W.; CAMPBELL, M.; RAYBURN, C. A micro pulsed plasma thruster (PPT) for the "Dawgstar" spacecraft. In: AEROSPACE CONFERENCE, 2000. **Proceedings...** [S.l.]: IEEE, 2000. p. 7–14. ISBN 0-7803-5846-5. ISSN 1095-323X. 5
- FERRONI, F. **Magneto-hydrodynamic simulations of liquid metal flows in fusion reactors**. London: [s.n.], 2012. 58 p. 38
- GATSONIS, N. A.; LU, Y.; BLANDINO, J.; DEMETRIOU, M. A.; PASCHALIDIS, N. Micropulsed plasma thrusters for attitude control of a low-earth-orbiting CubeSat. **Journal of Spacecraft and Rockets**, v. 53, n. 1, p. 57–73, jan 2016. ISSN 0022-4650. 2
- GOEBEL, D. M.; KATZ, I. **Fundamentals of electric propulsion**. Hoboken, NJ, USA: John Wiley & Sons, 2008. 486 p. ISBN 9780470436448. 2

- GOMBOSI, T. I. **Gaskinetic theory**. New York: Cambridge University Press, 1994. 312 p. ISBN 978-0521439664. 16
- GREENSHIELDS, C. J. **OpenFOAM programmer's guide**. [S.l.: s.n.], 2015. 100 p. 36, 37
- GREENSHIELDS, C. J.; WELLER, H. G.; GASPARINI, L.; REESE, J. M. Implementation of semi-discrete, non-staggered central schemes in a colocated, polyhedral, finite volume framework, for high-speed viscous flow. **International Journal for Numerical Methods in Fluids**, v. 14, n. 14, p. 4427–36, jul 2008. ISSN 1078-0432. 26, 27, 29, 30, 31, 32, 39, 40, 41, 42, 54
- GUIDE, U. **OpenFoam - user guide**. [S.l.: s.n.], 2017. 158 p. 33, 34, 35, 36, 39, 42, 57
- GUMAN, W.; VONDRA, R. J.; THOMASSEN, K. Pulsed plasma propulsion system studies. In: ELECTRIC PROPULSION CONFERENCE, 1970. **Proceedings...** [S.l.]: AIAA, 1970. v. 8. 13
- HUANG, Z. **OpenFOAM simulation for electromagnetic problems**. 1–54 p. Dissertação (Mestrado em Engenharia Elétrica) — Chalmers University of Technology, Göteborg, Sweden, 2010. 34, 35, 36, 38
- JAHN, R. G.; LYMAN, F. A. Physics of electric propulsion. **Journal of Applied Mechanics**, v. 36, n. 3, p. 655, 1969. ISSN 00218936. 1, 2, 6
- JASAK, H.; JEMCOV, A.; TUKOVIC, Z. OpenFOAM : A C++ library for complex physics simulations. In: INTERNATIONAL WORKSHOP ON COUPLED METHODS IN NUMERICAL DYNAMICS, 2007. **Proceedings...** [S.l.], 2007. p. 1–20. 37, 38
- KEIDAR, M.; BOYD, I.; BEILIS, I. Electrical discharge in the Teflon cavity of a coaxial pulsed plasma thruster. **IEEE Transactions on Plasma Science**, v. 28, n. 2, p. 376–385, 2000. ISSN 00933813. 15
- KEIDAR, M.; BOYD, I. D.; BEILIS, I. I. On the model of Teflon ablation in an ablation-controlled discharge. **Journal of Physics D: Applied Physics**, v. 34, n. 11, p. 1675–1677, jun 2001. ISSN 0022-3727. 15, 18, 19, 20
- _____. Ionization and ablation phenomena in an ablative plasma accelerator. **Journal of Applied Physics**, v. 96, n. 10, p. 5420–5428, nov 2004. ISSN 0021-8979. 19, 20

KEIDAR, M.; FAN, J.; BOYD, I. D.; BEILIS, I. I. Vaporization of heated materials into discharge plasmas. **Journal of Applied Physics**, v. 89, n. 6, p. 3095–3098, mar 2001. ISSN 0021-8979. 15, 16, 18, 79

KNIGHT, C. J. Theoretical modeling of rapid surface vaporization with back pressure. **AIAA Journal**, v. 17, n. 5, p. 519–523, may 1979. ISSN 0001-1452. Disponível em: <<http://arc.aiaa.org/doi/10.2514/3.61164>>. 22

KURGANOV, A.; NOELLE, S.; PETROVA, G. Semi discrete central-upwind schemes for hyperbolic conservation laws and Hamilton-Jacobi equations. **SIAM Journal of Scientific Computing**, v. 23, n. 3, p. 707–740, 2001. Disponível em: <<https://doi.org/10.1137/S1064827500373413>>. 39

KURGANOV, A.; TADMOR, E. New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations. **Journal of Computational Physics**, v. 160, n. 1, p. 241–282, 2000. ISSN 00219991. Disponível em: <<http://linkinghub.elsevier.com/retrieve/pii/S0021999100964593>>. 39

LANGMUIR, I. The vapor pressure of metallic tungsten. **Physical Review**, v. 2, n. 5, p. 329–342, nov 1913. ISSN 0031-899X. 15

LEI, Y.; YUPING, H.; TANG, H.; LIU, X. Ablation and ionization phenomenon in a teflon pulsed plasma thruster. In: INTERNATIONAL SYMPOSIUM ON SPACE TECHNOLOGY AND SCIENCE, 30.; INTERANTAIIONAL ELECTRIC PROPULSION CONFERENCE, 34.; NANO-SATELLITE SYMPOSIUM, 6., 2015. **Proceedings...** [S.l.], 2015. p. 1–21. 85

LI, Y.; ZHANG, R.; ZONG, J. H. Study on the plume characteristics of pulsed plasma thruster. **Applied Mechanics and Materials**, v. 347-350, p. 55–58, aug 2003. ISSN 1662-7482. 13

MARIN, L. F. C. **Análise do desempenho de um propulsor a plasma pulsado de dupla descarga através da variação da distribuição de energia entre os seus dois estágios**. 133 p. Dissertação (Mestrado em Engenharia e Tecnologia Espaciais / Combustão a Propulsão) — Instituto Nacional de Pesquisas Espaciais, 2014. 2, 4, 5, 6, 8

MARQUES, R. I. **A mechanism to accelerate the late ablation in pulsed plasma thrusters**. 193 p. Tese (Doutorado em Engenharia) — University of Southampton, Southampton, 2009. 3, 4, 5, 6, 7, 8, 9, 10, 11

MIKELLIDES, P.; TURCHI, P. Modeling of late-time ablation in Teflon pulsed plasma thrusters. In: JOINT PROPULSION CONFERENCE AND EXHIBIT, 32., 1996. **Proceedings...** AIAA, 1996. p. 2733. ISBN 9780000000002. Disponível em: <<http://arc.aiaa.org/doi/10.2514/6.1996-2733>>. 89, 90

MIKELLIDES, Y. **Theoretical modeling and optimization of ablation-fed pulsed plasma thruster**. Tese (Doutorado) — The Ohio State University, Columbus, 1999. 15, 20, 24, 52, 83, 85, 86, 88, 89

MOELLER, T.; CHANG, Y.-K. MACH2 Simulations of a micro laser ablation plasma thruster. **Aerospace Science and Technology**, v. 11, n. 6, p. 481–489, sep 2007. ISSN 12709638. 24

MOUKALLED, F.; MANGANI, L.; DARWISH, M. **The finite volume method in computational fluid dynamics**. Cham: Springer International Publishing, 2016. 817 p. (Fluid Mechanics and Its Applications, v. 113). ISSN 0926-5112. ISBN 978-3-319-16873-9. Disponível em: <<http://www.scopus.com/inward/record.url?eid=2-s2.0-84939129919&partnerID=tZ0tx3y1http://link.springer.com/10.1007/978-3-319-16874-6>>. 25, 26, 27, 28, 29, 30

POPOV, G.; ORLOV, M.; ANTROPOV, N.; GOMILKA, L.; DIAKONOV, G.; KRIVONOSOV, I. Parameters of plasmoids injected by PPT. In: JOINT PROPULSION CONFERENCE AND EXHIBIT, 33., 1997. **Proceedings**. [S.l.]: AIAA, 1997. 13

REZENDE, J. M. D. Linguagem médica e marcapasso. *Jornal Brasileiro de Arritmias Cardíacas*, v. 8, n. 2, p. 69–70, 1995. 17

ROGHAIR, I.; ENDE, D. V. D.; MUGELE, F. An OpenFOAM-based electro-hydrodynamical model. In: INTERNATIONAL CONFERENCE ON MULTIPHASE FLOW, 8., 2013. **Proceedings...** [S.l.], 2013. p. 1–5. 38

SASS-TISOVSKAYA, M. **Plasma arc-welding simulation with OpenFOAM**. 85 p. Dissertação (Mestrado em Dinâmica dos Fluidos) — Chalmers University of Technology, Göteborg, Sweden, 2009. 38

SELSTROM, J. J. **Thrust and performance study of micro pulsed plasma thruster**. 148 p. Dissertação (Mestrado em Engenharia Aeronáutica) — Air Force Institute of Technology, Dayton, Ohio, 2010. 10

SUTTON, G. P.; BIBLARZ, O. **Rocket propulsion elements**. 7. ed. Cambridge: Cambridge University Press, 2001. 1–764 p. ISSN 1098-6596. ISBN 0471326429. 1, 2, 3

TURCHI, P.; MIKELLIDES, P. Modeling of ablation-fed pulsed plasma thrusters. In: JOINT PROPULSION CONFERENCE AND EXHIBIT, 31., 1995. **Proceedings**. [S.l.]: AIAA, 1995. ISBN 9780000000002. 14, 25, 47, 83, 84

ZAGHLOUL, M. R. On the vaporization of Teflon and heated compound-materials in ablation-controlled arcs. **Journal of Applied Physics**, v. 95, n. 7, p. 3339–3343, apr 2004. ISSN 0021-8979. 15, 20, 21, 22, 23, 79

APÊNDICE A - DICIONÁRIOS DE CONFIGURAÇÃO

Esse apêndice lista os dicionários de configuração do código *rhoCentralFoam*. Os arquivos são agrupados e apresentados de acordo com a pasta que os contêm.

A.1 Diretório *constant*

A.1.1 *thermophysicalProperties*

```
/*-----*- C++ -*-----*\
| ===== |
| \\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \\ / O p e r a t i o n | Version: 4.1 |
| \\ / A n d | Web: www.OpenFOAM.org |
| \\ / M a n i p u l a t i o n |
\*-----*/
FoamFile
{
    version    2.0;
    format     ascii;
    class      dictionary;
    location   "constant";
    object     thermophysicalProperties;
}
// * * * * *

thermoType
{
    type        hePsiThermo;
    mixture     pureMixture;
    transport   const;
    thermo      hConst;
    equationOfState perfectGas;
    specie      specie;
    energy      sensibleInternalEnergy;
}

mixture
{
```

```

specie
{
    nMoles      1;
    molWeight   100;
}
thermodynamics
{
    Cp          1004.5;
    Hf          0;
}
transport
{
    mu          1.8e-05;
    Pr          0.7;
}
}

```

// ***** //

A.1.2 *transportProperties*

```

/*-----* C++ *-----*\
| ===== |
| \\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox |
| \\ / O p e r a t i o n | Version: 4.1 |
| \\ / A n d | Web: www.OpenFOAM.org |
| \\ / M a n i p u l a t i o n |
\*-----*/

```

```

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "constant";
    object       transportProperties;
}

```

// ***** //

```

DT          DT [0 2 -1 0 0 0 0] 1.28e-7; //Difusividade térmica

cp          cp [0 2 -2 -1 0 0 0] 1250;

rho         rho [1 -3 0 0 0 0 0] 2150;

T_Ablation  T_Ablation [0 0 0 1 0 0 0] 650;

m_0         m_0 [1 0 0 0 0 0 0] 1.66e-25;

m_1         m_1 [1 0 0 0 0 0 0] 1.66e-25;

m_2         m_2 [1 0 0 0 0 0 0] 1.66e-25;

L L [1 2 -2 0 0 -2 0] 3.4e-8;

R R [1 2 -3 0 0 -2 0] 3e-2;

C C [-1 -2 4 0 0 2 0] 2e-6;

```

```
// ***** //
```

A.1.3 *turbulenceProperties*

```

/*-----*- C++ -*-----*\
| ===== |
| \\      / F ield      | OpenFOAM: The Open Source CFD Toolbox |
| \\      / O peration  | Version: 4.1 |
|  \\    / A nd         | Web:      www.OpenFOAM.org |
|   \\/   M anipulation | |
\*-----*/
FoamFile
{
    version 2.0;

```

```

format      ascii;
class       dictionary;
location    "constant";
object      turbulenceProperties;
}
// * * * * *

```

```
simulationType laminar;
```

```
// ***** //
```

A.2 Diretório *system*

A.2.1 *blockMeshDict*

```

/*-----*- C++ -*-----*\
| ===== |
| \\      / F ield      | OpenFOAM: The Open Source CFD Toolbox |
| \\      / O peration  | Version: 4.1 |
|  \\    / A nd         | Web:      www.OpenFOAM.org |
|   \\/   M anipulation | |
\*-----*/

```

```

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       blockMeshDict;
}
// * * * * *

```

```
convertToMeters 0.01;
```

```

vertices
(
    (0 0 0)
    (0.01 0 0)

```



```

(0.01 3 0)
(0 3 0)
(0 0 1)
(0.01 0 1)
(0.01 3 1)
(0 3 1)

(0.01 0 0)
(2 0 0)
(2 3 0)
(0.01 3 0)
(0.01 0 1)
(2 0 1)
(2 3 1)
(0.01 3 1)
);

blocks
(
  hex (0 1 2 3 4 5 6 7) teflon (40 15 1) simpleGrading (1 1 1)
  hex (8 9 10 11 12 13 14 15) plasma (50 15 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
(
  top_teflon
  {
    type patch;
    faces
    ( (3 7 6 2) );
  }
  left
  {

```

```

    type patch;
    faces
      ( (0 4 7 3) );
  }
bottom_teflon
{
  type patch;
  faces
    ( (1 5 4 0) );
}

teflon_to_plasma
{
  type patch;
  faces
    ( (1 2 6 5) );
}

plasma_to_teflon
{
  type patch;
  faces
    ( (8 12 15 11) );
}

top_plasma
{
  type patch;
  faces
    (
      (11 15 14 10)
    );
}
right
{
  type patch;
  faces

```

```

        (
            (10 14 13 9)
        );
    }
    bottom_plasma
    {
        type patch;
        faces
        ( (9 13 12 8) );
    }

    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)

            (8 11 10 9)
            (12 13 14 15)
        );
    }
);

mergePatchPairs
(
    (teflon_to_plasma plasma_to_teflon)
);

// ***** //

```


APÊNDICE B - CONDIÇÃO DE CONTORNO DE ABLAÇÃO

Esse apêndice lista os códigos fonte da condição de contorno de ablação, chamada *Ablation*.

B.1 Condição de Contorno de Ablação

B.1.1 *AblationFvPatchField.C*

```
/*-----*\
===== |
\\      / F i e l d      | OpenFOAM: The Open Source CFD Toolbox
\\      / O p e r a t i o n      |
\\      / A n d      | Copyright (C) 2011-2016 OpenFOAM Foundation
  \\/      M a n i p u l a t i o n      |
-----\

License

This file is part of OpenFOAM.

OpenFOAM is free software: you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License
for more details.

You should have received a copy of the GNU General Public License
along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

\*-----*/

#include "AblationFvPatchField.H"

// * * * * * Constructors * * * * * //

template<class Type>
```

```

Foam::AblationFvPatchField<Type>::AblationFvPatchField
(
    const fvPatch& p,
    const DimensionedField<Type, volMesh>& iF
)
:
    mixedFvPatchField<Type>(p, iF),
    CurrentName_("I"),
    Gamma(1),
    Cp(1250),
    k(0.344),
    DeltaH(1.610e6),
    T_0(300)
{
    this->refValue() = Zero;
    this->refGrad() = Zero;
    this->valueFraction() = 1.0;
}

```

```

template<class Type>
Foam::AblationFvPatchField<Type>::AblationFvPatchField
(
    const AblationFvPatchField<Type>& ptf,
    const fvPatch& p,
    const DimensionedField<Type, volMesh>& iF,
    const fvPatchFieldMapper& mapper
)
:
    mixedFvPatchField<Type>(ptf, p, iF, mapper),
    CurrentName_(ptf.CurrentName_),

    Gamma(ptf.Gamma),
    Cp(ptf.Cp),
    k(ptf.k),
    DeltaH(ptf.DeltaH),
    T_0(ptf.T_0)

```

```
{}
```

```
template<class Type>
Foam::AblationFvPatchField<Type>::AblationFvPatchField
(
    const fvPatch& p,
    const DimensionedField<Type, volMesh>& iF,
    const dictionary& dict
)
:
    mixedFvPatchField<Type>(p, iF),
    CurrentName_(dict.lookupOrDefault<word>("Current", "I")),

    Gamma(readScalar(dict.lookup("Gamma"))),
    Cp(readScalar(dict.lookup("Cp"))),
    k(readScalar(dict.lookup("k"))),
    DeltaH(readScalar(dict.lookup("DeltaH"))),
    T_0(readScalar(dict.lookup("T_0")))
{
    this->refValue() = Field<Type>(this->size(), pTraits<Type>::one)*(T_0);
    this->valueFraction() = Cp*Gamma/(Cp*Gamma + k*this->patch().deltaCoeffs());
    this->refGrad() = Zero;

    fvPatchField<Type>::operator=( this->valueFraction()*this->refValue()
    + (1 - this->valueFraction())*this->patchInternalField() );
}

```

```
template<class Type>
Foam::AblationFvPatchField<Type>::AblationFvPatchField
(
    const AblationFvPatchField<Type>& ptf
)
:
    mixedFvPatchField<Type>(ptf),
    CurrentName_(ptf.CurrentName_),

```

```

    Gamma(ptf.Gamma),
    Cp(ptf.Cp),
    k(ptf.k),
    DeltaH(ptf.DeltaH),
    T_0(ptf.T_0)
{}

```

```

template<class Type>
Foam::AblationFvPatchField<Type>::AblationFvPatchField
(
    const AblationFvPatchField<Type>& ptf,
    const DimensionedField<Type, volMesh>& iF
)
:
    mixedFvPatchField<Type>(ptf, iF),
    CurrentName_(ptf.CurrentName_),

    Gamma(ptf.Gamma),
    Cp(ptf.Cp),
    k(ptf.k),
    DeltaH(ptf.DeltaH),
    T_0(ptf.T_0)
{}

```

```

// * * * * * Member Functions * * * * * //

```

```

template<class Type>
void Foam::AblationFvPatchField<Type>::updateCoeffs()
{
    if (this->updated())
    {
        return;
    }
}

```



```

//Discover label of "plasma_to_teflon" boundary
    scalarField plsamaMesh = \
this->db().parent().objectRegistry::lookupObject<fvMesh>("plasma")
    const label patchID_plasma = \
plsamaMesh.boundaryMesh().findPatchID("plasma_to_teflon");

//Read electric current value based on CurrentName_
    const fvPatchField<scalar>& Curr = \
this->patch().template lookupPatchField<volScalarField, scalar>(CurrentName_);

//Calculate total area of (this) boundary
    const scalarField area = this->patch().magSf();
    scalar areaTotal = gSum(area);

//Calculate heat generate by current
    Field<scalar> q(pow(Curr,2)*0.03/areaTotal);

//Read rhoU field from plasma region entrance
    const fvPatchField<vector>& rhoU_patch = \
this->db().parent().subRegistry("plasma") \
.objectRegistry::lookupObject<volVectorField>("rhoU") \
.boundaryField()[patchID_plasma];

//Read transportProperties dictionary
    const dictionary& transport_Properties = this->db() \
.objectRegistry::lookupObject<IOdictionary>("transportProperties");

//Read ablation temperature(T_Ablation) from transportProperties dictionary
    const dimensionedScalar T_Ablation(transport_Properties.lookup("T_Ablation"));

    scalar Ts = \
this->patch().template lookupPatchField<volScalarField, scalar>("T0") \
.patchInternalField()()[0];

    scalarField Ga = Gamma*max(mag(rhoU_patch),0.01/Gamma);

```

```

Info << "Ga: " << Ga[0] << nl << endl;

    if(Ts >= T_Ablation.value() )
    {
this->refValue() = Field<Type>(this->size(), pTraits<Type>::one) \
*(T_0 - DeltaH/Cp + q/Cp/Ga);

this->refGrad() = Zero;

this->valueFraction() = Cp*Ga/(Cp*Ga + k*this->patch().deltaCoeffs());

Info <<"DeltaCoeffs: " << this->patch().deltaCoeffs() << nl << endl;
    }
    else
    {
Info <<"Calor q/k: " << (q/k)()[0] << nl << endl;

Info <<"Ts: " << Ts << nl << endl;

this->refValue() = Zero;

this->refGrad() = Field<Type>(this->size(), pTraits<Type>::one)*(q/k);

this->valueFraction() = 0.0;
    }

    mixedFvPatchField<Type>::updateCoeffs();
}

template<class Type>
void Foam::AblationFvPatchField<Type>::write(Ostream& os) const
{
    fvPatchField<Type>::write(os);

    os.writeKeyword("Current") << CurrentName_ << token::END_STATEMENT << nl;

```

```

os.writeKeyword("Gamma") << Gamma << token::END_STATEMENT <<nl;
os.writeKeyword("Cp") << Cp << token::END_STATEMENT <<nl;
os.writeKeyword("k") << k << token::END_STATEMENT <<nl;
os.writeKeyword("DeltaH") << DeltaH << token::END_STATEMENT <<nl;
os.writeKeyword("T_0") << T_0 << token::END_STATEMENT <<nl;
}

// * * * * * Member Operators * * * * * //

template<class Type>
void Foam::AblationFvPatchField<Type>::operator=
(
    const fvPatchField<Type>& ptf
)
{
    fvPatchField<Type>::operator=
    (
        this->valueFraction()*this->refValue()
        + (1 - this->valueFraction())*ptf
    );
}

// ***** //

```

B.1.2 *AblationFvPatchField.H*

```

/*-----*\
===== |
\\      / F i e l d      | OpenFOAM: The Open Source CFD Toolbox
\\      / O p e r a t i o n |
\\      / A n d           | Copyright (C) 2011-2016 OpenFOAM Foundation
  \\    / M a n i p u l a t i o n |
-----*/

```

License

This file is part of OpenFOAM.

OpenFOAM is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

Class

Foam::AblationFvPatchField

Group

grpOutletBoundaryConditions

Description

This boundary condition provides a generic outflow condition, with specified inflow for the case of return flow.

Usage

```
\table
  Property      | Description                | Required  | Default value
  phi           | Flux field name           | no       | phi
  inletValue    | Inlet value for reverse flow | yes      |
\endtable
```

Example of the boundary condition specification:

```
\verbatim
<patchName>
{
    type            Ablation;
    phi             phi;
    inletValue      uniform 0;
```

```

        value            uniform 0;
    }
\endverbatim

```

The mode of operation is determined by the sign of the flux across the patch faces.

Note

Sign conventions:

- Positive flux (out of domain): apply zero-gradient condition
- Negative flux (into of domain): apply the "inletValue" fixed-value

See also

```

Foam::mixedFvPatchField
Foam::zeroGradientFvPatchField
Foam::outletInletFvPatchField

```

SourceFiles

```

AblationFvPatchField.C

```

```

\*-----*/

```

```

#ifndef AblationFvPatchField_H
#define AblationFvPatchField_H

```

```

#include "mixedFvPatchField.H"

```

```

// * * * * *

```

```

namespace Foam
{

```

```

\*-----\

```

```

        Class AblationFvPatchField Declaration

```

```

\*-----*/

```

```

template<class Type>

```

```

class AblationFvPatchField
:
    public mixedFvPatchField<Type>
{

protected:

    // Protected data

        //- Name of flux field
        word CurrentName_;

scalar Gamma;
scalar Cp;
scalar k;
scalar DeltaH;
scalar T_0;

public:

    //- Runtime type information
    TypeName("Ablation");

    // Constructors

        //- Construct from patch and internal field
        AblationFvPatchField
        (
            const fvPatch&,
            const DimensionedField<Type, volMesh>&
        );

        //- Construct from patch, internal field and dictionary
        AblationFvPatchField
        (
            const fvPatch&,

```

```

        const DimensionedField<Type, volMesh>&,
        const dictionary&
    );

    //- Construct by mapping given AblationFvPatchField onto a new patch
    AblationFvPatchField
    (
        const AblationFvPatchField<Type>&,
        const fvPatch&,
        const DimensionedField<Type, volMesh>&,
        const fvPatchFieldMapper&
    );

    //- Construct as copy
    AblationFvPatchField
    (
        const AblationFvPatchField<Type>&
    );

    //- Construct and return a clone
    virtual tmp<fvPatchField<Type>> clone() const
    {
        return tmp<fvPatchField<Type>>
        (
            new AblationFvPatchField<Type>(*this)
        );
    }

    //- Construct as copy setting internal field reference
    AblationFvPatchField
    (
        const AblationFvPatchField<Type>&,
        const DimensionedField<Type, volMesh>&
    );

    //- Construct and return a clone setting internal field reference
    virtual tmp<fvPatchField<Type>> clone

```

```

(
    const DimensionedField<Type, volMesh>& iF
) const
{
    return tmp<fvPatchField<Type>>
    (
        new AblationFvPatchField<Type>(*this, iF)
    );
}

// Member functions

// Attributes

//- Return true: this patch field is altered by assignment
virtual bool assignable() const
{
    return true;
}

//- Update the coefficients associated with the patch field
virtual void updateCoeffs();

//- Write
virtual void write(Ostream&) const;

// Member operators

virtual void operator=(const fvPatchField<Type>& pvf);
};

// * * * * * //

```



```

} // End namespace Foam

// * * * * *

#ifdef NoRepository
    #include "AblationFvPatchField.C"
#endif

// * * * * *

#endif

// *****

```

B.1.3 *AblationFvPatchFields.C*

```

/*-----*\
===== |
\\      / F i e l d      | OpenFOAM: The Open Source CFD Toolbox
\\      / O p e r a t i o n |
\\      / A n d           | Copyright (C) 2011 OpenFOAM Foundation
  \\/      M a n i p u l a t i o n |
-----*/

```

License

This file is part of OpenFOAM.

OpenFOAM is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

```

\*-----*/

#include "volFields.H"
#include "surfaceFields.H"
#include "AblationFvPatchFields.H"
#include "addToRunTimeSelectionTable.H"

// * * * * *

namespace Foam
{

// * * * * * Static Data Members * * * * *

makePatchFields(Ablation);

// * * * * *

} // End namespace Foam

// *****

```

B.1.4 *AblationFvPatchFields.H*

```

\*-----*\
===== |
\\      /  F i e l d      | OpenFOAM: The Open Source CFD Toolbox
\\      /  O p e r a t i o n |
  \\    /   A n d          | Copyright (C) 2011 OpenFOAM Foundation
    \\ /    M a n i p u l a t i o n |
-----

```

License

This file is part of OpenFOAM.

OpenFOAM is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or

(at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

```
\*-----*/

#ifdef AblationFvPatchFields_H
#define AblationFvPatchFields_H

#include "AblationFvPatchField.H"
#include "fieldTypes.H"

// * * * * * //

namespace Foam
{

// * * * * * //

makePatchTypeFieldTypedefs(Ablation);

// * * * * * //

} // End namespace Foam

// * * * * * //

#endif

// ***** //
```

B.1.5 *AblationFvPatchFieldsFwd.H*

```
/*-----*\
===== |
\\      /  F ield      | OpenFOAM: The Open Source CFD Toolbox
\\      /  O peration  |
\\      /  A nd        | Copyright (C) 2011 OpenFOAM Foundation
  \\    /  M anipulation |
-----*/
```

License

This file is part of OpenFOAM.

OpenFOAM is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

```
\*-----*/
```

```
#ifndef AblationFvPatchFieldsFwd_H
```

```
#define AblationFvPatchFieldsFwd_H
```

```
#include "fieldTypes.H"
```

```
// * * * * * //
```

```
namespace Foam
```

```
{
```

```
// * * * * * //
```


APÊNDICE C - PPT_FOAM

Esse apêndice lista os códigos fonte do PPT_Foam.

C.1 Código Base do PPT_Foam

C.1.1 *directionInterpolate.H*

```
namespace Foam
{

//- Interpolate field vf according to direction dir
template<class Type>
tmp<GeometricField<Type, fvsPatchField, surfaceMesh>> interpolate
(
    const GeometricField<Type, fvPatchField, volMesh>& vf,
    const surfaceScalarField& dir,
    const word& reconFieldName = word::null
)
{
    tmp<GeometricField<Type, fvsPatchField, surfaceMesh>> tsf
    (
        fvc::interpolate
        (
            vf,
            dir,
            "reconstruct("
            + (reconFieldName != word::null ? reconFieldName : vf.name())
            + ')'
        )
    );

    GeometricField<Type, fvsPatchField, surfaceMesh>& sf = tsf.ref();

    sf.rename(vf.name() + '_' + dir.name());

    return tsf;
}
```

```
}
```

C.1.2 *createFluidMesh.H*

```
Foam::Info
    << "Create plasma mesh for time = "
    << runTime.timeName() << Foam::nl << Foam::endl;

Foam::fvMesh mesh
(
    Foam::IOobject
    (
        "plasma",
        runTime.timeName(),
        runTime,
        Foam::IOobject::MUST_READ
    )
);
```

C.1.3 *createSolidMesh.H*

```
Foam::Info
    << "Create teflon mesh for time = "
    << runTime.timeName() << Foam::nl << Foam::endl;

Foam::fvMesh solidMesh
(
    Foam::IOobject
    (
        "teflon",
        runTime.timeName(),
        runTime,
        Foam::IOobject::MUST_READ
    )
);
```

C.1.4 *createFields.H*

```
Info<< "Reading thermophysical properties\n" << endl;
```



```

autoPtr<psiThermo> pThermo
(
    psiThermo::New(mesh)
);
psiThermo& thermo = pThermo();

volScalarField& e = thermo.he();

Info<< "Reading field Mass\n" << endl;
volScalarField Mp
(
    IOobject
    (
        "Mp",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field U\n" << endl;
volVectorField U
(
    IOobject
    (
        "U",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

```

```

volScalarField rho
(
    IOobject
    (
        "rho",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    thermo.rho()
);

```

```
rho.write();
```

```

volVectorField rhoU
(
    IOobject
    (
        "rhoU",
        runTime.timeName(),
        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    rho*U
);

```

```
rhoU.write();
```

```

volScalarField rhoE
(
    IOobject
    (
        "rhoE",
        runTime.timeName(),

```

```

        mesh,
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
    ),
    rho*(e + 0.5*magSqr(U))
);

rhoE.write();

surfaceScalarField pos
(
    IOobject
    (
        "pos",
        runTime.timeName(),
        mesh
    ),
    mesh,
    dimensionedScalar("pos", dimless, 1.0)
);

surfaceScalarField neg
(
    IOobject
    (
        "neg",
        runTime.timeName(),
        mesh
    ),
    mesh,
    dimensionedScalar("neg", dimless, -1.0)
);

surfaceScalarField phi("phi", fvc::flux(rhoU));

Info<< "Creating turbulence model\n" << endl;

```

```

autoPtr<compressible::turbulenceModel> turbulence
(
    compressible::turbulenceModel::New
    (
        rho,
        U,
        phi,
        thermo
    )
);

```

C.1.5 *createSolidFields.H*

```

Info<< "Creating field Current\n" << endl;
volScalarField I
(
    IOobject
    (
        "I",
        runTime.timeName(),
        solidMesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    solidMesh
);

I.write();

Info<< "Reading field T0\n" << endl;
volScalarField T0
(
    IOobject
    (
        "T0",
        runTime.timeName(),
        solidMesh,
        IOobject::MUST_READ,

```

```

        IOobject::AUTO_WRITE
    ),
    solidMesh
);

volScalarField T_tef(T0);

Info<< "Reading field Charge\n" << endl;
volScalarField Q
(
    IOobject
    (
        "Q",
        runTime.timeName(),
        solidMesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    solidMesh
);

Q.write();

Info<< "Reading transportProperties\n" << endl;

IOdictionary transportProperties
(
    IOobject
    (
        "transportProperties",
        runTime.constant(),
        solidMesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);

```

```

//Thermodynamics Properties

Info<< "Reading diffusivity DT\n" << endl;

dimensionedScalar DT
(
    transportProperties.lookup("DT")
);

dimensionedScalar cp
(
    transportProperties.lookup("cp")
);

dimensionedScalar Density
(
    transportProperties.lookup("rho")
);

dimensionedScalar T_Ablation
(
    transportProperties.lookup("T_Ablation")
);

dimensionedScalar Gamma
(
    transportProperties.lookup("Gamma")
);

dimensionedScalar Fator
(
    transportProperties.lookup("Fator")
);

dimensionedScalar m_0
(

```

```

        transportProperties.lookup("m_0")
    );

dimensionedScalar m_1
(
    transportProperties.lookup("m_1")
);

dimensionedScalar m_2
(
    transportProperties.lookup("m_2")
);

//Electric Properties

dimensionedScalar L
(
    transportProperties.lookup("L")
);

dimensionedScalar R
(
    transportProperties.lookup("R")
);

dimensionedScalar C
(
    transportProperties.lookup("C")
);

dimensionedScalar sigma
(
    transportProperties.lookup("sigma")
);

```

C.1.6 *createFieldRefs.H*

```
volScalarField& p = thermo.p();
volScalarField& T = thermo.T();
const volScalarField& psi = thermo.psi();
const volScalarField& mu = thermo.mu();
```

```
bool inviscid(true);
if (max(mu.primitiveField()) > 0.0)
{
    inviscid = false;
}
```

C.1.7 *readFluxScheme.H*

```
word fluxScheme("Kurganov");
if (mesh.schemesDict().readIfPresent("fluxScheme", fluxScheme))
{
    if ((fluxScheme == "Tadmor") || (fluxScheme == "Kurganov"))
    {
        Info<< "fluxScheme: " << fluxScheme << endl;
    }
    else
    {
        FatalErrorInFunction
            << "fluxScheme: " << fluxScheme
            << " is not a valid choice. "
            << "Options are: Tadmor, Kurganov"
            << abort(FatalError);
    }
}
}
```

C.1.8 *CreateAblationFields.H*

```
const label patchID_teflon = \
solidMesh.boundaryMesh().findPatchID("teflon_to_plasma");

if(patchID_teflon == -1)
{
```



```

FatalError << "patch not found on teflon!" << exit(FatalError);
}

const label patchID_plasma = \
mesh.boundaryMesh().findPatchID("plasma_to_teflon");

if(patchID_plasma == -1)
{
FatalError << "patch not found on plasma!" << exit(FatalError);
}

const label patchID_plasma_exit = mesh.boundaryMesh().findPatchID("right");

if(patchID_plasma_exit == -1)
{
FatalError << "patch not found on plasma right!" << exit(FatalError);
}

const scalar Kb(Foam::constant::physicoChemical::k.value());

const scalar pi(Foam::constant::mathematical::pi);

const scalar m0(m_0.value());

const scalar m1(m_1.value());

const scalar m2(m_2.value());

bool calcula_alpha = false;

const scalar gamma(Gamma.value());

scalar Area(sum(mesh.magSf().boundaryField()[patchID_plasma]));

scalar MassaAblada(0.0);

```

```

scalar Impulse_bit(0.0);

scalar A_V = 1;

scalarField alpha(T0.boundaryField()[patchID_teflon].size(), 0.0);

scalarField T_teflon(T0.boundaryField()[patchID_teflon].patchInternalField());

dimensionedScalar TotalEnergy(sum(mesh.V()*rhoE));

dimensionedScalar SolidEnergy( sum( (solidMesh.V()*(T0-T_tef())*Density*cp)));

scalarField n0(1.847e15*exp(-20815/T_teflon)/Kb/T_teflon);

scalarField p0(p.boundaryField()[patchID_teflon].patchInternalField());

scalarField T1( m1/m0*T_teflon*pow( sqrt(1 + pi*pow((gamma-1) \
*alpha/(gamma+1)/2, 2)) - Foam::sqrt(pi)*(gamma-1)*alpha/(gamma+1)/2 ,2) );

scalarField T2(T.boundaryField()[patchID_plasma_exit]);

scalarField n2(p.boundaryField()[patchID_plasma_exit]/Kb/T2);

scalarField T_exit(T.boundaryField()[patchID_plasma_exit]);

scalarField n_exit(rho.boundaryField()[patchID_plasma_exit]/m2);

```

```
scalarField p_exit(p.boundaryField()[patchID_plasma_exit]);
```

```
scalarField n1( (m0*n0)*sqrt(m1*T_teflon/m0/T1)*((pow(alpha,2) + 0.5) \
*exp(pow(alpha,2))*erfc(alpha) - alpha/Foam::sqrt(pi)) \
+0.5*m1*T_teflon/m0/T1*( 1 - Foam::sqrt(pi)*alpha*exp(pow(alpha,2))*erfc(alpha)));
```

```
scalarField beta( ((2*pow(alpha,2)+1) \
-alpha*sqrt(alpha)*sqrt(m1*T_teflon/m0/T1)) * exp(pow(alpha,2)) * (m0*n0)/n1 \
*sqrt(m1*T_teflon/m0/T1) );
```

C.1.9 *solveCurrent.H*

```
solve
(
    fvm::d2dt2(L,Q) + fvm::ddt(R,Q) + fvc::Sp(1/C,Q)
);

I = fvc::ddt(Q);
```

C.1.10 *solveHeatPropagation.H*

```
fvScalarMatrix T0Eqn
(
    fvm::ddt(T0)
    -fvm::laplacian(DT,T0)
);

T0Eqn.relax();
T0Eqn.solve();

SolidEnergy = sum( (solidMesh.V()*(T0-T_tef())*Density*cp) );

Info << "Energia do Teflon solido: " << SolidEnergy.value() << nl << endl;
```

C.1.11 *UpdateAblationFields.H*

```
Info << "Temperatura na saida: " << T_exit[0] << nl << endl;
T_exit = T.boundaryField()[patchID_plasma_exit];

Info << "Densidade numerica na saida: " << n_exit[0] << nl << endl;
n_exit = rho.boundaryField()[patchID_plasma_exit]/m2;

p_exit = p.boundaryField()[patchID_plasma_exit];

Info << "Pressao T2: " << T2[0] << nl << endl;
Info << "Pressao n2: " << n2[0] << nl << endl;

Info << "Pressao P0: " << (1.847e15*exp(-20815/T_teflon))()[0] << nl << endl;
Info << "Pressao P2: " << p_exit[0] << nl << endl;

    if(T0.boundaryField()[patchID_teflon].patchInternalField()[0]>= \
T_Ablation.value() || calcula_alpha)
{
calcula_alpha = true;

T1 = ( m1/m0*T_teflon*pow( Foam::sqrt(1 + pi*pow((gamma-1) \
*alpha/(gamma+1)/2, 2)) \
- Foam::sqrt(pi)*(gamma-1)*alpha/(gamma+1)/2 ,2) );

n1 = (m0*n0/m1)*sqrt(m1*T_teflon/m0/T1)*((pow(alpha,2) + 0.5) \
*exp(pow(alpha,2))*erfc(alpha) - alpha/Foam::sqrt(pi)) \
+0.5*m1*T_teflon/m0/T1 \
*( 1 - Foam::sqrt(pi)*alpha*exp(pow(alpha,2))*erfc(alpha) );

beta = ( (2*pow(alpha,2)+1) - alpha*Foam::sqrt(pi) \
*sqrt(m1*T_teflon/m0/T1) ) * exp(pow(alpha,2)) \
*(m0*n0)/m1/n1 * sqrt(m1*T_teflon/m0/T1);

U.boundaryFieldRef()[patchID_plasma] == \
vector( (alpha*sqrt(2*Kb*T1/m1))()[0], 0, 0);

thermo.T().boundaryFieldRef()[patchID_plasma] == T1;
```

```

count++;

alpha = scalarField( T1.size(), Fator.value() \
*Foam::exp(-pow(runTime.value()-6e-6,2)*1e11) \
+(runTime.value())>2e-5 ? 0.001:0 );
}

forAll(mesh.boundary()[patchID_plasma], faceI)
{
Info << "Area/Volume: " \
<< mesh.boundary()[patchID_plasma].magSf()[faceI] \
/ mesh.V()[mesh.boundary()[patchID_plasma].faceCells()[faceI]] \
<< nl << endl;

A_V = mesh.boundary()[patchID_plasma].magSf()[faceI] \
/ mesh.V()[mesh.boundary()[patchID_plasma].faceCells()[faceI]];

Mp[mesh.boundary()[patchID_plasma].faceCells()[faceI]] = \
(m1*n1*(alpha*sqrt(2*Kb*T1/m1))*A_V )()[0];
}

Info << "Area: " << Area << nl << endl;
Info << "Temperatura do plasma: " << T1[0] << nl << endl;
Info << "n1: " << n1[0] << nl << endl;
Info << "rho1: " << (m1*n1)()[0] << nl << endl;
Info << "alpha: " << alpha[0] << nl << endl;
Info << "alpha_calc: " << \
(((T2*n2/(T1*n1) - 1)/(1 - m1*n1/(m2*n2)))/2)()[0]>0 \
? sqrt( ( (T2*n2/(T1*n1) - 1)/(1 - m1*n1/(m2*n2)))/2 )()[0] : 0) \
<< nl << endl;
Info << "beta: " << beta[0] << nl << endl;
Info << "Pressao_P1: "<<p.boundaryFieldRef()[patchID_plasma][0] << nl << endl;
Info << "P1: " << (n1*Kb*T1)()[0] << nl << endl;

Info << "MassFlowRate: " << (m1*n1*(alpha*sqrt(2*Kb*T1/m1)))[0] << nl << endl;

```

```

Info << "MassaAblada: " << MassaAblada << nl << endl;
MassaAblada = MassaAblada \
+(m1*n1*(alpha*sqrt(2*Kb*T1/m1))*Area)()[0] \
*runTime.deltaT().value();

Info << "Impulse_bit: " << Impulse_bit << nl << endl;
Impulse_bit = Impulse_bit \
+(m1*n1*(alpha*sqrt(2*Kb*T1/m1))*Area \
*(alpha*sqrt(2*Kb*T1/m1)))()[0]*runTime.deltaT().value();

Info << "Temperatura do teflon: " << T_teflon[0] << nl << endl;
T_teflon = T0.boundaryField()[patchID_teflon].patchInternalField();
Info << "Densidade numerica n0: " << n0[0] << nl << endl;
n0 = (1.847e15*exp(-20815/T_teflon)/Kb/T_teflon);

Info << "V1: " << (alpha*sqrt(2*Kb*T1/m1))()[0] << nl << endl;
Info << "Vs: " << sqrt(gamma*Kb*T1/m1)()[0] << nl << endl;
Info << "Mach_v1: " << (alpha*sqrt(2*Kb*T1/m1)/(sqrt(gamma*Kb*T1/m1)))()[0] \
<< nl << endl;

//Calculate Total Energy on Plasma

TotalEnergy = sum(mesh.V()*rhoE);

Info << "Energia Total: " << TotalEnergy.value() << nl << endl;

```

C.1.12 *centralCourantNo.H*

```

/*-----*\
===== |
\\ / F ield | OpenFOAM: The Open Source CFD Toolbox
\\ / O peration |
\\ / A nd | Copyright (C) 2011-2016 OpenFOAM Foundation
\\ / M anipulation |

```

License

This file is part of OpenFOAM.

OpenFOAM is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

Global

centralCourantNo

Description

Calculates the mean and maximum wave speed based Courant Numbers.

-----/

```
if (mesh.nInternalFaces())
{
    scalarField sumAmaxSf(fvc::surfaceSum(amaxSf()).primitiveField());

    CoNum = 0.5*gMax(sumAmaxSf/mesh.V().field()*runTime.deltaTValue());

    meanCoNum =
        0.5*(gSum(sumAmaxSf)/gSum(mesh.V().field()))*runTime.deltaTValue();
}
```

```
Info<< "Mean and max Courant Numbers = "
<< meanCoNum << " " << CoNum << endl;
```

```
// ***** //
```

C.1.13 *setRDeltaT.H*

```
{
    volScalarField& rDeltaT = trDeltaT.ref();

    scalar rDeltaTSmoothingCoeff
    (
        runTime.controlDict().lookupOrDefault<scalar>
        (
            "rDeltaTSmoothingCoeff",
            0.02
        )
    );

    // Set the reciprocal time-step from the local Courant number
    rDeltaT.ref() = max
    (
        1/dimensionedScalar("maxDeltaT", dimTime, maxDeltaT),
        fvc::surfaceSum(amaxSf)()()
        /((2*maxCo)*mesh.V())
    );

    // Update the boundary values of the reciprocal time-step
    rDeltaT.correctBoundaryConditions();

    fvc::smooth(rDeltaT, rDeltaTSmoothingCoeff);

    Info<< "Flow time scale min/max = "
        << gMin(1/rDeltaT.primitiveField())
        << ", " << gMax(1/rDeltaT.primitiveField()) << endl;
}
```

C.1.14 *PPT_Foam.C*

```
/*-----*\
===== |
```



```
\\      /  F i e l d      | OpenFOAM: The Open Source CFD Toolbox
\\      /  O p e r a t i o n      |
\\      /  A n d      | Copyright (C) 2011-2016 OpenFOAM Foundation
  \\ /  M a n i p u l a t i o n      |
```

License

This file is part of OpenFOAM.

OpenFOAM is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

OpenFOAM is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with OpenFOAM. If not, see <http://www.gnu.org/licenses/>.

Application

PPT_Foam

Description

Codigo PPT_Foam baseado no codigo rhoCentralFoam.

-----/

```
#include "fvCFD.H"
#include "psiThermo.H"
#include "turbulentFluidThermoModel.H"
#include "fixedRhoFvPatchScalarField.H"
#include "directionInterpolate.H"
#include "localEulerDdtScheme.H"
#include "fvcSmooth.H"
```



```

    while (runTime.run())
    {
//Solve RLC circuit
#include "solveCurrent.H"

//Solve Heat Propagation on Solid
#include "solveHeatPropagation.H"

//Update the Ablation Fields on Solid
#include "UpdateAblationFields.H"

        // --- Directed interpolation of primitive fields onto faces

        surfaceScalarField rho_pos(interpolate(rho, pos));
        surfaceScalarField rho_neg(interpolate(rho, neg));

        surfaceVectorField rhoU_pos(interpolate(rhoU, pos, U.name()));
        surfaceVectorField rhoU_neg(interpolate(rhoU, neg, U.name()));

        volScalarField rPsi("rPsi", 1.0/psi);
        surfaceScalarField rPsi_pos(interpolate(rPsi, pos, T.name()));
        surfaceScalarField rPsi_neg(interpolate(rPsi, neg, T.name()));

        surfaceScalarField e_pos(interpolate(e, pos, T.name()));
        surfaceScalarField e_neg(interpolate(e, neg, T.name()));

        surfaceVectorField U_pos("U_pos", rhoU_pos/rho_pos);
        surfaceVectorField U_neg("U_neg", rhoU_neg/rho_neg);

        surfaceScalarField p_pos("p_pos", rho_pos*rPsi_pos);
        surfaceScalarField p_neg("p_neg", rho_neg*rPsi_neg);

        surfaceScalarField phiv_pos("phiv_pos", U_pos & mesh.Sf());
        surfaceScalarField phiv_neg("phiv_neg", U_neg & mesh.Sf());

```

```

volScalarField c("c", sqrt(thermo.Cp()/thermo.Cv()*rPsi));
surfaceScalarField cSf_pos
(
    "cSf_pos",
    interpolate(c, pos, T.name())*mesh.magSf()
);
surfaceScalarField cSf_neg
(
    "cSf_neg",
    interpolate(c, neg, T.name())*mesh.magSf()
);

surfaceScalarField ap
(
    "ap",
    max(max(phiv_pos + cSf_pos, phiv_neg + cSf_neg), v_zero)
);
surfaceScalarField am
(
    "am",
    min(min(phiv_pos - cSf_pos, phiv_neg - cSf_neg), v_zero)
);

surfaceScalarField a_pos("a_pos", ap/(ap - am));

surfaceScalarField amaxSf("amaxSf", max(mag(am), mag(ap)));

surfaceScalarField aSf("aSf", am*a_pos);

if (fluxScheme == "Tadmor")
{
    aSf = -0.5*amaxSf;
    a_pos = 0.5;
}

surfaceScalarField a_neg("a_neg", 1.0 - a_pos);

```

```

phiv_pos *= a_pos;
phiv_neg *= a_neg;

surfaceScalarField aphiv_pos("aphiv_pos", phiv_pos - aSf);
surfaceScalarField aphiv_neg("aphiv_neg", phiv_neg + aSf);

// Reuse amaxSf for the maximum positive and negative fluxes
// estimated by the central scheme
amaxSf = max(mag(aphiv_pos), mag(aphiv_neg));

#include "centralCourantNo.H"
#include "readTimeControls.H"

if (LTS)
{
    #include "setRDeltaT.H"
}
else
{
    #include "setDeltaT.H"
}

runTime++;

Info<< "Time = " << runTime.timeName() << nl << endl;

phi = aphiv_pos*rho_pos + aphiv_neg*rho_neg;

surfaceVectorField phiUp
(
    (aphiv_pos*rhoU_pos + aphiv_neg*rhoU_neg)
    + (a_pos*p_pos + a_neg*p_neg)*mesh.Sf()
);

surfaceScalarField phiEp
(
    "phiEp",

```

```

        aphiv_pos*(rho_pos*(e_pos + 0.5*magSqr(U_pos)) + p_pos)
    + aphiv_neg*(rho_neg*(e_neg + 0.5*magSqr(U_neg)) + p_neg)
    + aSf*p_pos - aSf*p_neg
);

volScalarField muEff("muEff", turbulence->muEff());
volTensorField tauMC("tauMC", muEff*dev2(Foam::T(fvc::grad(U))));

// --- Solve density
solve(fvm::ddt(rho) + fvc::div(phi) );

// --- Solve momentum
solve(fvm::ddt(rhoU) + fvc::div(phiUp));

U.ref() =
    rhoU()
    /rho();
U.correctBoundaryConditions();
rhoU.boundaryFieldRef() == rho.boundaryField()*U.boundaryField();

if (!inviscid)
{
    solve
    (
        fvm::ddt(rho, U) - fvc::ddt(rho, U)
        - fvm::laplacian(muEff, U)
        - fvc::div(tauMC)
    );
    rhoU = rho*U;
}

// --- Solve energy
surfaceScalarField sigmaDotU
(
    "sigmaDotU",
    (
        fvc::interpolate(muEff)*mesh.magSf()*fvc::snGrad(U)

```

```

        + fvc::dotInterpolate(mesh.Sf(), tauMC)
    )
    & (a_pos*U_pos + a_neg*U_neg)
);

thermo.correct();

solve
(
    fvm::ddt(rhoE)
    + fvc::div(phiEp)
    - fvc::div(sigmaDotU)
);

e = rhoE/rho - 0.5*magSqr(U);
e.correctBoundaryConditions();
thermo.correct();
rhoE.boundaryFieldRef() ==
    rho.boundaryField()*
    (
        e.boundaryField() + 0.5*magSqr(U.boundaryField())
    );

if (!inviscid)
{
    solve
    (
        fvm::ddt(rho, e) - fvc::ddt(rho, e)
        - fvm::laplacian(turbulence->alphaEff(), e)
    );
    thermo.correct();
    rhoE = rho*(e + 0.5*magSqr(U));
}

p.ref() =
    rho()
    /psi();

```

```

p.correctBoundaryConditions();
rho.boundaryFieldRef() == psi.boundaryField()*p.boundaryField();

turbulence->correct();

runTime.write();

Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
      << "   ClockTime = " << runTime.elapsedClockTime() << " s"
      << nl << endl;
}

Info<< "End\n" << endl;

return 0;
}

// ***** //

```


PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programas de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Contam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. Aceitam-se tanto programas fonte quanto os executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.