DEPARTAMENTO DE MATEMÁTICA APLICADA
INSTITUTO DE MATEMÁTICA - UNIVERSIDADE FEDERAL DO RIO GRANDE

DESENVOLVIMENTO DE UM MODELO COMPUTACIONAL MHD VISCO-RESISTIVO PARA ESTUDOS DE *PLASMA START-UP* NO TOKAMAK TCABR

Aluno: Kévi Pegoraro (FURG)

Orientador: Elizando Domingues dos Santos (FURG)

Co-orientador: Gustavo Paganini Canal (IFUSP)

Resumo

Resumo ...

Abstract

Abstract ...

Conteúdo

1 Introdução

1.1 A necessidade de novas fontes de energia

Devido ao crescimento da população mundial, e o aumento do consumo de energia per-capita, um dos maiores desafios da atualidade é atender esta crescente demanda por energia de maneira responsável e sustentável. Avanços recentes na área mostram a viabilidade de se obter energia "fundindo" átomos, maneira que atende a essas necessidades. As reações de fusão nuclear são limpas, seguras e a quantidade de combustível na Terra é virtualmente inesgotável. Há mais de 50 anos cientistas no mundo todo buscam construir máquinas capazes de realizar fusão com um balanço positivo de energia. Hoje a configuração mais promissora é o tokamak, que confina o plasma dentro de uma câmera de vácuo, usando potentes campos magnéticos.

1.2 A fusão termonuclear controlada

A fusão termonuclear, que significa alcançar fusão através da altas temperaturas, é a reação que gera a energia de todas as estrelas. Em tais reações, núcleos de baixa massa se combinam, ou se fundem, para formar núcleos mais massivos. A palavra controlada aqui significa recriar esta reação na terra com um balanço positivo de energia.

No sol, uma sequência de reações de fusão, denominada cadeia p-p, começa com prótons - núcleos de hidrogênio comum - termina com partículas alfa - núcleos de átomos de hélio. Após uma reação de fusão, as massas finais são menores do que as iniciais e a diferença de massa é convertida em energia. O processo de fusão é naturalmente seguro, embora a reação de fusão também produza resíduos radioativos. No entanto, tais subprodutos são as partículas α (núcleos de Hélio). O fluxo de nêutrons em um reator tornará os materiais estruturais radioativos. O trítio tem uma meia-vida de apenas 12 anos, enquanto a escolha apropriada de materiais pode resultar em resíduos que têm meias-vidas de dezenas de anos, em vez de milhares de anos, como na fissão. Outra enorme vantagem da fusão é que os materiais usados para reação de fusão podem ser extraídas da água do mar. Portanto, a fusão é considerada uma fonte de energia virtualmente inesgotável. Como a fusão deve ser continuamente alimentada, e sua manutenção depende estritamente do equilíbrio MHD, ela é facilmente interrompida. Mesmo nos piores acidentes imagináveis, o plasma contido no tokamak não terá energia suficiente para causar a ruptura da câmera de vácuo.

1.3 O confinamento magnético e o conceito tokamak

O tokamak é um conceito de máquina de confinamento de plasmas que tem como objetivo criar condições onde a fusão nuclear aconteça. O objetivo final da pesquisa com tokamaks é tornar viável a construção de reatores nucleares de fusão com balanço positivo de energia, ou seja, a energia extraída é maior que a gasta para manter as reações de fusão. Atingir isto é o desafio das pesquisas atuais sobre a fusão que está na construção de reatores, econômica e tecnologicamente viáveis. Os dois processos de confinamento de plasmas mais utilizados experimentalmente em laboratórios são: confinamento magnético e confinamento inercial. Na fusão por confinamento magnético, o gás ionizado a elevadas temperaturas é confinado por meio de campos magnéticos. Na fusão por confinamento inercial, o combustível é fortemente comprimido até acontecer a ignição. Quanto à geometria, o confinamento magnético pode ser dividido nas configurações lineares e a toroidal. a) Na configuração linear, as partículas movimentam-se entre as duas extremidades de um cilindro com simetria axial. Nesse tipo de geometria, os campos magnéticos tem maior intensidade nas extremidades onde estão os chamados "espelhos magnéticos". Este movimento de vaivém das partículas está no entanto sujeito a grandes perdas na região dos espelhos e por isso esta configuração não é indicada para os reatores. Na configuração toroidal, as linhas de campos magnéticos são curvados e se fecham em si mesmas, portanto, não se tem as perdas que ocorrem nas máquinas lineares. Para se ter confinamento nesta configuração existe a necessidade de sobrepor ao campo toroidal uma componente que seja perpendicular a ele (campo poloidal). As linhas de força do campo total tem a forma de hélices em torno das quais se movimentam as partículas do plasma. Portanto, basicamente, um tokamak é um potente eletroímã que produz um campo magnético toroidal. Um toroide é a configuração mais simples com linhas de campo magnético fechadas, condição está necessária para evitar a perda do plasma. O gradiente da amplitude do campo magnético e a curvatura das linhas de campo levam a movimentos de íons e elétrons em direções verticais opostas, o que resulta em uma separação de cargas. Esta, por sua vez, cria um campo elétrico que, junto com o campo magnético produzirá uma deriva das partículas na direção radial.

Nas máquinas Tokamak, tem-se uma câmara, em geral de aço inoxidável, onde um campo magnético forte na direção toroidal ($\boldsymbol{B}_T$) é gerado a partir de uma corrente que circula num conjunto de bobinas regularmente espaçadas em torno do eixo secundário da máquina (direção toroidal). A coluna de plasma tem a tendência a expandir e chocar-se contra as paredes da câmera. Para manter a coluna de plasma no centro da câmera é necessário à aplicação de um campo vertical ($\boldsymbol{B}_v$) que, interagindo com a corrente de plasma,

cria uma força $\boldsymbol{J} \times \boldsymbol{B}$ na direção radial com sentido apontando para o centro da máquina. A corrente de plasma (I_p) é gerada indutivamente pelo transformador de Aquecimento Ôhmico e esta corrente gera o campo magnético poloidal ($\boldsymbol{B}_p$) que tem, também, importância na estabilidade da coluna de plasma. O Sistema de Aquecimento Ôhmico é igualmente responsável pelo aquecimento do plasma. Da soma do campo magnético toroidal (gerado pelo Sistema Toroidal) com o campo poloidal, gerado pela corrente de plasma, resulta num campo magnético cujas linhas de força são helicoidais. A reação de fusão mais promissora é a de deutério e trítio. A grande quantidade de energia liberada servirá para aquecer água, produzir vapor e assim mover uma turbina acoplada a um gerador elétrico [1, pg 16].

1.4 A geração de plasmas em tokamaks: o *start-up*

No interior da câmera de vácuo de um tokamak ocorre uma emissão de elétrons que são altamente acelerados pelo campo elétrico, que então provocam uma ruptura (*breakdown*) do gás de trabalho gerando a corrente de plasma, que irá formar o plasma que deve ser contido no espaço limitado da câmera de vácuo. O campo magnético não pode permitir que o plasma toque nas paredes internas da câmera de vácuo, tanto para não danificá-la, quanto para não dissipar a energia do plasma via condução térmica ou contaminar o plasma com átomos e moléculas pesadas. Alguns tokamaks tem seção reta da câmera retangular, como o TCABR da USP, enquanto outros possuem seção reta circular, como o NOVA-FURG, ou até alongada em formato elíptico, como o ITER. Outra característica dos tokamaks é o uso da corrente de plasma para gerar a componente helicoidal do seu intenso campo magnético, necessária para um equilíbrio MHD [2, pg. 34], [3, pg. 10].

1.4.1 A fase de *breakdown*

A fase de *breakdown* em um tokamak é dominada por colisões entre elétrons livres e partículas neutras. A fase de *breakdown* começa com a primeira ionização e dura até que as colisões de Coulomb comecem a dominar as colisões neutras com elétrons [4, pg. 55]. Este processo pode ser descrito pelo modelo desenvolvido por Townsend [5]. A descarga de Townsend (*Townsend discharge*) é um processo de ionização de gás onde os elétrons livres, sob efeito de um campo elétrico intenso, são acelerados para então colidirem com moléculas de gás e liberarem elétrons adicionais, Figura 1. Elétrons que também são acelerados, colidem e liberam mais elétrons adicionais, resultando em uma avalanche que permitirá a passagem de corrente pelo gás. Em um tokamak, o plasma parcialmente ionizado é condutor e uma corrente de plasma toroidal é formada devido ao campo elétrico toroidal. A descarga requer

uma fonte de elétrons livres e um campo elétrico suficientemente intenso. O primeiro coeficiente de Townsend, explicado e modelado na seção 3.1, corresponde ao número de reações de ionização, por unidade de comprimento, causadas por um elétron que se move paralelamente ao campo elétrico. O campo magnético poloidal, gerado pela corrente de plasma toroidal, começa a aumentar até que a corrente se estabiliza. São formadas então, superfícies de fluxo magnético fechadas que reduzem a perda de íons e elétrons, o que causa um aumento na taxa de crescimento da corrente de plasma. Uma explicação mais detalhada para a fase de *breakdown* pode ser encontrada em [6, pg. 40].

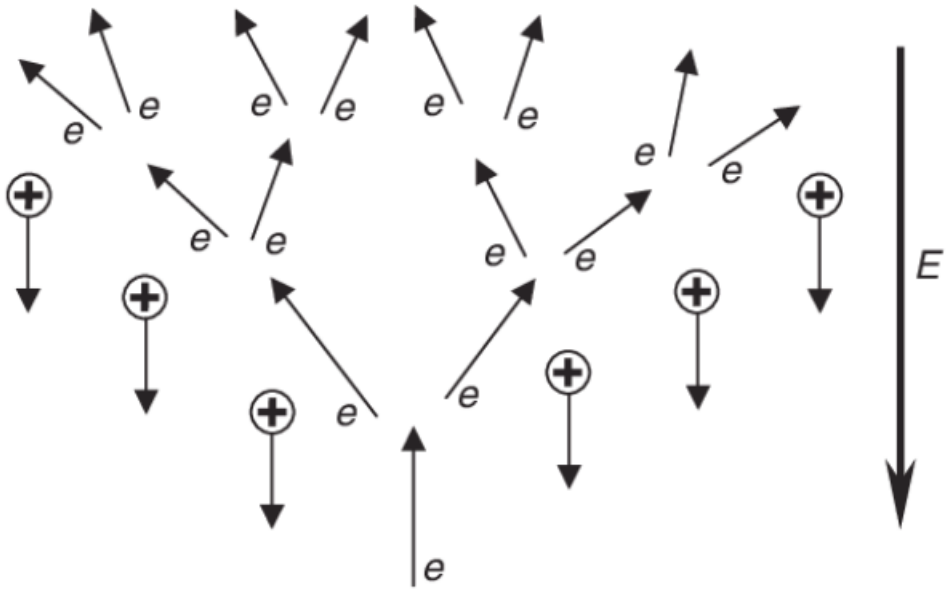

Figura 1: Ilustração do breakdown de Townsend [7].

1.4.2 A fase de *burn-through*

A fase de *burn-through* é caracterizada pelo aumento da corrente de plasma causada pelo aumento da ionização do plasma. A ionização do gás neutro e radiações de linha das impurezas presentes no plasma, resultam na perda de uma parte significativa do poder de aquecimento [8, 9]. Esta perda de potência é proporcional ao produto da densidade de elétrons com a densidade de partículas neutras, e tem seu máximo na chamada barreira de radiação. O plasma precisa "queimar" essa barreira de radiação antes que a potência de aquecimento possa elevar a temperatura do plasma. Um estado de alta ionização de impurezas é normalmente alcançado após a "queima" do gás principal. Uma "queima" (*burn-through*) de plasma bem sucedida só pode ser obtida se a potência de aquecimento ôh-

mico exceder a perda de potência por ionização e radiação. Depois que a queima é realizada, a corrente de plasma é tipicamente aumentada até que o valor máximo de corrente de plasma (*flattop*) seja atingido. Durante a fase de aceleração da corrente de plasma (*ramp up* 1.4.3), é essencial evitar interrupções causadas por instabilidades MHD. As fases de *breakdown*, *burn-through* e *ramp-up* não são necessariamente fases consecutivas, mas processos que podem ocorrer simultaneamente. É fundamental para o funcionamento do tokamak a obtenção da configuração de superfícies de fluxo fechado. Para isso, a corrente de plasma deve crescer e o gás combustível deve ser totalmente ionizado durante a fase de queima de plasma (*burn-through*). Uma das características cruciais na fase de *burn-through* é a transição da configuração do campo magnético, ou seja, a mudança de configuração de linhas de campo abertas para as superfícies de fluxo fechadas. A Figura 2 mostra dados típicos da fase de *burn-through* no tokamak JET. Mais detalhes da fase de *burn-through* podem ser encontrados em [8, 10].

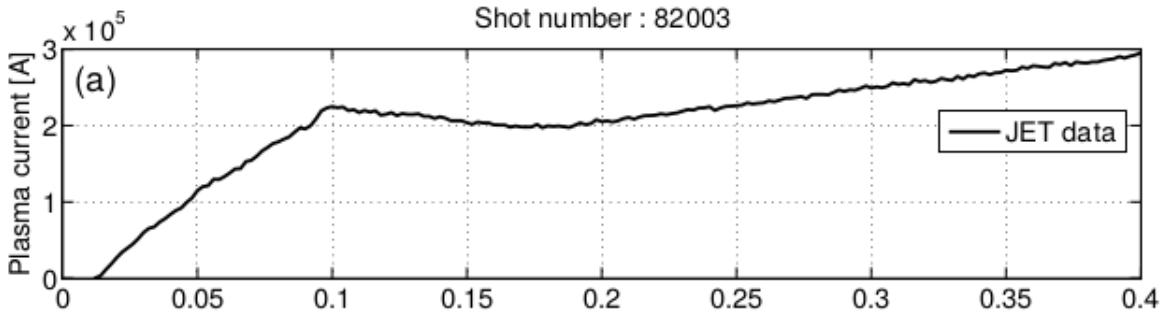

Figura 2: Dados experimentais típicos de corrente de plasma medidos durante a fase de *burn-through* no tokamak JET [11].

1.4.3 A fase de *current ramp-up*

A fase de *ramp-up* inicia-se após a queima do gás principal, mas é independente da quantidade de impurezas, podendo, então, se sobrepor à fase de *burn-through*. A Figura 3 apresenta um diagrama de como são caracterizadas, ao longo do tempo, as fases de *breakdown*, *burn-through* e *ramp-up*. Uma explicação mais detalhada para a fase de *ramp-up* pode ser encontrada em [6, pg. 49]. A Figura 3 apresenta um diagrama com as três fases ilustradas.

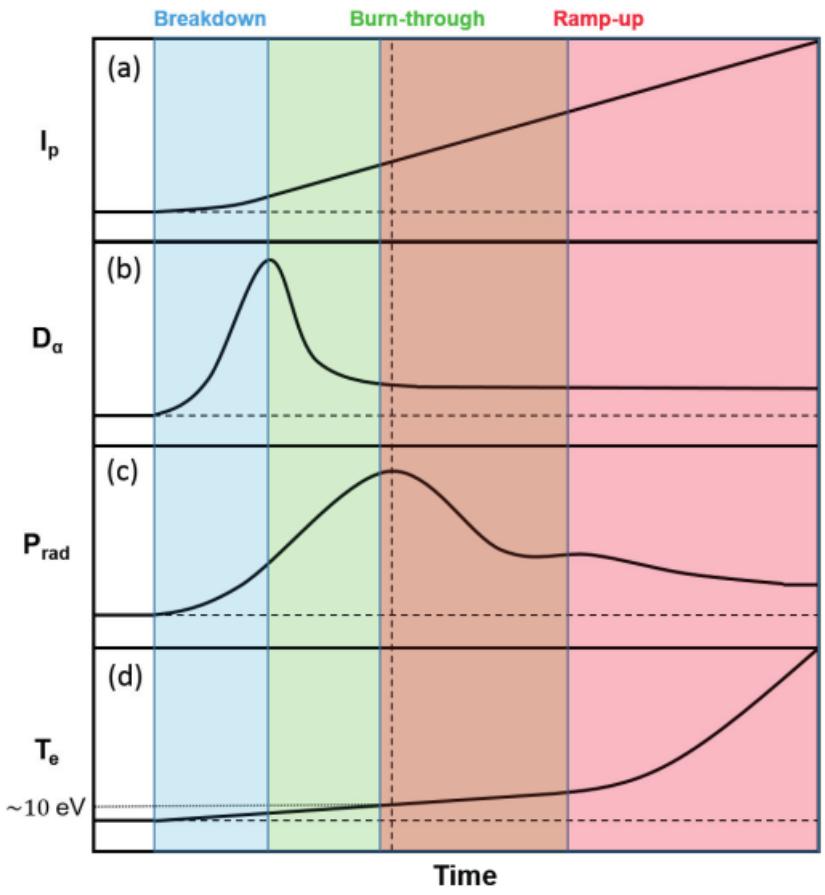

Figura 3: Figura esquemática da evolução temporal da (a) corrente de plasma, (b) emissão de D_α, (c) perda de potência de radiação e ionização e (d) temperatura do elétron em uma formação típica de plasma de deutério. As fases de *breakdown* (azul), *burn-through* (verde), de *ramp-up* (vermelho) e a sobreposição entre a fase de *burn-through* e a fase de *ramp-up* (marrom) são marcadas com as respectivas cores. A linha tracejada vertical representa a barreira de radiação [4].

1.5 A necessidade de um modelo validado de *start-up*

2 Objetivos

Falar da necessidade de termos modelos computacionais validados para projetar cenários de plasma *start-up* em tokamaks.

Os objetivos específicos deste trabalho são:

- Desenvolver um modelo de fluido para estudos de breakdown
- Desenvolver um modelo computacional robusto que possa ser usado para o projeto de cenários de breakdown no tokamak TCABR

3 Modelagem teórica do *start-up*

3.1 A avalanche de Townsend

O modelo de Townsend assume que os campos elétricos acionados externamente são predominantes no dispositivo. Tal suposição é verdadeira para dispositivos com gás neutro de baixa pressão e curto espaço entre os eletrodos, porque a quantidade de carga espacial local produzida durante o inicio do avalanche de elétrons é desprezível devido à pequena taxa de crescimento, α_T. Portanto as características da avalanche de elétrons em dispositivos de baixa pressão de gás dependem dos campos elétricos externos e podem ser descritas pelo modelo de Townsend.

Assumindo que, devido à sua maior massa, os íons são estacionários, e os elétrons livres são responsáveis pela ionização. O aumento da densidade eletrônica, n_e, é proporcional à diferença entre a taxa de ionização, ν_{in} (geração de elétrons) e a taxa de perda de elétrons, ν_{loss}

$$\frac{dn_e}{dt} = (\nu_{in} - \nu_{loss}) n_e \tag{1}$$

portanto, nesta fase, a densidade eletrônica $n_e(t)$ pode ser expressa como

$$n_e(t) = n_{e0} e^{(\nu_{in} - \nu_{loss})t} \tag{2}$$

onde t descreve o tempo, n_{e0} é a densidade de elétrons em $t = 0$ o *breakdown* ocorre quando a taxa de geração de elétrons excede a taxa de perda de elétrons. Estes elétrons então são acelerados pelo campo elétrico toroidal e alcançam a velocidade de deriva (*drift velocity*). Observe que a eq. 2 é válida somente quando o grau de ionização permanece pequeno, deste modo as colisões de elétrons com as partículas neutras predominam sobre as colisões Coulombianas. Para modelar o processo de ionização, vamos escrever a taxa de ionização em termos do primeiro coeficiente de Townsend e a velocidade de deriva do elétron

$$\nu_{in} = \alpha_T u_{D,||} \tag{3}$$

onde $u_{D,||}$ é a velocidade de deriva, α_T é o primeiro coeficiente de Townsend que é dado por

$$\alpha_T = A p_n e^{B \frac{p_n}{E_\Theta}} \tag{4}$$

Aqui, A e B são constantes que dependem do gás de trabalho, p_n é a pressão de gás neutro e E_Θ é a magnitude do campo elétrio externo. A taxa de perda de elétrons devido ao seu

movimento ao longo das linhas do campo magnético pode ser expressa por

$$\nu_{loss} = \frac{u_{D,||}}{L_{\text{eff}}} \tag{5}$$

onde L_{eff} é a distância média percorrida pelo elétron antes de se chocar com a parede. Fazendo o lado direito da eq. 1 ir para zero, cria-se uma condição para o início do *breakdown*

$$Ap_n e^{B\frac{p_n}{E_\Theta}} = \frac{1}{L_{\text{eff}}} \tag{6}$$

Esta equação mostra que um *breakdown* bem-sucedido em um tokamak depende da escolha da pressão do gás neutro, da intensidade do campo elétrico toroidal e do comprimento efetivo da conexão (L_{eff}), que depende da configuração do campo poloidal durante a fase inicial. Existe então um valor mínimo do campo elétrico toroidal induzido para uma dada pressão de gás neutro e L_{eff} para que se possa obter um *breakdown*. Uma explicação detalhada para a modelagem da teoria de Townsend pode ser encontrada em [12].

3.2 A curva de Paschen

A Lei de Paschen afirma essencialmente que, em pressões mais altas (acima de algumas torr), as características de degradação de uma lacuna são uma função (geralmente não linear) do produto da pressão do gás e do comprimento da lacuna, geralmente escrito como $V = f(pd)$, onde p é a pressão e d é a distância da lacuna [13]. A curva de Paschen descrevendo a tensão de breakdown de um gás de hidrogênio entre placas paralelas é mostrada na Figura 4.

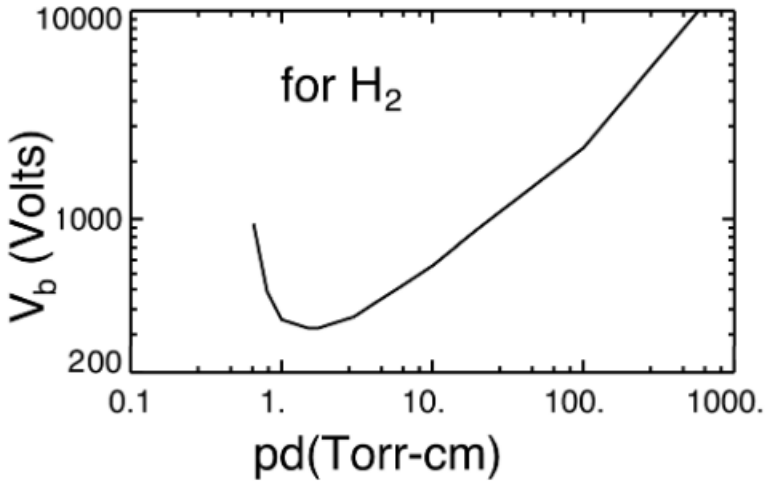

Figura 4: A curva de Paschen da tensão de breakdown, V, entre placas paralelas separadas por uma distância d na pressão p para o hidrogênio. Observe que existe um pd mínimo para o qual ocorre a quebra e, acima desse mínimo, V aumenta aproximadamente linearmente, de modo que para a separação fixa $\frac{E}{p}$ é aproximadamente constante ($\boldsymbol{E}$ é o campo elétrico). [13].

3.3 Desenvolvimento do modelo físico do *start-up*

Uma vez que o processo de *start-up* envolve a ionização de um gás inicialmente neutro pela aplicação de um campo elétrico, é necessário modelar esse processo utilizando dois conjuntos de equações: um para modelar o gás neutro e outro para modelar o plasma. Para o gás neutro, será usado um modelo de fluido não-viscoso,

$$\left(\frac{\partial}{\partial t} + \boldsymbol{u_n}\cdot\boldsymbol{\nabla}\right)\rho_{mn} = -\rho_{mn}\left(\boldsymbol{\nabla}\cdot\boldsymbol{u_n}\right) + S_n \tag{7}$$

$$\rho_{mn}\left(\frac{\partial}{\partial t} + \boldsymbol{u_n}\cdot\boldsymbol{\nabla}\right)\boldsymbol{u_n} = -\boldsymbol{\nabla}p_n - \boldsymbol{u_n}S_n + \boldsymbol{A_n} \tag{8}$$

$$\frac{1}{\Gamma_n - 1}\left(\frac{\partial}{\partial t} + \boldsymbol{u_n}\cdot\boldsymbol{\nabla}\right)p_n = -\frac{\Gamma_n p_n\left(\boldsymbol{\nabla}\cdot\boldsymbol{u_n}\right)}{\Gamma_n - 1} - \boldsymbol{\nabla}\cdot\boldsymbol{q_n} + \frac{u_n^2 S_n}{2} - \boldsymbol{u_n}\cdot\boldsymbol{A_n} + \mathscr{E}_n. \tag{9}$$

Neste modelo, o estado do gás neutro é caracterizado por sua densidade de massa ρ_{mn}, sua velocidade $\boldsymbol{u_n}$, sua pressão p_n, seu fluxo de calor $\boldsymbol{q_n}$, sua constante adiabática Γ_n e os termos de fonte S_n, $\boldsymbol{A_n}$ e $\mathscr{E}_n$, que correspondem às fontes de massa, momento e energia, respectivamente.

Para modelar o plasma, será usado um modelo magnetohidrodinâmico (MHD) visco-

resistivo [14],

$$\left(\frac{\partial}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\right)\rho_m=-\rho_m\left(\boldsymbol{\nabla}\cdot\boldsymbol{u}\right)+S \tag{10}$$

$$\rho_m\left(\frac{\partial}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\right)\boldsymbol{u}=\boldsymbol{J}\times\boldsymbol{B}-\boldsymbol{\nabla}\cdot\boldsymbol{\mathcal{P}}-\boldsymbol{u}S+\boldsymbol{A} \tag{11}$$

$$\frac{1}{\Gamma-1}\left(\frac{\partial}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\right)p=-\frac{p\left(\boldsymbol{\nabla}\cdot\boldsymbol{u}\right)}{\Gamma-1}-\boldsymbol{\nabla}\cdot\boldsymbol{q}-\left(\boldsymbol{\mathcal{P}}\cdot\boldsymbol{\nabla}\right)\cdot\boldsymbol{u}+\boldsymbol{J}\cdot\boldsymbol{E}-\boldsymbol{u}\cdot\left(\boldsymbol{J}\times\boldsymbol{B}\right)+\frac{u^2S}{2}-\boldsymbol{u}\cdot\boldsymbol{A}+\mathscr{E} \tag{12}$$

$$\boldsymbol{\nabla}\times\boldsymbol{E}=-\frac{\partial\boldsymbol{B}}{\partial t} \tag{13}$$

$$\boldsymbol{\nabla}\times\boldsymbol{B}=\mu_0\boldsymbol{J} \tag{14}$$

$$\boldsymbol{E}+\boldsymbol{u}\times\boldsymbol{B}=\eta\boldsymbol{J}+\frac{m_i}{\rho_m e}\boldsymbol{J}\times\boldsymbol{B} \tag{15}$$

Neste modelo, o estado do plasma é caracterizado por sua densidade de massa ρ_m, sua velocidade $\boldsymbol{u}$, seu tensor pressão $\boldsymbol{\mathcal{P}}$, seu fluxo de calor $\boldsymbol{q}$, sua constante adiabática Γ, o campo elétrico $\boldsymbol{E}$, o campo magnético $\boldsymbol{B}$, a densidade de corrente $\boldsymbol{J}$ e os termos de fonte S, $\boldsymbol{A}$ e $\mathscr{E}$, que correspondem às fontes de massa, momento e energia, respectivamente. Aqui, μ_0 é a permeabilidade magnética do vácuo, e é a carga elétrica elementar, m_i é a massa atômica do íon e η é a resistividade elétrica do plasma.

Termos de fonte

Descrever aqui os termos de fonte...

Aqui, $S_n = S_{n,ext} - S_{n,ionz}$, onde , $\boldsymbol{A_n} = \boldsymbol{A_{n,ext}} + \boldsymbol{A_{n,coll}} - \boldsymbol{A_{n,ex}}$, onde ... $\mathscr{E}_n = \mathscr{E}_{n,ext} - \mathscr{E}_{ex}$ ...

$S = S_e + S_i = -S_n$ the plasma mass source, $\boldsymbol{A} = \boldsymbol{A_e} + \boldsymbol{A_i} = -\boldsymbol{A_n}$ the plasma momentum source, $\mathscr{E} = \mathscr{E}_e + \mathscr{E}_i = -\mathscr{E}_n$ the plasma energy source, and $\Gamma = 5/3$ the ratio of specific heats. In addition, one will assume that both the electric resistivity and viscosity are constant across the plasma.

$$\boldsymbol{A_{\mathbf{n,coll}}} = -\rho_{mn}\nu_{np}\left(\boldsymbol{u_n}-\boldsymbol{u}\right) \tag{16}$$

Fluxo de calor

Nesse modelo, iremos modelar o fluxo de calor como sendo

$$\boldsymbol{q}=-\kappa\boldsymbol{\nabla}p. \tag{17}$$

Dessa forma, o termo $-\boldsymbol{\nabla}\cdot\boldsymbol{q}$ torna-se

$$-\boldsymbol{\nabla}\cdot\boldsymbol{q} = \kappa\boldsymbol{\nabla}^2 p. \tag{18}$$

Tensor pressão de plasma

Nesse modelo, iremos decompor o tensor pressão de plasma numa pressão escalar, p, e num tensor de estresse viscoso, $\boldsymbol{\Pi}$, ou seja, $\boldsymbol{\mathcal{P}} = p\boldsymbol{I} + \boldsymbol{\Pi}$, onde $\boldsymbol{I}$ é a matrix unitária. Além disso, iremos expressar o tensor de estresse viscoso como

$$\boldsymbol{\Pi} = -\boldsymbol{\mu}\cdot\left[\boldsymbol{\nabla u} + (\boldsymbol{\nabla u})^T - \frac{2}{3}(\boldsymbol{\nabla}\cdot\boldsymbol{u})\,\boldsymbol{I}\right], \tag{19}$$

onde $\boldsymbol{\mu} = \mu_\perp \boldsymbol{I} + \left(\mu_{||} - \mu_\perp\right)\hat{\boldsymbol{b}}\hat{\boldsymbol{b}}$ é a viscosidade cinemática anisotrópica do plasma.

Resistividade elétrica do plasma

By computing the divergence of the plasma pressure tensor, and using that $\boldsymbol{\nabla}\cdot(\boldsymbol{\nabla}\boldsymbol{u}) = \boldsymbol{\nabla}^2\boldsymbol{u}$ and $\boldsymbol{\nabla}\cdot\left[(\boldsymbol{\nabla}\boldsymbol{u})^T\right] = \boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{u})$, the compressible momentum equation becomes

$$\rho_m\left(\frac{\partial}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\right)\boldsymbol{u} = \boldsymbol{J}\times\boldsymbol{B}-\boldsymbol{\nabla}p+\mu\rho_m\left[\nabla^2\boldsymbol{u}+\frac{1}{3}\boldsymbol{\nabla}(\boldsymbol{\nabla}\cdot\boldsymbol{u})\right]-\boldsymbol{u}S_p+\boldsymbol{A}. \tag{20}$$

Using the rate-of-strain tensor above, the energy conservation equation equation becomes

$$\frac{1}{\Gamma-1}\left(\frac{\partial}{\partial t}+\boldsymbol{u}\cdot\boldsymbol{\nabla}\right)p = -\frac{\Gamma\, p\,(\boldsymbol{\nabla}\cdot\boldsymbol{u})}{\Gamma-1}-\boldsymbol{\nabla}\cdot\boldsymbol{q}-(\boldsymbol{\Pi}\cdot\boldsymbol{\nabla})\cdot\boldsymbol{u}+\boldsymbol{J}\cdot\boldsymbol{E}-\boldsymbol{u}\cdot(\boldsymbol{J}\times\boldsymbol{B})-\boldsymbol{u}\cdot\boldsymbol{A}-\frac{1}{2}u^2S_p+\mathscr{E}. \tag{21}$$

Sistema de equações em coordenadas cartesianas

This set of equation can be written in a cartesian coordinate system as

$$\frac{\partial\rho_m}{\partial t}+u_x\frac{\partial\rho_m}{\partial x}+u_y\frac{\partial\rho_m}{\partial y}+u_z\frac{\partial\rho_m}{\partial z} = -\rho_m\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)+S_p \tag{22}$$

$$\rho_m\left(\frac{\partial u_x}{\partial t}+u_x\frac{\partial u_x}{\partial x}+u_y\frac{\partial u_x}{\partial y}+u_z\frac{\partial u_x}{\partial z}\right) = J_yB_z-J_zB_y-\frac{\partial p}{\partial x}-u_xS_p+A_x+ \tag{23}$$

$$+\mu\rho_m\left(\frac{4}{3}\frac{\partial^2u_x}{\partial x^2}+\frac{\partial^2u_x}{\partial y^2}+\frac{\partial^2u_x}{\partial z^2}+\frac{1}{3}\frac{\partial^2u_y}{\partial x\partial y}+\frac{1}{3}\frac{\partial^2u_z}{\partial x\partial z}\right) \tag{24}$$

$$\rho_m\left(\frac{\partial u_y}{\partial t}+u_x\frac{\partial u_y}{\partial x}+u_y\frac{\partial u_y}{\partial y}+u_z\frac{\partial u_y}{\partial z}\right) = J_zB_x-J_xB_z-\frac{\partial p}{\partial y}-u_yS_p+A_y+ \tag{25}$$

$$+\mu\rho_m\left(\frac{\partial^2u_y}{\partial x^2}+\frac{4}{3}\frac{\partial^2u_y}{\partial y^2}+\frac{\partial^2u_y}{\partial z^2}+\frac{1}{3}\frac{\partial^2u_x}{\partial y\partial x}+\frac{1}{3}\frac{\partial^2u_z}{\partial y\partial z}\right) \tag{26}$$

$$\rho_m\left(\frac{\partial u_z}{\partial t}+u_x\frac{\partial u_z}{\partial x}+u_y\frac{\partial u_z}{\partial y}+u_z\frac{\partial u_z}{\partial z}\right) = J_xB_y-J_yB_x-\frac{\partial p}{\partial z}-u_zS_p+A_z+ \tag{27}$$

$$+\mu\rho_m\left(\frac{\partial^2u_z}{\partial x^2}+\frac{\partial^2u_z}{\partial y^2}+\frac{4}{3}\frac{\partial^2u_z}{\partial z^2}+\frac{1}{3}\frac{\partial^2u_x}{\partial z\partial x}+\frac{1}{3}\frac{\partial^2u_y}{\partial z\partial y}\right) \tag{28}$$

$$\frac{\partial p}{\partial t}+u_x\frac{\partial p}{\partial x}+u_y\frac{\partial p}{\partial y}+u_z\frac{\partial p}{\partial z} = -\Gamma p\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)+(\Gamma-1)\eta\left(J_x^2+J_y^2+J_z^2\right) \tag{29}$$

$$\frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} = -\frac{\partial B_x}{\partial t} \qquad \frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x} = -\frac{\partial B_y}{\partial t} \qquad \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} = -\frac{\partial B_z}{\partial t} \tag{30}$$

$$\frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z} = \mu_0 J_x \qquad \frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x} = \mu_0 J_y \qquad \frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y} = \mu_0 J_z \tag{31}$$

$$E_x + u_y B_z - u_z B_y = \eta J_x \qquad E_y + u_z B_x - u_x B_z = \eta J_y \qquad E_z + u_x B_y - u_y B_x = \eta J_z \tag{32}$$

4 Métodos numéricos e ferramentas computacionais

4.1 O método de volumes finitos

Rascunho

O Método de Volumes Finitos (FVM) é uma técnica numérica para transformar as equações diferenciais parciais que representam leis de conservação sobre volumes diferenciais em equações algébricas discretas sobre volumes finitos. Funciona de maneira semelhante aos métodos das diferenças finitas e dos elementos finitos, o primeiro passo no processo de solução é a discretização do domínio geométrico, que é discretizado em elementos não sobrepostos ou volumes finitos. As equações diferenciais parciais são então discretizadas transformadas em equações algébricas, integrando-as sobre cada elemento discreto. O sistema de equações algébricas é então resolvido para calcular os valores da variável dependente para cada um dos elementos [15]. No método dos volumes finitos, alguns dos termos da equação de conservação são transformados em fluxos de faces e avaliados nas faces de volumes finitos. Como o fluxo que entra em um determinado volume é idêntico ao que sai do volume adjacente, o FVM é estritamente conservador. Esta propriedade de conservação inerente do FVM torna o método preferido em CFD (Dinâmica dos fluidos computacional ou em inglês: Computational Fluid Dynamics). Outro atributo importante do FVM é que ele pode ser formulado no espaço físico em malhas poligonais não estruturadas. Finalmente, no FVM é bastante fácil implementar uma variedade de condições de contorno de uma maneira não invasiva, uma vez que as variáveis desconhecidas são avaliadas nos centroides dos elementos de volume, não em suas faces de contorno. Essas características tornaram o Método de Volume Finito bastante adequado para a simulação numérica de uma variedade de aplicações envolvendo fluxo de fluido e transferência de calor e massa, e os desenvolvimentos no método foram intimamente ligados aos avanços em CFD. De um potencial

limitado no início confinado à solução de física e geometria simples em grades estruturadas, o FVM agora é capaz de lidar com todos os tipos de física e aplicações complexas.

4.2 Algorítimo base para resolver as equações

Existem diversos algorítimos base, cada um com estratégias, condições e resultados diferentes, para resolver as equações CFD. Nesta seção irei citar alguns e após uma analise escolher qual é o mais adequado para o solver elaborado neste trabalho.

4.2.1 SIMPLE

O algorítimo SIMPLE (*Semi-Implicit Method for Pressure Linked Equations*) [16] trabalha com fluidos incompressíveis e usa a pressão kinematica ($p = \frac{p}{\rho}$).

$$\boldsymbol{MU} = -\nabla p \tag{33}$$

$$\boldsymbol{H} = \boldsymbol{AU} - \boldsymbol{MU} \tag{34}$$

$$\nabla \cdot \left(\boldsymbol{A}^{-1} \nabla p\right) = \nabla \cdot \left(\boldsymbol{A}^{-1} \boldsymbol{H}\right) \tag{35}$$

$$\boldsymbol{U} = \boldsymbol{A}^{-1} \boldsymbol{H} - \boldsymbol{A}^{-1} \nabla p \tag{36}$$

Para o cálculo da pressão é então derivada das equações de momento Eq. 33 e continuidade ($\nabla \cdot \boldsymbol{U} = 0$) uma equação pra pressão Eq. 35 (já que não é possível usar a equação dos gases ideias uma vez que ρ e T são constantes), e de maneira iterativa o algorítimo SIMPLE corrige $\boldsymbol{U}$ através da Eq. 36 e em seguida p através da Eq. 35 até atingir a tolerância estipulada pelo usuário. A sequencia de execução é:

1. Avança para a próxima iteração $t = t^{n+1}$
2. Inicializa $\boldsymbol{U}^{n+1}$ e p^{n+1} usando o último valor disponível de $\boldsymbol{U}$ e p
3. Constrói as equações de momento
4. Relaxa a matriz de momento
5. Resolve as equações de momento para obter uma previsão para $\boldsymbol{U}^{n+1}$
6. Constrói a equação de pressão
7. Resolve a equação de pressão para p^{n+1}

8. Corrige o fluxo para ϕ^{n+1}
9. Relaxa p^{n+1}
10. Corrige a velocidade para $\boldsymbol{U}^{n+1}$
11. Se não convergir, volta para o passo **2**

Algumas vantagens e desvantagens do algorítimo SIMPLE:

1. Juntamente com o tratamento de tempo implícito das variáveis de fluxo, pode-se obter com eficiência uma solução de estado estacionário ou usar etapas de tempo bastante grandes para cálculos instáveis de fluxo
2. O efeito direto da correção de pressão na velocidade é retirado da equação de correção de velocidade, o que causa correções de pressão muito grandes. Embora os relaxamentos corrijam a pressão, esses fatores são dependentes dos problemas, o que é uma grande desvantagem para o algoritmo
3. Como resultado, esse algoritmo destrói um bom campo de velocidade, a menos que também haja uma boa estimativa de pressão.
4. O tempo de computação pode ser reduzido usando este algoritmo. Para problemas relativamente simples (por exemplo, laminar, onde a convergência é limitada pelo acoplamento pressão-velocidade), a convergência pode ser obtida mais rapidamente.
5. Em contraste com as correções de pressão, tem boas correções de velocidade.

4.2.2 PISO

O algorítimo PISO (*Pressure Implicit with Splitting of Operators*) [16] pode ser resumido da seguinte forma:

1. Defina as condições de contorno.
2. Resolva a equação de momento discretizado para calcular um campo de velocidade intermediário.
3. Calcule os fluxos de massa nas faces das células.
4. Resolva a equação da pressão.

5. Corrija os fluxos de massa nas faces das células.
6. Corrija as velocidades com base no novo campo de pressão.
7. Atualize as condições de contorno.
8. Repita a partir do passo **3** pelo número prescrito de vezes.
9. Aumente o intervalo de tempo e repita a partir do passo **1**.

Para o algoritmo SIMPLE, os passos **4** e **5** podem ser repetidos por um número prescrito de vezes para corrigir a não ortogonalidade. Algumas vantagens e desvantagens do algorítimo PISO:

1. Geralmente fornece resultados mais estáveis e leva menos tempo de CPU, mas não é adequado para todos os processos.
2. Esquemas numéricos adequados para resolver a equação vinculada à pressão-velocidade.
3. Para laminar voltado para trás, o passo PISO é mais rápido do que SIMPLES, mas é mais lento no que diz respeito ao fluxo através do estabilizador vertical aquecido.
4. Se o momento e a equação escalar tiverem um acoplamento fraco ou nenhum, então PISO é melhor do que SIMPLEC (uma forma modificada do algoritmo SIMPLE).

4.2.3 PIMPLE

O algorítimo PIMPLE é uma combinação do algorítimo PISO com o SIMPLE [16]. No OpenFOAM estão implementados, em solucionadores, as versões incompressíveis e compressíveis destes três algorítimos.

4.3 Estabilidade numérica

Para garantir a estabilidade numérica é fundamental garantir a condição de convergência de Courant – Friedrichs – Lewy que é uma condição necessária para a convergência ao resolver numericamente o modelo MHD visco-resistivo. Surge na análise numérica de esquemas explícitos de integração temporal, quando estes são usados para a solução numérica. Como consequência, o intervalo de tempo deve ser menor que um certo tempo em muitas simulações de computador de marcha no tempo explícitas, caso contrário, a simulação produzirá resultados incorretos. A condição recebeu o nome de Richard Courant, Kurt Friedrichs e

Hans Lewy, que a descreveu em seu artigo de 1928 [17]. Nos códigos CFD é representada pela quantia *Courant Number* (Co), que tem um valor para cada célula da malha em cada passo temporal, mas os solucionadores normalmente usam apenas o valor máximo de Co.

Para melhorar a estabilidade numérica a malha foi refinada nas bordas e o domínio computacional suavizado a fim de remover quinas. O código CFD também usa do artificio chamado *wall function*, que trata as células na borda do domínio computacional como tendo, se necessário, uma dependência não linear da variação do campo em seu interior, oque permite melhor definir e calcular os gradientes nestas regiões.

4.3.1 Solução independente da malha

A malha de volumes finitos deve ser suficientemente refinada para identificar as características importantes do escoamento. É de capital importância calcular com precisão (novamente um critério deve ser empregado) os gradientes das variáveis, pois eles permitem o cálculo dos fluxos que interessam para a engenharia. Logo, o refino da malha deve ser realizado nas regiões de grandes gradientes, normalmente perto de paredes. A maneira mais adequada de verificar se a malha está cumprindo esta característica é realizar a análise em duas malhas de refinamento diferente e comparar os resultados obtidos para cada uma delas. Se o resultado não varia, novamente de acordo com um critério especificado pelo usuário, tem-se a chamada "independência de malha". Caso os resultados sejam diferentes entre as malhas, deve-se construir uma nova malha, mais refinada, e comparar seus resultados com a anterior. Esse processo deve ser repetido até que duas malhas de refinamento diferente promovam resultados que não sejam diferentes dentro do critério empregado.

4.3.2 Número de Reynolds

É um número adimensional usado para prever o perfil do fluxo do fluído, laminar (baixos valores) ou turbulento (altos valores), calculado via:

$$R_e = \frac{\rho \boldsymbol{U} L}{\mu} \tag{37}$$

onde μ é a viscosidade dinâmica [Kg/(m.s)], e L é a dimensão linear característica [m], que para o fluxo em um cano pode ser calculada através de $L = \frac{4A}{P}$, onde A é a área da seção transversal do cano e P é o perímetro. No caso deste trabalho, a seção do tokamak TCABR tem um valor de $L = 0,4536m$.

4.4 O gerador de malhas Gmsh

Gmsh é um software livre de código aberto, é um gerador de malha de elementos finitos desenvolvido por Christophe Geuzaine e Jean-François Remacle. Ele contém 4 módulos: para descrição geométrica, geração de malha, resolução e pós-processamento. O Gmsh suporta equações paramétricas e possui mecanismos de visualização avançada. Scripts em Gmsh se baseiam na definição de pontos através de suas coordenadas; na definição de linhas, a partir dos pontos criados; na definição de caminhos fechados (closed loops) a partir das linhas; na definição de superfícies a partir dos caminhos fechados; e na definição de volumes a partir das superfícies. Uma vez definido o volume (ou a superfície) com o módulo de descrição geométrica, utiliza-se o módulo de geração de malha para criar a malha de elementos finitos propriamente dita. Esta malha pode ser exportada em algum formato de arquivo padrão de malhas de elementos finitos ou pode ser utilizada pelo módulo de resolução. Módulo este que permite integrar o Gmsh a um software de integração numérica. Por fim o módulo de pós-processamento permite aplicar mapeamentos de cores aos campos escalares, vetoriais ou tensoriais gerados pelo integrador numérico. A entrada de informações em cada um desses módulos pode ser feita de três maneiras: i) através da interface gráfica de forma interativa; ii) através de um arquivo ASCII (.geo) ou iii) utilizando uma API em C, C++ ou Python.

4.5 O código openFOAM

OpenFOAM é uma estrutura de código aberto de suporte essencial para o desenvolvimento de códigos usando uma ampla variedade de funções contidas em bibliotecas C++. Os aplicativos pré-construídos podem ser classificados em duas categorias, conforme mencionado em [16]:

1. Solucionadores, que são usados para resolver um problema específico em mecânica de fluidos ou sólidos.

2. Utilitários, que são usados para realizar várias tarefas, como manipulação e pós-processamento de dados.

Os solucionadores do OpenFOAM podem resolver uma variedade de problemas em dinâmica de fluidos. O OpenFOAM é fornecido com ferramentas de pré e pós-processamento. A interface para o pré e pós-processamento são, em si, utilitários OpenFOAM. Isso garante a manipulação de dados compatível em todos os campos. A dependência de técnicas de programação de computador levou a um foco nos níveis mais baixos da programação. O objetivo é executar problemas matemáticos e físicos complexos, desenvolvendo um código

utilizando a linguagem de programação C++. Isso pode ser alcançado criando altos níveis de código que se assemelham muito à notação de tensor e vetor padrão. No CFD, os campos de velocidade e pressão, em si, são a solução para um conjunto de equações diferenciais parciais (PDEs). Para atingir esse objetivo, é utilizada a metodologia OOPs. É geralmente reconhecido que a Programação Orientada a Objetos (OOP) produz o código que é mais fácil de escrever, validar e manter do que outras técnicas de programação.

As informações geométricas são atribuídas aos campos geométricos pela classe *fvMesh*, que contém informações de vértices, células internas e retalhos de fronteira. Os 11 vértices indicam a geometria da malha e a topologia das células indica a relação entre a ordem na lista e os vértices. As formas primitivas são definidas em tempo de execução e, portanto, o intervalo de formas primitivas pode ser estendido com facilidade, embora o conjunto 3D tetraedro (4 vértices), pirâmide (5 vértices), prisma (6 vértices) e hexaedro (8 vértices) cobrir a maioria das eventualidades. Além disso, cada forma primitiva n-dimensional sabe sobre sua decomposição em formas (n21)-dimensionais, que são usadas na criação de listas de endereçamento como, por exemplo, conectividade célula a célula. As condições de limite são parte integrante do campo. O limite externo do domínio pode ser definido nos patches *fvMesh*. Cada patch carrega uma condição de contorno, que é tratada pelo operador fvm (finite volume method) de maneira apropriada. Todas as condições de contorno, como calculado, valor fixo, gradiente fixo, gradiente zero, simetria, cíclico, etc. que são derivadas de *patchField*, são atendidas de forma diferente. As condições de contorno devem ter a mesma interface, mas implementações diferentes. com base nos elementos básicos, limites adequados para entradas, saídas, paredes, etc, podem ser usados dependendo da situação. Um processador nomeado *patchField* também está disponível para computação paralela. Isso é feito por decomposição de domínio, onde o domínio é dividido em subdomínios com base no número de processadores no sistema. Cada processador lida com um subdomínio por vez. No entanto, processadores separados podem mesclar as informações no solucionador. A vantagem da paralelização é a velocidade de processamento rápida para um pequeno número de processadores e a desvantagem é que ela se torna menos eficiente quando o número de processadores envolvidos é aumentado.

4.6 O software Paraview

ParaView é um aplicativo de visualização e análise de dados multiplataforma de código aberto. Paraview é conhecido e usado em muitas comunidades diferentes, principalmente na comunidade CFD, para analisar e visualizar conjuntos de dados científicos. Ele pode

ser usado para construir visualizações para analisar dados usando técnicas qualitativas e quantitativas. A exploração de dados pode ser feita interativamente em 3D ou programaticamente usando os recursos de processamento do ParaView. O ParaView foi desenvolvido para analisar conjuntos de dados extremamente grandes usando recursos de computação de memória distribuída. Ele pode ser executado em supercomputadores para analisar conjuntos de dados de terabytes de tamanho, bem como em computadores domésticos para dados menores. ParaView é uma estrutura de aplicativo e também um aplicativo pronto para uso. A base de código do ParaView é projetada de forma que todos os seus componentes possam ser reutilizados para desenvolver rapidamente aplicativos verticais. Essa flexibilidade permite que os desenvolvedores do ParaView desenvolvam rapidamente aplicativos com funcionalidade específica para um domínio de problema específico. ParaView é executado em sistemas de processador único e paralelo de memória distribuída e compartilhada.

5 Implementação numérica do modelo MHD visco-resistivo

5.1 O tokamak TCABR

RASCUNHO

O tokamak TCABR utiliza um núcleo de ferro como transformador ôhmico, possui uma parede de aço inox com uma camada condutora de alumínio e tratamento superficial da parede interna com titânio. E possui um filamento de tungstênio que fornece elétrons para ajudar no *breakdown*. Detalhes dos parâmetros geométricos e correntes do tokamak TCABR podem ser vistos na Figura 5 e na Figura 6 podemos ver como o TCABR ficara apos ter as bobinas 3D adicionadas.

PARÂMETRO	SÍMBOLO	UNIDADE	VALOR
Raio maior (geométrico)	R_o	m	0,615
Raio menor (limitador)	a	m	0,18
Razão de aspecto	R_o/a	---	3,4
Corrente de plasma	I_p	A	135.10^3
Duração da corrente	T_d	s	$\leq 150.10^{-3}$
Temperatura iônica no centro	T_{io}	eV	200
Temperatura eletrônica no centro	T_{eo}	eV	600
Densidade eletrônica	n_e	m^{-3}	6.10^{19}
Tempo de confinamento de energia dos elétrons	τ_E	s	1.10^{-3}
Logaritmo de Coulomb	Λ		15
Campo magnético toroidal	B_T	T	1,5
Número atômico efetivo	Z_{ef}	----	1,5 à 4,5
Campo elétrico mínimo de ruptura do gás (obtido à pressão de $2{,}1.10^{-8}$ bar)	$E_{min.}$	V/m	2,0
Gás de trabalho	H_2	Hidrogênio	---
Pressão de base	P_{base}	bar	$1{,}5.10^{-10}$

Figura 5: Dados gerais Tokamak TCABR [1, pg 19]

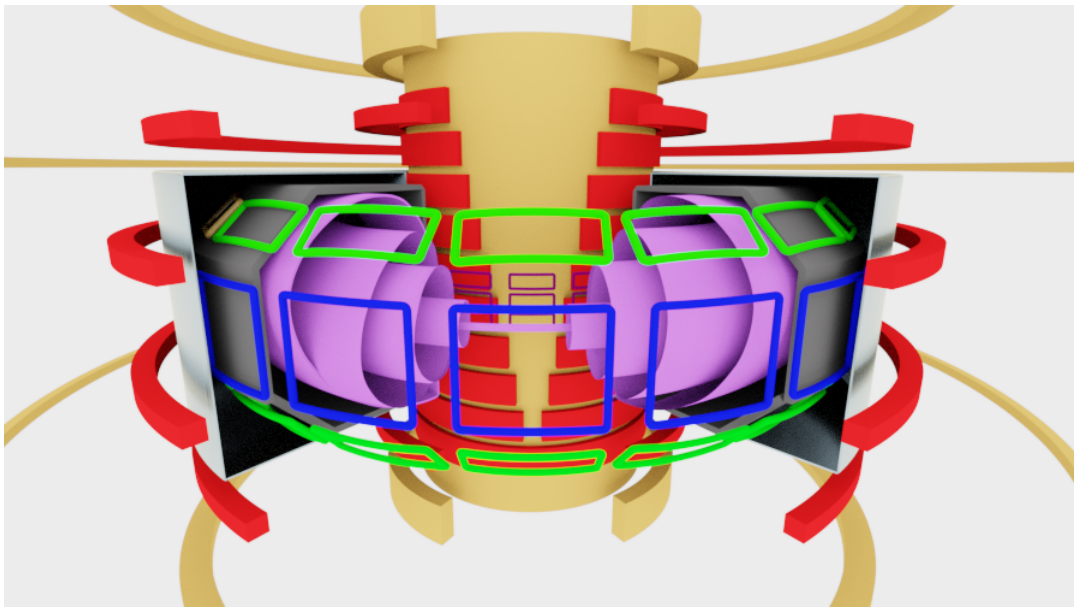

Figura 6: Tokamak TCABR após o upgrate

5.2 Malha computacional

No apêndice A temos o script Gmsh que gera uma malha tridimensional refinada nas bordas, com simetria azimutal e borda computacional com o formato suavizado da borda do tokamak TCABR (dados pelos vetores r, y). Também permite delimitar o numero de extrusões pela variável **numberExtrudes** e o numero de graus de cada extrusão pela variável **ngrau**. Além de permitir o controle do refinamento geral e nas bordas através das variáveis **refiner** e **refiner2**, respectivamente. O script também define as entidades físicas da malha, que neste caso são: Inlet (Entrada do plasma), Outlet (Saída do plasma) e Walls (Paredes), que são representados na figura X. Após escolher estas variáveis no script e ter a malha gerada no Gmsh a malha é exportada, em ASSCI 2, para a pasta da simulação e rodasse pelo terminal o script *gmshToFoam*, que converte de msh para o formato OpemFoam. É preciso mudar o formato das entidades Inlet e Outlet dentro do arquivo boudary, de patch para cyclic uma vez que o valor de cada célula no Inlet é igual ao da célula correspondente no Outlet. Dentro do OpemFoam ainda é rodado o script *checkMesh* para verificar se esta tudo correto com a malha. Tal script gerou a malha mostrada na Figura 7 que será usada paras as simulações feitas neste trabalho. Para conseguir rodar o script *gmshToFoam* foi necessário baixar o Gmsh diretamente do site oficial, pois ao instalar por repositório no linux o programa vem com algum defeito que não permite a execução deste script, impedindo assim a conversão para malha OpenFoam.

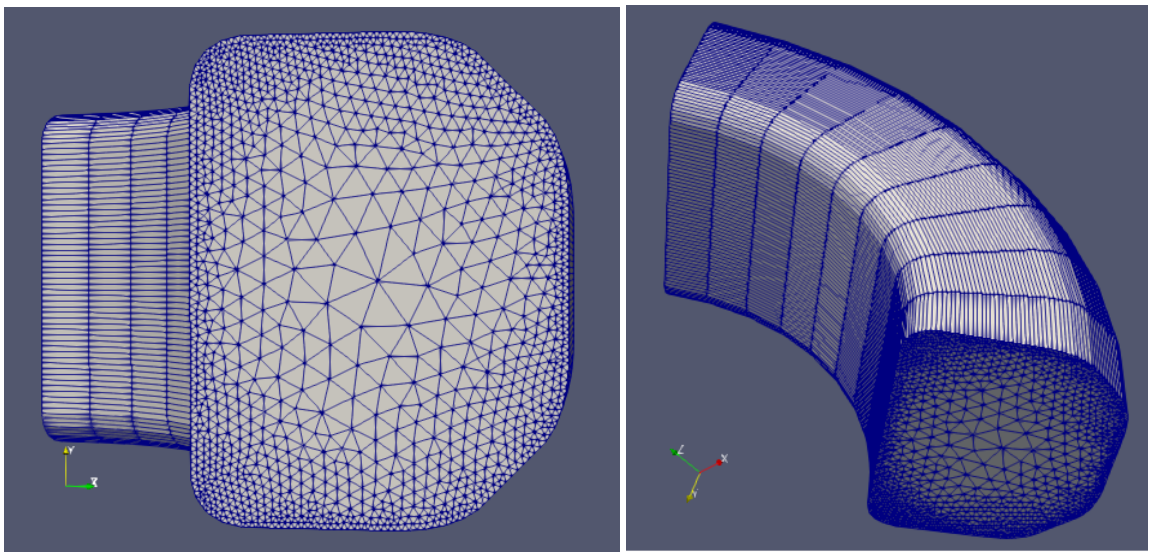

Figura 7: 10 extrusions of 10 degrees each, a 90 degree mesh. Mesh generated by Gmsh of TCABR tokamak.

5.3 Implementação das equações no openFOAM

5.4 Condições de contorno

Foi usado condição *noSlip* para ter velocidade normal 0 nas paredes do domínio computacional, para a pressão p foi setado a condição *zeroGradient* que .... e para ρ foi setado a condão *fixedValue* que é uma condição de dirichelt com um valor fixo para as paredes (o mesmo valor da condição inicial).

5.5 Condições iniciais

Após tudo pronto com a malha vem o preparo das condições iniciais, que não podem ser constantes devido ao uso de coordenadas cilíndricas para a condição inicial e cartesianas para a simulação em sí (uma velocidade na direção x após rotacionar 90º no tokamak vira uma velocidade em y, oque não pode ser representado por constantes). A condição inicial para as velocidades do plasma neste trabalho será uma gaussiana, e devido ao fato de possuirmos um modelo com simetria azimutal nossa gaussiana é melhor escrita em cilíndricas. A obtenção dos valores numéricos dos centros de cada célula da malha é feita através do script *cellCentres* que gera um arquivo texto contendo em cada linha os três componentes de cada centro. Este arquivo de texto precisa ser tratado afim de permitir a execução do cálculo dos valores iniciais. Para isso foi elaborado o script A.1, em MATLAB, que importa os dados, os converte para float, e em seguida para cilíndricas, faz as contas e salva os dados em arquivos que serão importados nas condições iniciais da simulação. O script *cellCentres* foi baixado do GitHub e acoplado a minha biblioteca OpemFoam.

5.6 Validação do modelo para uma situação de solução conhecida

5.7 Modelo de fluido simplificado para teste de convergência

$$\frac{\partial \rho_m}{\partial t} + u_x\frac{\partial \rho_m}{\partial x} + u_y\frac{\partial \rho_m}{\partial y} + u_z\frac{\partial \rho_m}{\partial z} = -\rho_m\left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z}\right) + D\left(\frac{\partial^2 \rho_m}{\partial x^2} + \frac{\partial^2 \rho_m}{\partial y^2} + \frac{\partial^2 \rho_m}{\partial z^2}\right) \tag{38}$$

$$\rho_m\left(\frac{\partial u_x}{\partial t} + u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y} + u_z\frac{\partial u_x}{\partial z}\right) = -\frac{\partial p}{\partial x} + \tag{39}$$

$$+ \mu\rho_m\left(\frac{4}{3}\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2} + \frac{1}{3}\frac{\partial^2 u_y}{\partial x \partial y} + \frac{1}{3}\frac{\partial^2 u_z}{\partial x \partial z}\right) \tag{40}$$

$$\rho_m\left(\frac{\partial u_y}{\partial t}+u_x\frac{\partial u_y}{\partial x}+u_y\frac{\partial u_y}{\partial y}+\ u_z\frac{\partial u_y}{\partial z}\right)=-\frac{\partial p}{\partial y}+ \tag{41}$$

$$+\mu\rho_m\left(\frac{\partial^2 u_y}{\partial x^2}+\frac{4}{3}\frac{\partial^2 u_y}{\partial y^2}+\frac{\partial^2 u_y}{\partial z^2}+\frac{1}{3}\frac{\partial^2 u_x}{\partial y\partial x}+\frac{1}{3}\frac{\partial^2 u_z}{\partial y\partial z}\right) \tag{42}$$

$$\rho_m\left(\frac{\partial u_z}{\partial t}+u_x\frac{\partial u_z}{\partial x}+u_y\frac{\partial u_z}{\partial y}+\ u_z\frac{\partial u_z}{\partial z}\right)=-\frac{\partial p}{\partial z}+ \tag{43}$$

$$+\mu\rho_m\left(\frac{\partial^2 u_z}{\partial x^2}+\frac{\partial^2 u_z}{\partial y^2}+\frac{4}{3}\frac{\partial^2 u_z}{\partial z^2}+\frac{1}{3}\frac{\partial^2 u_x}{\partial z\partial x}+\frac{1}{3}\frac{\partial^2 u_y}{\partial z\partial y}\right) \tag{44}$$

$$\frac{\partial p}{\partial t}+u_x\frac{\partial p}{\partial x}+u_y\frac{\partial p}{\partial y}+u_z\frac{\partial p}{\partial z}=-\Gamma p\left(\frac{\partial u_x}{\partial x}+\frac{\partial u_y}{\partial y}+\frac{\partial u_z}{\partial z}\right)+\kappa\ (\Gamma-1)\left(\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}+\frac{\partial^2 p}{\partial z^2}\right) \tag{45}$$

The magnetic field generate by plasma is presented in Figure 8, and the magnetic field generate by coils is presented in Figure 9.

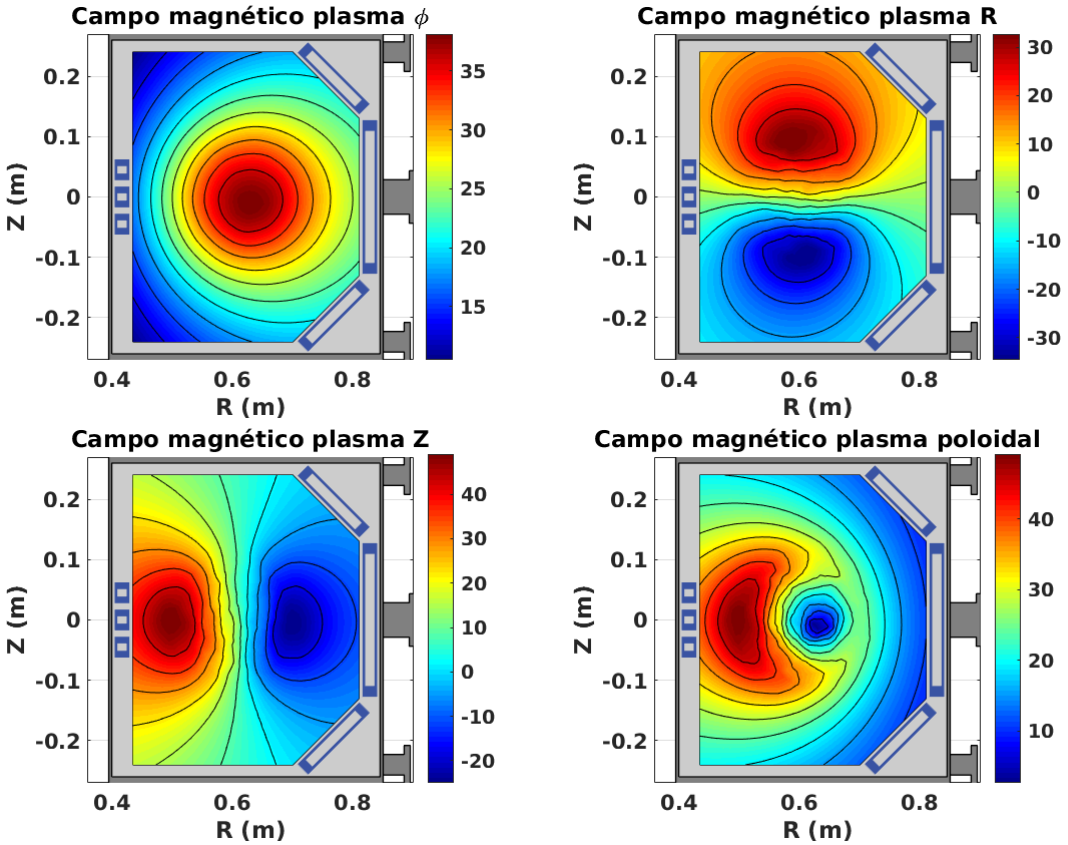

Figura 8: Magnetic field generated by plasma in the TCABR tokamak.

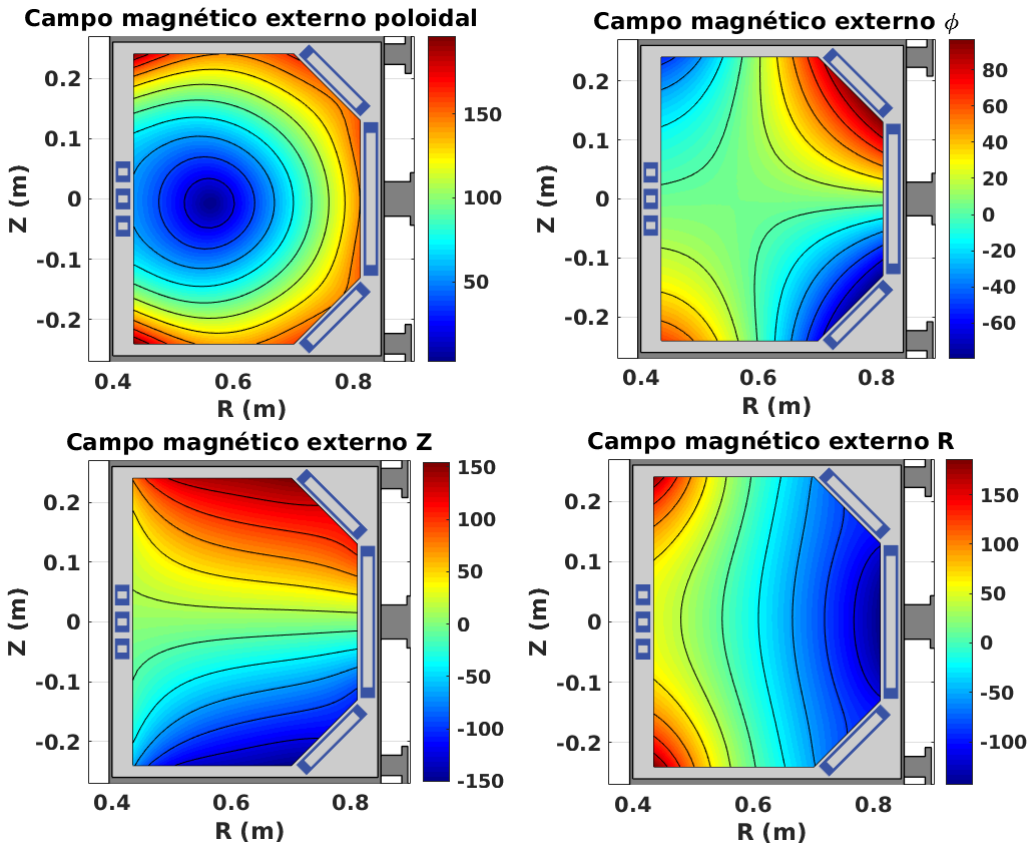

Figura 9: Magnetic field generated by coils of the TCABR tokamak.

6 Simulações de *start-up* no TCABR em vários cenários

6.1 Efeito da pressão do gás de trabalho

6.2 Efeito do campo elétrico toroidal

6.3 Efeito do tipo de gás de trabalho

Poderemos usar hidrogênio, deutério e hélio.

6.4 Efeito do campo magnético toroidal

6.5 Efeito da geometria do campo magnético poloidal

6.6 Comparação das simulações com dados do TCABR

Referências

[1] António Manuel Marques FONSECA. Alguns aspectos do desempenho do tokamak tcabr: Modelamento, simulações e resultados experimentais. *Dissertação (Mestrado em Física Nuclear) - Instituto de Física*, 2000. doi:10.11606/D.43.2000.tde-30082006-121405.Acessoem:2020-10-30.

[2] J.A. Wesson. *Tokamaks*. Clarendon Press, Oxford, 2^{th} edition, 1997.

[3] Alfredo Pironti Marco Ariola. *Magnetic Control of Tokamak Plasmas*. Springer, -, 2008.

[4] Joyeeta Sinha. Plasma breakdown and current formation in single core and doublet configurations on tcv. Technical report, EPFL, 2017.

[5] B. Lloyd *et al.* Low voltage ohmic and electron cyclotron heating assisted startup in DIII-D. *Nuclear Fusion*, 31(11):2031–2053, 1991. https://doi.org/10.1088%2F0029-5515%2F31%2F11%2F001.

[6] Francesco Piras. Extremely shaped plasmas to improve the tokamak concept. *EPFL: Phd Thesis*, page 171, 2011. http://infoscience.epfl.ch/record/163544.

[7] Alexander Gutsol, A. Fridman, and Young Cho. Non-thermal atmospheric pressure plasma. *Advances in Heat Transfer*, 40:1–142, 01 2007.

[8] C. D. Warrick B. Lloyd, P. G. Carolan. *ECRH-assisted start-up in ITER*. Plasma Phys Control. Fusion, 38:1627, 1996.

[9] D Mueller. *The physics of tokamak start-up*. Physics of Plasmas, 20(2013):058101, 2013.

[10] Hyun-Tae Kim, ACC Sips, PC De Vries, JET-EFDA Contributors, et al. Plasma burn-through simulations using the DYON code and predictions for ITER. *Plasma Physics and Controlled Fusion*, 55(12):124032, 2013.

[11] Hyun Tae Kim. Physics and computational simulations of plasma burn-through for tokamak start-up. *Imperial College London: PhD thesis*, 2013. https://doi.org/10.25560/18082.

[12] M Yoo, Yong-Su Na, J Kim, Y An, B Jung, Y Hwang, S Shim, H Lee, and T Hahm. On ohmic breakdown physics in a tokamak. *Proc. 25th IAEA Fusion Energy Conf.(St. Petersburg, Russia, 17 October 2014)*, 2014.

[13] D Mueller. The physics of tokamak start-up. *Physics of Plasmas*, 20(5):058101, 2013.

[14] J.P. Freidberg. *Plasma Physics and Fusion Energy*. Cambridge University Press, London, 2007.

[15] M Darwish F Moukalled L Mangani. The finite volume method in computational fluid dynamics, 2016.

[16] CJ Greenshields. OpenFOAM User Guide. *OpenFOAM Foundation Ltd.*, 2016. http://openfoam.org.

[17] Richard Courant, Kurt Friedrichs, and Hans Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 100(1):32–74, 1928.

Apêndices

A Criação da malha computacional no Gmsh

```
// Gmsh project by Kevi Pegoraro and Gustavo P. Canal 2020

//Variaveis
boxdim = 0.5;
refiner  = 60; //80
refiner2 = 0.12;
gridsize = boxdim/refiner;
Dx=(0.825+0.424)/2;
Dy=0;
gridsize2= gridsize/refiner2;
ngrau=10;
numberExtrudes = 9;

// Entradas do usuario
r={0.4350, 0.4350, 0.4350, 0.4350, 0.4350, 0.4350, 0.4350, 0.4350,
   0.4350, 0.4403, 0.4509, 0.4668, 0.4880, 0.5145, 0.5410, 0.5675,
   0.5940, 0.6205, 0.6470, 0.6704, 0.6907, 0.7079, 0.7220, 0.7330,
   0.7440, 0.7550, 0.7660, 0.7770, 0.7880, 0.7968, 0.8034, 0.8078,
   0.8100, 0.8100, 0.8100, 0.8100, 0.8100, 0.8100, 0.8100, 0.8078,
   0.8034, 0.7968, 0.7880, 0.7770, 0.7660, 0.7550, 0.7440, 0.7330,
   0.7220, 0.7079, 0.6907, 0.6704, 0.6470, 0.6205, 0.5940, 0.5675,
   0.5410, 0.5145, 0.4880, 0.4668, 0.4509, 0.4403, 0.4350, 0.4350,
   0.4350, 0.4350, 0.4350, 0.4350, 0.4350, 0.4350};

y={0.0000,-0.0240,-0.0480,-0.0720,-0.0960,-0.1200,-0.1440,-0.1680,
  -0.1920,-0.2112,-0.2256,-0.2352,-0.2400,-0.2400,-0.2400,-0.2400,
  -0.2400,-0.2400,-0.2400,-0.2378,-0.2334,-0.2268,-0.2180,-0.2070,
  -0.1960,-0.1850,-0.1740,-0.1630,-0.1520,-0.1380,-0.1210,-0.1010,
  -0.0780,-0.0520,-0.0260, 0.0000, 0.0260, 0.0520, 0.0780, 0.1010,
   0.1210, 0.1380, 0.1520, 0.1630, 0.1740, 0.1850, 0.1960, 0.2070,
   0.2180, 0.2268, 0.2334, 0.2378, 0.2400, 0.2400, 0.2400, 0.2400,
```

```
   0.2400, 0.2400, 0.2400, 0.2352, 0.2256, 0.2112, 0.1920, 0.1680,
   0.1440, 0.1200, 0.0960, 0.0720, 0.0480, 0.0240};

numberPoints = 70;

 //Fazendo os pontos
Printf('Number of points %f',numberPoints);
For i In {0:numberPoints-1}
Printf('Making point %f with r= %f and y= %f',i,r[i],y[i]);
Point(i+1) = {r[i], y[i], 0, gridsize};
EndFor

 // Fazendo as linhas
For i In {1:numberPoints-1}
Printf('Making line from %f to %f',i,i+1);
Line(numberPoints+i) = {i,i+1};
EndFor
Printf('Making line from %f to %f',numberPoints,1);
Line(2*numberPoints) = {numberPoints, 1};

 // Fazendo o curve loop
Curve Loop(2*numberPoints+2) = {numberPoints+1:2*numberPoints};

 // Fazendo a superficie inlet
Plane Surface(2*numberPoints+3) = {2*numberPoints+2};

 // Rotacionando-a para que a mesh fique simetrica ao eixo y
Rotate {{0,1,0},{0,0,0},numberExtrudes/2*ngrau*Pi/180.0}
{
Surface{2*numberPoints+3};
}

 // Criando e rotacionando o ponto responsavel pelo refinamento menor
 no centro
N=2*numberPoints+1;
```

```
Printf('Making the point %f that controls the mesh refinement with r=%f
and z=%f',N,Dx,Dy);
Point(N) = {Dx, Dy, 0,gridsize2};
Rotate {{0,1,0},{0,0,0},numberExtrudes/2*ngrau*Pi/180.0}
{
Point{N};
}
Point{N} In Surface{2*numberPoints+3};

 // Criando a primeira extruzão que servira de base para as proximas
 Printf('Extruding inlet');
 Printf('Doing the 1th extrude');
new_entities[] = Extrude {{0,1,0},{0,0,0},-ngrau*Pi/180.0}
{
Surface{2*numberPoints+3};
Layers{1};
Recombine;
};
Physical Surface("walls") = {new_entities[]};
Physical Surface("walls") -= {new_entities[0]};
Physical Volume(3000) = {new_entities[1]};
 // Loop que faz as numberExtrudes
For i In {1:numberExtrudes-1}
         Printf('Doing the %fth extrude',i+1);
new_entities[] = Extrude {{0,1,0},{0,0,0},-ngrau*Pi/180.0}
{
Surface{new_entities[0]};
Layers{1};
Recombine;
};
Physical Surface("walls") += {new_entities[]};
Physical Surface("walls") -= {new_entities[0]};
Physical Volume(3000) += {new_entities[1]};
EndFor
```

```
 // Criando os grupos físicos
Physical Surface("inlet") = {2*numberPoints+3};
Physical Surface("outlet") = {new_entities[0]};
```

A.1 Preparação das condições iniciais

```
clear all
fn = '_90';
name=['points' fn '.txt'];
A = importdata(name);
A = string(A);
A = erase(A,'(');
A = erase(A,')');
A = split(A);
A = double(A);
[theta,rho,z] = cart2pol(A(:,1),A(:,3),A(:,2));
% Calculando velocidade
L = 10e3*exp(-50*(((0.825+0.424)/2-rho).^2+(z).^2));
%for i=1:n_extrude eval([''])
Uz = cos(theta).*L;
Ux = sin(theta).*L;
name=['Uz_' fn]; save('-ascii',name, 'Uz')
name=['Ux_' fn]; save('-ascii',name, 'Ux')
%end
% Calculando p
Lp=L/100;
name=['Lp' fn]; save('-ascii',name,'Lp')
% Calculando p
Lrho=L/1000;
name=['Lrho' fn]; save('-ascii',name,'Lrho')
```

B Implementação das equações no OpenFOAM

Abaixo temos a implementação do solver oneFluidPFoam, do modelo simplificado

```
/*---------------------------------------------------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Copyright(C) 2011-2018 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------

Application
    ondefluidFoam

Description
    Solves .

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

#include "fvCFD.H"
#include "fvOptions.H"
//#include "simpleControl.H"
#include "pisoControl.H"
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    #include "setRootCaseLists.H"
    #include "createTime.H"
    #include "createMesh.H"

  //  simpleControl simple(mesh);
    pisoControl piso(mesh);

    #include "createFields.H"

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

```
Info<< nl << "Starting time loop" << endl;

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

        // Derivadas espaciais de p
        volVectorField dp=fvc::grad(p);

        volScalarField dpdx=dp.component(0);
        volScalarField dpdy=dp.component(1);
        volScalarField dpdz=dp.component(2);

        // Derivadas espaciais do rho
        volVectorField drho=fvc::grad(rho);

        volScalarField drhodx=drho.component(0);
        volScalarField drhody=drho.component(1);
        volScalarField drhodz=drho.component(2);

        // Derivadas espaciais de U
        volVectorField dUx=fvc::grad(Ux);
       volVectorField dUy=fvc::grad(Uy);
        volVectorField dUz=fvc::grad(Uz);

        volScalarField dUxdx=dUx.component(0);
        volScalarField dUxdy=dUx.component(1);
        volScalarField dUxdz=dUx.component(2);

        volScalarField dUydx=dUy.component(0);
        volScalarField dUydy=dUy.component(1);
        volScalarField dUydz=dUy.component(2);

        volScalarField dUzdx=dUz.component(0);
        volScalarField dUzdy=dUz.component(1);
```

```
volScalarField dUzdz=dUz.component(2);

// Derivadas segunda de Ux
volVectorField d2Uxdx=fvc::grad(dUxdx);

volScalarField d2Uxdxdx=d2Uxdx.component(0);
volScalarField d2Uxdxdy=d2Uxdx.component(1);
volScalarField d2Uxdxdz=d2Uxdx.component(2);

volVectorField d2Uxdy=fvc::grad(dUxdy);

volScalarField d2Uxdydx=d2Uxdy.component(0);
volScalarField d2Uxdydy=d2Uxdy.component(1);
volScalarField d2Uxdydz=d2Uxdy.component(2);

volVectorField d2Uxdz=fvc::grad(dUxdz);

volScalarField d2Uxdzdx=d2Uxdz.component(0);
volScalarField d2Uxdzdy=d2Uxdz.component(1);
volScalarField d2Uxdzdz=d2Uxdz.component(2);

// Derivadas segunda de Uy
volVectorField d2Uydx=fvc::grad(dUydx);

volScalarField d2Uydxdx=d2Uydx.component(0);
volScalarField d2Uydxdy=d2Uydx.component(1);
volScalarField d2Uydxdz=d2Uydx.component(2);

volVectorField d2Uydy=fvc::grad(dUydy);

volScalarField d2Uydydx=d2Uydy.component(0);
volScalarField d2Uydydy=d2Uydy.component(1);
volScalarField d2Uydydz=d2Uydy.component(2);

volVectorField d2Uydz=fvc::grad(dUydz);
```

```
volScalarField d2Uydzdx=d2Uydz.component(0);
volScalarField d2Uydzdy=d2Uydz.component(1);
volScalarField d2Uydzdz=d2Uydz.component(2);

// Derivadas segunda de Uz
volVectorField d2Uzdx=fvc::grad(dUzdx);

volScalarField d2Uzdxdx=d2Uzdx.component(0);
volScalarField d2Uzdxdy=d2Uzdx.component(1);
volScalarField d2Uzdxdz=d2Uzdx.component(2);

volVectorField d2Uzdy=fvc::grad(dUzdy);

volScalarField d2Uzdydx=d2Uzdy.component(0);
volScalarField d2Uzdydy=d2Uzdy.component(1);
volScalarField d2Uzdydz=d2Uzdy.component(2);

volVectorField d2Uzdz=fvc::grad(dUzdz);

volScalarField d2Uzdzdx=d2Uzdz.component(0);
volScalarField d2Uzdzdy=d2Uzdz.component(1);
volScalarField d2Uzdzdz=d2Uzdz.component(2);


// Definindo as equações
while (piso.correctNonOrthogonal())
{

   // Equação da continuidade
   fvScalarMatrix rhoEqn
   (
       fvm::ddt(rho)
       + Ux*drhodx + Uy*drhody + Uz*drhodz
       + rho*(dUxdx + dUydy + dUzdz)
```

```
);

// rhoEqn.relax();
// fvOptions.constrain(rhoEqn);  // fvOptions(rho)
rhoEqn.solve();
// fvOptions.correct(rho);

// Componente x da equação de momento
fvScalarMatrix UxEqn
(
    rho*(fvm::ddt(Ux)
    + Ux*dUxdx + Uy*dUxdy + Uz*dUxdz)
    == rho*(mu*(4/3*d2Uxdxdx + d2Uxdydy + d2Uxdzdz)
    + mu/3*(d2Uydxdy + d2Uzdxdz)) - dpdx
);

// UxEqn.relax();
UxEqn.solve();

// Componente y da equação de momento
fvScalarMatrix UyEqn
(
    rho*(fvm::ddt(Uy)
    + Ux*dUydx + Uy*dUydy + Uz*dUydz)
    == rho*(mu*(d2Uydxdx + 4/3*d2Uydydy + d2Uydzdz)
    + mu/3*(d2Uxdydx + d2Uzdydz)) - dpdy
);

// UyEqn.relax();
UyEqn.solve();

// Componente z da equação de momento
fvScalarMatrix UzEqn
(
    rho*(fvm::ddt(Uz)
```

```
                + Ux*dUzdx + Uy*dUzdy + Uz*dUzdz)
                == rho*(mu*(d2Uzdxdx + d2Uzdydy + 4/3*d2Uzdzdz)
                + mu/3*(d2Uxdzdx + d2Uydzdy)) - dpdz
            );

            // UzEqn.relax();
            UzEqn.solve();

            // Equação da energia
            fvScalarMatrix pEqn
            (
                fvm::ddt(p)
                + Ux*dpdx + Uy*dpdy + Uz*dpdz
                == -gama*p*(dUxdx + dUydy + dUzdz)
            );

            // pEqn.relax();
            pEqn.solve();
        }


        runTime.write();
    }

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

    return 0;
}
```

Create fields

```
    Info<< "Reading field rho\n" << endl;

volScalarField rho
(
```

```
    IOobject
    (
        "rho",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field p\n" << endl;

volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);

Info<< "Reading field Ux\n" << endl;

volScalarField Ux
(
    IOobject
    (
        "Ux",
        runTime.timeName(),
        mesh,
```

```
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
Info<< "Reading field Uy\n" << endl;

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

volScalarField Uz
(
    IOobject
    (
        "Uz",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

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

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


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

dimensionedScalar mu
(
    transportProperties.lookup("mu")
);
Info<< "Reading diffusivity gama\n" << endl;

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

//#include "createPhis.H"

//#include "createFvOptions.H"
```