UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

SIMULAÇÃO NUMÉRICA DA CONVECÇÃO NATURAL EM GERADORES DE VAPOR

Pedro de Almeida Pacheco Amaral

São Paulo
2005

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

1 INTRODUÇÃO

O projeto tem como objetivo, a modelagem e simulação da circulação de água-vapor em um gerador de vapor aquotubular.

O fenômeno a ser estudado é a convecção natural. Ela se dá devido à transferência do calor proveniente dos gases quentes de combustão por radiação (fornalha) e convecção (fornalha e feixes convectivos). A medida que a água nos tubos da caldeira, já em estado de saturação, recebe essa carga térmica, ela vai, parcialmente, se tornando vapor e conseqüentemente a sua densidade vai diminuindo. O tubulão de vapor se encontra acima dos tubos que estão sendo aquecidos e está conectado a eles. Nele o vapor se separa da água em estado líquido e sai da caldeira normalmente em direção ao superaquecedor. A mesma massa que é retirada deve ser reposta na forma de água saturada, vinda do economizador. Assim, a mistura água-vapor formada nos tubos, menos densa, se encontra abaixo da água em estado líquido do tubulão de modo que ela tende a subir pela tubulação. Já a água do tubulão, mais densa, tende a descer pelos tubos que não são aquecidos (downcomers), também conectados a ele. Cria-se assim uma circulação causada por uma força resultante da diferença entre as densidades do fluido no decorrer do seu percurso. Essa força atua no fluido no sentido da circulação e a força de atrito do fluido com o tubo atua no sentido contrário. Em regime permanente elas têm mesma intensidade o que resulta em um escoamento de circulação constante.

Como a tubulação do gerador de vapor deve ser mantida a uma temperatura muito abaixo da temperatura dos gases de combustão, projeta-se um gerador para que, no lado frio, haja apenas ebulição nucleada e em hipótese alguma ebulição em película já que esta causaria um indesejado aumento da temperatura do metal. Sendo assim, a parede do tubo troca calor com água em estado líquido e não vapor (com exceção dos superaquecedores, mas neles os gases de combustão já tiveram sua temperatura diminuída e não há radiação proveniente da chama). Obtém-se um coeficiente de troca

térmica muito maior para o lado frio que para o lado quente dos gases (estes têm um calor específico muito menor). Por isso, a resistência térmica do sistema depende da resistência térmica convectiva dos gases e condutiva da parede do tubo. Pelo fato do projeto visar a modelagem do lado frio do trocador de calor (água-vapor), a carga térmica trocada no sistema será uma condição de contorno do modelo já que é o lado quente quem controla a troca de calor.

O modelo da caldeira a ser simulado será composto por um tubulão, um tubulão de lama, dois risers e um downcomer. Os risers receberão cargas térmicas diferentes. O mais aquecido representa os tubos da fornalha, que recebe calor por radiação e convecção. O segundo riser, menos aquecido, representa o feixe convectivo, onde só ocorre convecção. O downcomer representa toda a tubulação de descida da água do tubulão de vapor ao tubulão de lama. Ele não recebe carga térmica.

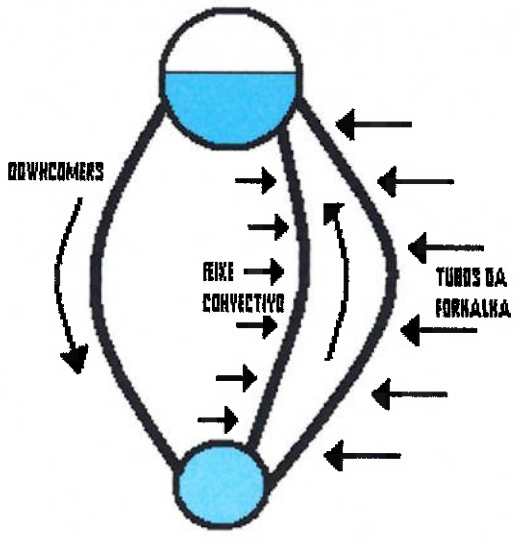

Figura 1. Modelo do gerador de vapor a ser simulado.

2 AVALIAÇÃO DA VIABILIDADE TÉCNICO-ECONÔMICA

No projeto de um gerador de vapor, deve-se ter um prognóstico do funcionamento do equipamento de modo que, quando construído, ele tenha a performance desejada. Além disso, no processo do projeto do gerador, é necessário que uma configuração ótima seja escolhida entre as alternativas possíveis, devendo-se prever o funcionamento de todas elas. Ainda no processo de projeto do gerador, o principal requisito é que a ebulição nucleada seja mantida em todo o sistema para qualquer condição de operação de modo a garantir que a tubulação aquecida esteja sempre em contato com água no seu estado líquido. Um critério de projeto é a fração volumétrica de vapor permitida (%SBV). Esse limite é função de variáveis como pressão e fluxos de calor e massa. A determinação da %SBV e a verificação de que o título da mistura no interior dos tubos irá respeitar esse limite devem ser feitas ainda da etapa de projeto do gerador de vapor. O prognóstico do funcionamento do equipamento também é importante para seu controle e para que sejam evitados possíveis riscos como o aumento da pressão ou a queda do nível de água do tubulão, que podem danificar o equipamento. Muitas vezes é necessário, também, prever o funcionamento de caldeiras já em operação para condições diferentes daquelas em que ela está operando.

O prognóstico de processos de transferência de calor e escoamento de fluidos pode ser feito de duas maneiras: método experimental ou cálculo teórico.

A maneira mais confiável normalmente é dada pelo método experimental, através das medições do processo real. Uma simulação usando-se um protótipo em escala real normalmente tem um custo proibitivo. A alternativa é o uso de protótipos em pequena escala e a transformação dos dados para a escala real, mas as regras para tal transformação raramente estão disponíveis. Além disso, nem todos os aspectos dos modelos em escala real são simulados pelos protótipos em pequena escala.

O cálculo teórico traz inúmeras vantagens em relação ao método experimental que o torna viável.

Seu custo é incomparável a um procedimento experimental já que só é necessário o uso de um computador. Quanto mais complexo for o gerador de vapor a ser simulado, maior será essa diferença de custo, já que o preço de um modelo físico aumenta com sua complexidade.

A velocidade com que a simulação é feita é muito maior para o cálculo teórico. Pode-se, num curto espaço de tempo, simular a operação do equipamento com inúmeras configurações para inúmeras condições de modo a otimizá-lo.

Na simulação computacional pode-se obter todos os valores das grandezas relevantes (pressão, temperatura, densidade e velocidade entre outras) para todo o domínio de interesse. Além disso, não há nenhuma intervenção no sistema causada por aparelhos de medição (tubo de pitot, por exemplo). Deve-se lembrar que esses aparelhos, usados no método experimental, são passíveis de incertezas.

Por fim, pode-se, através da simulação numérica, trabalhar com condições que em modelos físicos são difíceis de lidar como temperaturas e pressões muito altas.

3 SOLUÇÃO NUMÉRICA

Para a simulação da convecção natural, será usado o método do volume finito já que ele é bastante eficiente para simulação de processos de transporte de massa. O sistema a ser calculado será divido em determinado número (N) de células (volumes de controle). Desse modo haverá sempre um volume de controle ao redor de cada ponto de cálculo (figura 2). As equações diferenciais serão discretizadas para cada um deles.

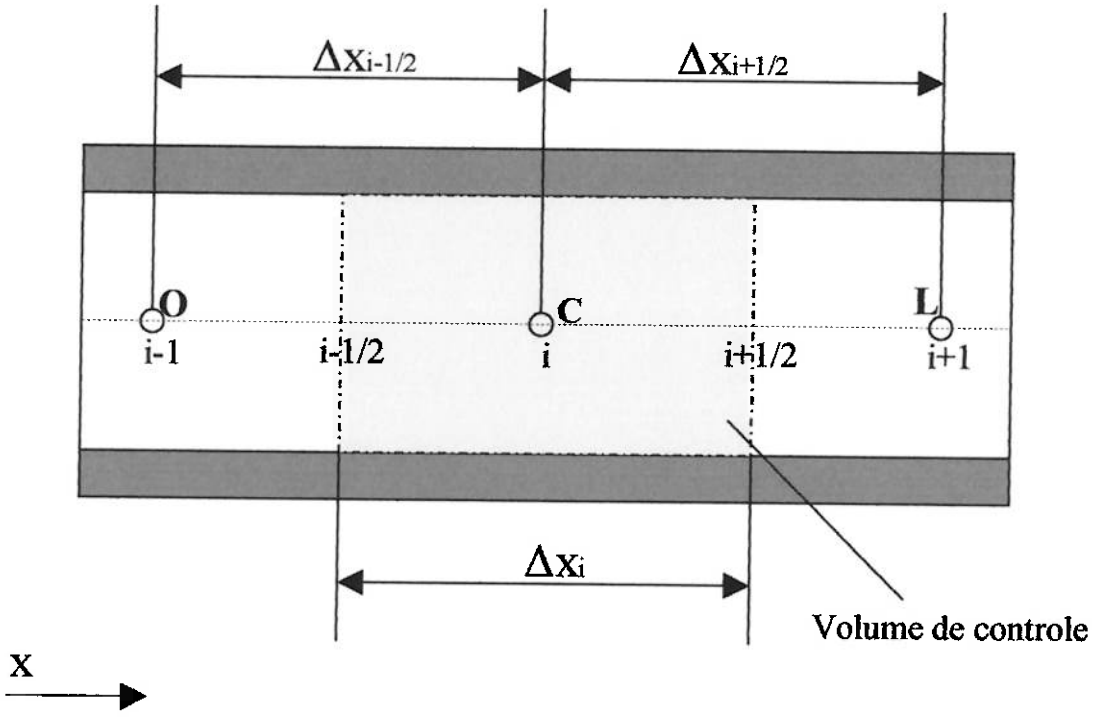

Figura 2. Representação de uma célula convencional

Em [4] Patankar introduz o algoritmo SIMPLE (Semi Implicit Method for Pressure Linked Equation), um processo para se calcular o perfil de escoamento. A necessidade do desenvolvimento de um método especial se dava porque não se podia calcular o perfil das velocidades sem se ter o perfil das pressões. Isso porque a velocidade pode ser obtida através do balanço de momento do qual faz parte o gradiente de pressão. Sabe-se que se um perfil de pressões correto é substituído no balanço de momento, obtém-se um perfil de velocidades que satisfaz a equação da continuidade. Criou-se então o método SIMPLE que é um processo iterativo que através de um perfil de pressões aleatório converge para o perfil de escoamento.

As operações se iniciam com a determinação de um perfil de pressões aleatório. Na seqüência resolve-se o balanço de momento para se determinar um perfil de velocidades. Calcula-se o valor da correção da pressão através da combinação dos balanços de massa e momento aplicados ao perfil de velocidades que foi determinado (essas equações serão discutidas com detalhe a seguir). Depois de calculado o valor da correção, corrige-se o perfil de pressões. Em seguida, calcula-se um novo valor para a velocidade através do balanço de massa e dos valores da correção da pressão. Resolve-se todas as equações que determinem outras grandezas que influenciam no escoamento (como o balanço de energia para determinação da temperatura). Se uma grandeza em particular não tiver essa influência ela deve ser calculada apenas uma vez no final, após a solução. Por fim verifica-se se a solução é satisfatória. Caso não seja, o valor atualizado do perfil de pressões será usado no primeiro passo. O processo vai se repetindo até convergir para uma solução satisfatória.

No algoritmo SIMPLE, para que não se tenha o problema de tomar equivocadamente um campo de pressão ou de velocidades em onda (alternando valores) como campo constante, calcula-se a velocidade nas margens das células enquanto que todas as outras grandezas são calculadas no seu centro. Por isso, para o balanço de momento, o volume de controle será uma célula deslocada (figura 3) enquanto para a determinação das outras propriedades será usada a célula convencional da figura 2.

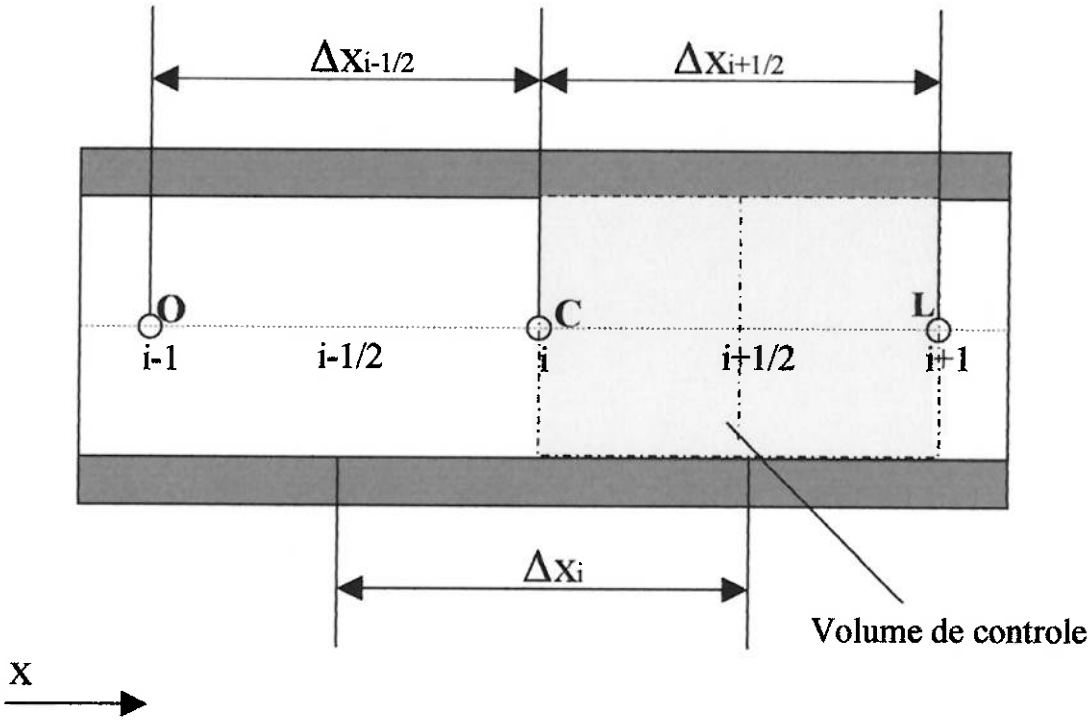

Figura 3. Representação de uma célula deslocada.

4 APLICAÇÃO DA SOLUÇÃO PARA CONVECÇÃO NATURAL EM G.V.

A aplicação da solução de Patankar especificamente para a convecção natural em geradores de vapor foi desenvolvida por Ponweiser em [5]. Dessa referência vem a dedução apresentada abaixo.

Aplicação em uma grandeza genérica

Para a dedução da modelagem do sistema utilizando-se o método de volumes finitos, Ponweiser partiu da equação geral de transporte, considerando-se que todas as grandezas vetoriais têm a direção do eixo do tubo. Na equação o termo à esquerda está relacionado ao acumulo de massa enquanto o primeiro, segundo e último termo à direita estão relacionados à convecção, difusão e ao termo fonte respectivamente:

$$\int_V \frac{\partial}{\partial t}(\rho\phi)dV = -\int_V \frac{\partial}{\partial x}(\rho\omega\phi)dV + \int_V \frac{\partial}{\partial x}\Gamma_\phi \frac{\partial\phi}{\partial x}dV + \int_V S_\phi dV \quad (1)$$

Com a integral de Gauss, pode-se converter a integral no volume dos termos convectivos e difusivos em integral na superfície e obtém-se:

$$\int_V \frac{\partial}{\partial t}(\rho\phi)dV = -\int_A (\rho\omega\phi)dA + \int_A \Gamma_\phi \frac{\partial\phi}{\partial x}dA + \int_V S_\phi dV \quad (2)$$

Como não há troca de massa na superfície do volume de controle cilíndrico, a integral na superfície pode ser reduzida a uma diferença entre as trocas nas superfícies frontais. Suposta uma distribuição homogênea de todas as grandezas de estado e transporte sobre a secção transversal do tubo:

$$\int_x \frac{\partial}{\partial t}(\rho\phi)Adx = (\rho\omega\phi A)_{i-1/2} - (\rho\omega\phi A)_{i+1/2} - \left(\Gamma_\phi \frac{\partial\phi}{\partial x}A\right)_{i-1/2} + \left(\Gamma_\phi \frac{\partial\phi}{\partial x}A\right)_{i+1/2} + \int_x S_\phi A dx \quad (3)$$

UNIVERSIDADE DE SÃO PAULO ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

SIMULAÇÃO NUMÉRICA DA CONVECÇÃO NATURAL EM GERADORES DE VAPOR

Pedro de Almeida Pacheco Amaral

Orientador: Profº Dr. M. Pimenta

Área de Concentração
Engenharia Mecânica

São Paulo
2005

Não será pressuposta uma secção transversal constante pois ela pode variar na rede de tubos a ser modelada, desse modo cada célula terá sua área de secção (A_i). Além disso, o valor das grandezas de estado e transporte no ponto de cálculo i será considerado como o valor médio na célula correspondente:

$$\frac{\partial}{\partial t}(\rho\phi)_i A_i dx = (\rho\omega\phi A)_{i-1/2} - (\rho\omega\phi A)_{i+1/2} - \left(\Gamma_\phi \frac{\partial\phi}{\partial x} A\right)_{i-1/2} + \left(\Gamma_\phi \frac{\partial\phi}{\partial x} A\right)_{i+1/2} + (S_\phi A)_i \Delta x_i \quad (4)$$

É vantajoso, sintetizar os termos difusivos e convectivos:

$$J = \rho\omega\phi - \Gamma_\phi \frac{\partial\phi}{\partial x} \quad (5)$$

A derivada no termo difusivo será substituida por uma diferença de quocientes:

$$J_{i-1/2} = F_{i-1/2}\phi_{i-1/2} - D_{i-1/2}(\phi_i - \phi_{i-1}) \quad (6)$$

$$J_{i+1/2} = F_{i+1/2}\phi_{i+1/2} - D_{i+1/2}(\phi_{i+1} - \phi_i) \quad (7)$$

onde

$$F_{i-1/2} = (\rho\omega)_{i-1/2} \quad (8)$$

$$F_{i+1/2} = (\rho\omega)_{i+1/2} \quad (9)$$

e

$$D_{i-1/2} = \frac{\Gamma_{i-1/2}}{\Delta x_{i-1/2}} \quad (10)$$

$$D_{i+1/2} = \frac{\Gamma_{i+1/2}}{\Delta x_{i+1/2}} \quad (11)$$

Discretizando-se a derivada no tempo da equação (4) sob a hipótese de que $\rho_i\phi_i=(\rho\phi)_i$ e dividindo-se o termo fonte em uma parte constante e outra proporcional, obtém-se:

$$\frac{(\rho_i\phi_i - \rho_i^0\phi_i^0)A_i\Delta x_i}{\Delta t} = J_{i-1/2}A_{i-1/2} - J_{i+1/2}A_{i+1/2} + (S_{Ci} + S_{Pi}\phi_i)A_i\Delta x_i \quad (12)$$

A importância da divisão do termo fonte em uma parte constante e outra proporcional será discutida mais tarde. Na discretização no tempo da equação (4), as variáveis do ponto de tempo t-Δt, são identificadas com "0" e outras variáveis pertencem ao ponto de tempo t.

Para obtenção da forma desejada, deve-se combinar a equação (12) com a seguinte equação do balanço de massa:

$$\int_V \frac{\partial \rho}{\partial t} dV = -\int_A \rho(vdA) \quad (13)$$

Obtém-se, então:

$$\int_x \frac{\partial}{\partial t}(\rho) A dx = (\rho \omega A)_{i-1/2} - (\rho \omega A)_{i+1/2} \quad (14)$$

Análogo ao que foi feito antes para todas as grandezas, toma-se o valor da densidade no ponto i como a média da célula de cálculo, o que leva a:

$$\frac{\partial}{\partial t}(\rho)_i A_i \Delta x_i = (\rho \omega A)_{i-1/2} - (\rho \omega A)_{i+1/2} \quad (15)$$

Da discretização no tempo:

$$\frac{(\rho_i - \rho_i^o) A_i \Delta x_i}{\Delta t} = (\rho \omega A)_{i-1/2} - (\rho \omega A)_{i+1/2} \quad (16)$$

Através da multiplicação da equação (16) por ϕ_i e a subtração da equação (12), consegue-se:

$$(\phi_i - \phi_i^o) \frac{\rho_i^0 A_i \Delta x_i}{\Delta t} = (J_{i-1/2} - F_{i-1/2}\phi_i) A_{i-1/2} - (J_{i+1/2} - F_{i+1/2}\phi_i) A_{i+1/2} + (S_{Ci} + S_{Pi}\phi_i) A_i \Delta x_i \quad (17)$$

Nesse ponto, deve-se observar que para o algoritmo SIMPLE, Ponweiser usa o esquema UPWIND na determinação dos valores que se encontram entre os pontos de cálculo do volume de controle. Isso significa que as grandezas de estado no volume de

RESUMO

O presente trabalho tem o objetivo de simular numericamente o escoamento resultante da convecção natural em um gerador de vapor. O modelo a ser estudado é um gerador composto por um tubulão, um tubulão de lama, um tubo de descida de água e dois tubos aquecidos. O texto apresenta um estudo de viabilidade que discute a importância da simulação computacional e suas vantagens em relação aos métodos experimentais. Ele apresenta o método numérico a ser utilizado e a modelagem do sistema. O trabalho traz as condições de contorno a serem usadas e o método usado para a resolução dos sistemas de equações necessário a simulação, além do equacionamento das propriedades da água. Ele finaliza com a simulação de um gerador de vapor para diferentes condições.

ABSTRACT

The present report has the objective to simulate numerically the flow that results from the natural convection in a steam generator. The model to be studied is a generator composed by a drum, a collector, one downcomer and two risers. The text presents a viability study that discusses the importance of the computational simulation and its advantages in comparison with the experimental methods. It presents the numerical method to be used and the model of the system. It brings the boundary conditions to be used and the method used to solve the system of equations necessary in the simulation as well as the equations of the water properties. It finishes simulating a steam generator under different conditions.

controle serão consideradas constantes. As grandezas na margem jusante do volume de controle serão igualadas com aquela localizadas no seu interior:

$$\phi_{i-1/2} = \phi_{i-1}$$

se

$$F_{i-1/2} > 0$$

ou

$$\phi_{i-1/2} = \phi_i$$

se

$$F_{i-1/2} < 0$$

O mesmo é válido para a outra a margem .

Na utilização do esquema UPWIND pode-se substituir os termos, de modo que os fluxos totais e convectivos sejam reunidos, da seguinte forma:

$$J_{i-1/2} - F_{i-1/2}\phi_i = O_i(\phi_{i-1} - \phi_i) \tag{18}$$

$$J_{i+1/2} - F_{i+1/2}\phi_i = L_i(\phi_i - \phi_{i+1}) \tag{19}$$

Com os coeficientes:

$$O_i = (D_{i-1/2} + [[F_{i-1/2}, 0]])A_{i-1/2} \tag{20}$$

$$L_i = (D_{i+1/2} + [[-F_{i+1/2}, 0]])A_{i+1/2} \tag{21}$$

O operador [[A,B]] fornece o maior valor entre A e B.

Substituindo-se as equações (18) e (19) na equação 17, tem-se:

$$\phi_i(O_i + L_i + \frac{\rho_i^0 A_i \Delta x_i}{\Delta t} - S_{Pi} A_i \Delta x_i) = O_i\phi_{i-1} + L_i\phi_{i+1} + S_{Ci} A_i \Delta x_i + \frac{\rho_i^0 A_i \Delta x_i}{\Delta t}\phi_i^0 \tag{22}$$

Pode-se reescrevê-la da seguinte forma:

$$C_i\phi_i = O_i\phi_{i-1} + L_i\phi_{i+1} + k_i \tag{23}$$

Onde:

$$C_i^0 = \frac{\rho_i^0 A_i \Delta x_i}{\Delta t} \tag{24}$$

$$k_i = S_{Ci} A_i \Delta x_i + C_i^0 \phi_i^0 \tag{25}$$

$$C_i = O_i + L_i + C_i^0 - S_{Pi} A_i \Delta x_i \tag{26}$$

Como exigência fundamental, que deve ser satisfeita para a convergência do método, Patankar postulou em [4] que os coeficientes Ci, Oi e Li devem ser positivos. Para garantir isso, o termo fonte deve ser linearizado de tal modo que sua parte proporcional Sp seja negativa.

Para o cálculo de uma grandeza genérica de estado ou transporte ϕ em todos os N pontos de cálculo dos tubos discretizados, deve-se formular a equação (23) para cada um deles. Como a solução de cada ponto de cálculo é dependente da solução de seus dois vizinhos, obtém-se um sistema de equações com uma estrutura tridiagonal:

$$\begin{bmatrix} C_1 & -L_1 & & & \\ -O_2 & C_2 & -L_2 & & \\ & \cdots & \cdots & \cdots & \\ & & -O_{N-1} & C_{N-1} & -L_{N-1} \\ & & & -O_N & C_N \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \cdots \\ \phi_{N-1} \\ \phi_N \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ \cdots \\ k_{N-1} \\ k_N \end{bmatrix} \tag{27}$$

Cálculo dos perfis de velocidade e pressão

Como já foi dito, a dificuldade na simulação numérica de escoamentos está no cálculo dos perfis de velocidade e pressão que são descritos pela combinação dos balanços de massa e momento. No algoritmo SIMPLE, esses balanços, assim com o da energia, serão calculados separadamente para cada ponto de cálculo.

O balanço de momento é calculado para a célula deslocada já que a velocidade no algoritmo SIMPLE é calculada nas margens do volume de controle convencional. A equação discretizada para o balanço do momento na célula deslocada pode ser obtida substituindo-se a velocidade no lugar de ϕ e ajustando-se os coeficientes na equação (23). A força de atrito no volume de controle, responsável pela perda de carga no escoamento e a força da gravidade fazem parte do termo fonte. A força da pressão que atua na célula por ambos os lados fica fora do termo. Com isso obtém-se a equação para se determinar a velocidade no volume de controle:

$$l_i \omega_{i+1/2} = o_i \omega_{i-1/2} + l_i^l \omega_{i+3/2} + k_i^e + (p_i - p_{i+1}) A_{i+1/2} \quad (28)$$

Pelo fato de que a célula está deslocada, pode-se observar que o coeficiente l_i refere-se ao ponto de cálculo i+1/2, o coeficiente o_i ao ponto i-1/2 e o ponto l_i^l ao ponto i+3/2.

Os coeficientes são análogos aos das equações (24), (25) e (26):

$$l_i^0 = \frac{\rho_{i+1/2}^0 A_{i+1/2} \Delta x_{i+1/2}}{\Delta t} \quad (29)$$

$$k_i^e = S_{Ci}^p A_{i+1/2} \Delta x_{i+1/2} + l_i^0 \omega_{i+1/2}^0 \quad (30)$$

$$o_i = [[(\rho\omega)_{i-1/2}, 0]] A_{i-1/2} \quad (31)$$

$$l_i^l = [[-(\rho\omega)_{i+3/2}, 0]] A_{i+3/2} \quad (32)$$

$$l_i = o_i + l_i^l + l_i^0 - S_{Pi}^p A_{i+1/2} \Delta x_{i+1/2} \quad (33)$$

Não existem termos difusivos nos termos o e l_i pois o momento é exclusivamente convectivo.

As diferenças de pressão conseqüente da gravidade (ΔP_H) e do atrito (ΔP_R) fazem parte do termo fonte:

$$S_{Ci}^p = \Delta p_{Hi+1/2} \quad (34)$$

$$S_{Pi}^{p} = -\frac{|\Delta p_{Ri+1/2}|}{|\omega_{i+1/2}|} \quad (35)$$

Correção da velocidade

Através da equação (28), pode-se obter o perfil de velocidades exato somente se o perfil de pressão exato for conhecido. Como já foi dito, durante a iteração os valores dos perfis serão soluções aproximadas. Elas serão sinalizadas por "*". As soluções exatas da pressão e da velocidade vêm da solução aproximada somada a uma correção que será marcada por " ' ":

$$p = p^{*} + p' \quad (36)$$

$$\omega = \omega^{*} + \omega' \quad (37)$$

Substituindo-se a solução aproximada na equação (28) e subtraindo-se o resultado da mesma equação tem-se:

$$l_i \omega'_{i+1/2} = o_i \omega'_{i-1/2} + l_i^{l} \omega'_{i+3/2} + (p'_i - p'_{i+1}) A_{i+1/2} \quad (38)$$

Pode-se concluir, da equação (38) que a correção na velocidade de cada célula depende da correção de suas duas vizinhas. Para que, mais tarde, se possa chegar a uma matriz tridiagonal que solucione a velocidade em função da correção da pressão, esses termos relacionados às células vizinhas serão desprezados. Esse é o motivo da denominação semi-implícita do algoritmo SIMPLE. É importante frisar que a omissão desses termos não influencia na solução final do problema mas apenas no seu processo de convergência. Após a omissão dos termos tem-se:

$$l_i \omega'_{i+1/2} = (p'_i - p'_{i+1}) A_{i+1/2} \quad (39)$$

Que pode ser reescrita como:

$$\omega'_{i+1/2} = m_i^{l} (p'_i - p'_{i+1}) \quad (40)$$

Onde:

$$m_i^l = \frac{A_{i+1/2}}{l_i} \quad (41)$$

A equação (40) pode ser reescrita de modo a mostrar como a velocidade é corrigida pelas correções de pressão:

$$\omega_{i+1/2} = \omega^*_{i+1/2} + m_i^l (p'_i - p'_{i+1}) \quad (42)$$

Analogamente para a célula deslocada para esquerda:

$$\omega_{i-1/2} = \omega^*_{i-1/2} + m_i^o (p'_{i-1} - p'_i) \quad (43)$$

Com:

$$m_i^o = \frac{A_{i-1/2}}{l_{i-1}} \quad (44)$$

Correção de pressão

Como já foi dito anteriormente deve-se acoplar os balanços de massa e momento para que se obtenha uma equação que forneça um perfil de pressões corrigido para uma solução aproximada do perfil de velocidades. Isso pode se obtido através do balanço de massa discretizado (16):

$$\frac{(\rho_i - \rho_i^0) A_i \Delta x_i}{\Delta t} = (\rho \omega A)_{i-1/2} - (\rho \omega A)_{i+1/2} \quad (45)$$

Substituindo-se as velocidades pela expressão de correção da velocidade (42), obtém-se a equação para determinação da correção de pressão:

$$C_i^P p'_i = O_i^P p'_{i-1} + L_i^P p'_{i+1} + k_i^P \quad (46)$$

Com os coeficientes:

$$k_i^P = \frac{(\rho_i^0 - \rho_i)A_i \Delta x_i}{\Delta t} + (\rho\omega^* A)_{i-1/2} - (\rho\omega^* A)_{i+1/2} \quad (47)$$

$$O_i^P = (\rho A)_{i-1/2} m_i^o \quad (48)$$

$$L_i^P = (\rho A)_{i+1/2} m_i^l \quad (49)$$

$$C_i^P = O_i^P + L_i^P \quad (50)$$

Para que o balanço de massa seja satisfeito, o coeficiente k_{ip} deve ser igual a zero. Nesse caso o valor de w* é o valor exato da velocidade w. Portanto o valor de k_{ip} pode ser usado como critério de parada nas iterações. Como o tubo será dividido em N volumes de controle, cria-se uma matriz tridiagonal para a resolução de (47).

Balanço de energia

Como a equação do balanço de energia também tem a mesma forma da equação geral de transporte (1), pode-se substituir a variável genérica ϕ pela entalpia específica na equação (23). O balanço de energia é calculado na célula convencional e a equação obtida é:

$$C_i^h h_i = O_i^h h_{i-1} + L_i^h h_{i+1} + k_i^h \quad (51)$$

Com os coeficientes:

$$C_i^0 = \frac{\rho_i^0 A_i \Delta x_i}{\Delta t} \quad (52)$$

$$k_i^h = S_{Ci}^h A_i \Delta x_i + C_i^0 h_i^0 \quad (53)$$

$$O_i = [[(\rho\omega)_{i-1/2}, 0]] A_{i-1/2} \quad (54)$$

$$L_i = [[-(\rho\omega)_{i+1/2}, 0]] A_{i+1/2} \quad (55)$$

$$C_i^h = O_i + L_i + C_i^0 - S_{Pi}^h A_i \Delta x_i \quad (56)$$

Os coeficientes Oi e Li correspondem ao transporte de energia por convecção nas superfícies do volume de controle. O transporte de energia difusivo será desprezado.

O calor fornecido à célula Q˙ é parte do termo fonte. Para que se tenha um termo proporcional negativo por questão de convergência do método, as equações são montadas da seguinte forma:

$$S_{Ci}^{h} = [[\dot{Q}_i, 0]] \quad (57)$$

$$S_{Pi}^{h} = -\frac{[[-\dot{Q}_i, 0]]}{h_i} \quad (58)$$

A equação (51) também será resolvida usando-se uma matriz tridiagonal.

5 CONDIÇÕES DE CONTORNO

Para a simulação do escoamento em tubos, onde ocorre troca de calor com as paredes, algumas condições de contorno devem ser determinadas. Uma das condições é a determinação da magnitude da troca de calor com as paredes e sua variação ao longo do tempo e é aplicada através das equações (57) e (58). Além disso, os valores das grandezas de estado e transporte nas extremidades dos tubos e suas variações no tempo também são determinados por condições de contorno.

Será usada no trabalho a mesma definição estabelecida por Ponweiser para representar o começo e o final dos tubos e sua divisão em células de balanço. No início do tubo a corrente será definida positiva para escoamentos no sentido de entrada e no final do tubo ela será definida positiva para o fluido saindo do tubo. Como já foi dito anteriormente, os pontos de cálculo para a pressão, a entalpia e a densidade estão nos centros das células regulares que serão definidos como 1 a N. Os centros das células deslocadas se encontram nas margens das regulares e são definidos como 1 + ½ a N + ½. Neles serão calculadas as velocidades. A densidade será determinada no centro de ambas as células e seu cálculo será visto no próximo capítulo. Na figura abaixo estão representadas as células de balanço regulares e deslocadas nas extremidades dos tubos.

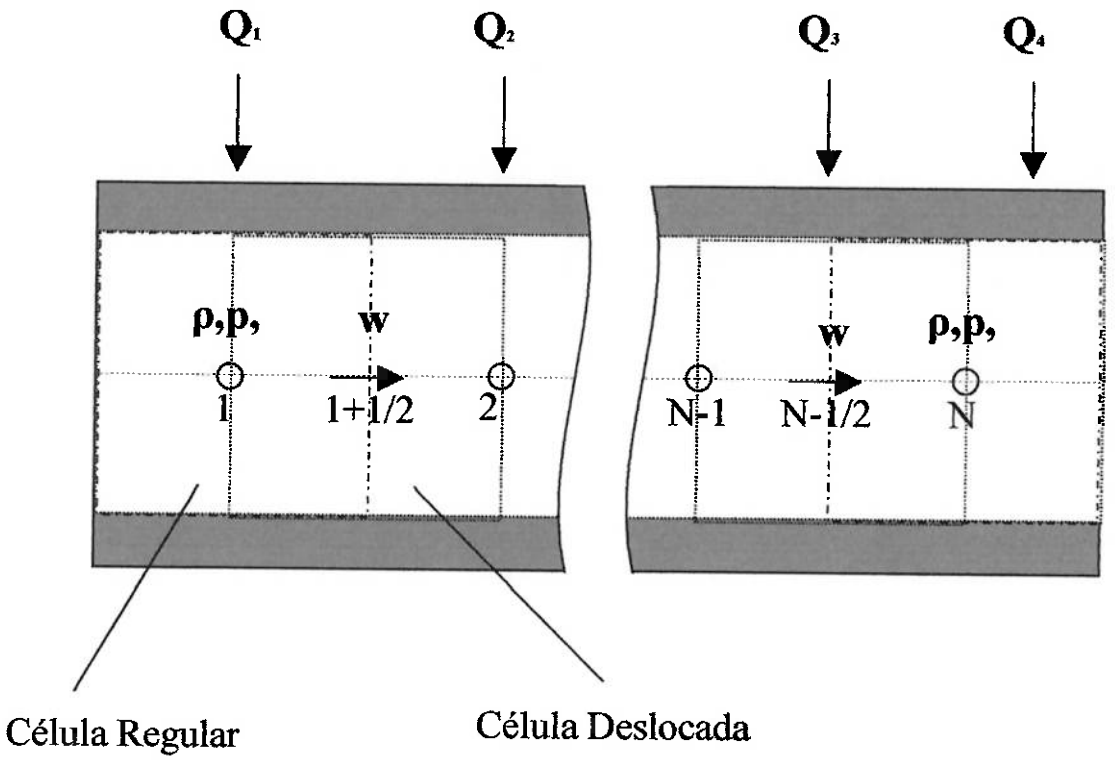

Figura 4. Representação das células nas extremidades dos tubos

Por motivos físicos, nas extremidades dos tubos em que o fluido entra, duas grandezas devem ser fornecidas como condições de contorno (entre pressão, densidade, entalpia específica ou velocidade) e, nas extremidades em que o fluido sai, o fornecimento de uma delas é necessário.

Na simulação de sistemas movidos por gravidade como na circulação natural de água-vapor em caldeiras deve-se fornecer a pressão em ambas as extremidades e a entalpia específica na extremidade de entrada (em função do tempo). Por outro lado, na simulação de geradores de vapor de apenas um passe, onde uma bomba impõe uma dada vazão mássica e que a pressão final é fixa, utiliza-se como condições de contorno fluxo mássico e entalpia específica na entrada e pressão na saída.

Como a velocidade é calculada no centro das células deslocadas (margens das regulares), para um tubo com N células, será formado um sistema com N-1 equações para a determinação das velocidades:

$$\begin{pmatrix} l_1 & -l_1^l & & & & & \\ -o_2 & l_2 & -l_2^l & & & & \\ & -o_3 & l_3 & -l_3^l & & & \\ & & \cdots & \cdots & \cdots & & \\ & & & -o_{N-3} & l_{N-3} & -l_{N-3}^l & \\ & & & & -o_{N-2} & l_{N-2} & -l_{N-2}^l \\ & & & & & -o_{N-1} & l_{N-1} \end{pmatrix} \cdot \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \cdots \\ \omega_{N-3} \\ \omega_{N-2} \\ \omega_{N-1} \end{pmatrix} = \begin{pmatrix} I_1 \\ I_2 \\ I_3 \\ \cdots \\ I_{N-3} \\ I_{N-2} \\ I_{N-1} \end{pmatrix} \tag{59}$$

Onde:

$I_i = k_i^e + (p_i - p_{i+1})A_{i+1/2}$ e

$\omega_i = \omega_{i+1/2}$ da equação (38)

Por outro lado, a pressão é calculada no centro da célula. Desse modo tem-se N equações de correção de pressão resultando-se na seguinte matriz:

$$\begin{pmatrix} C_1^P & -L_1^P & & & & & \\ -O_2^P & C_2^P & -L_2^P & & & & \\ & -O_3^P & C_3^P & -L_3^P & & & \\ & & \cdots & \cdots & \cdots & & \\ & & & -O_{N-2}^P & C_{N-2}^P & -L_{N-2}^P & \\ & & & & -O_{N-1}^P & C_{N-1}^P & -L_{N-1}^P \\ & & & & & -O_N^P & C_N^P \end{pmatrix} \cdot \begin{pmatrix} p'_1 \\ p'_2 \\ p'_3 \\ \cdots \\ p'_{N-2} \\ p'_{N-1} \\ p'_N \end{pmatrix} = \begin{pmatrix} k_1^P \\ k_2^P \\ k_3^P \\ \cdots \\ k_{N-2}^P \\ k_{N-1}^P \\ k_N^P \end{pmatrix} \tag{60}$$

Do mesmo modo, a entalpia também é calculada no centro das células gerando também N equações para o balanço de energia:

$$\begin{pmatrix} C_1^h & -L_1^h & & & & & \\ -O_2^h & C_2^h & -L_2^h & & & & \\ & -O_3^h & C_3^h & -L_3^h & & & \\ & & \cdots & \cdots & \cdots & & \\ & & & -O_{N-2}^h & C_{N-2}^h & -L_{N-2}^h & \\ & & & & -O_{N-1}^h & C_{N-1}^h & -L_{N-1}^h \\ & & & & & -O_N^h & C_N^h \end{pmatrix} \cdot \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ \cdots \\ h_{N-2} \\ h_{N-1} \\ h_N \end{pmatrix} = \begin{pmatrix} k_1^e \\ k_2^e \\ k_3^e \\ \cdots \\ k_{N-2}^e \\ k_{N-1}^e \\ k_N^e \end{pmatrix} \quad (61)$$

Condições de contorno: Pressão e entalpia na entrada do tubo e pressão na saída.

Como o sentido do escoamento depende da relação das pressões locais, ele é resultado dos cálculos. Poderia acontecer, para uma eventual igualdade de pressões das extremidades e devido a uma diminuição da densidade do fluido resultante de um aquecimento, a saída por ambos os lados. Nesse caso não seria necessário o fornecimento da entalpia como condição de contorno. Por outro lado, se na mesma situação houvesse um resfriamento ao invés do aquecimento e a conseqüente entrada de fluido por ambas as extremidades, deveríamos fornecer as duas entalpias.

Para a primeira célula no começo do tubo, os coeficientes para o balanço de momento serão:

$$l_1^0 = \frac{(\rho^0 A\, \Delta x)_{1+1/2}}{\Delta t} \quad (62)$$

$$k_1^e = S_{C1}^p A_{1+1/2} \Delta x_{1+1/2} + l_1^0 \omega_{1+1/2}^0 \quad (63)$$

$$l_1^l = [[-(\rho\omega)_{1+3/2}, 0]]A_{1+3/2} \quad (64)$$

$$l_1 = l_1^l + l_1^0 - (S_P^p A\Delta x)_{1+1/2} \quad (65)$$

$$I_1 = k_1^e + (p_1 - p_2)A_{1+1/2} \quad (66)$$

A corrente de impulso da célula a leste é governada pelo coeficiente l_1^l e será diferente de zero apenas para sentido de escoamento negativo. A falta do coeficiente o_1 satisfaz a hipótese de que as velocidades nas margens esquerda e direita da primeira célula são iguais.

Analogamente, para a célula na extremidade final do tubo os coeficientes são:

$$l_{N-1}^0 = \frac{(\rho^0 A\ \Delta x)_{N-1/2}}{\Delta t} \quad (67)$$

$$k_{N-1}^e = S_{CN-1}^p A_{N-1/2}\Delta x_{N-1/2} + l_{N-1}^0 \omega_{N-1/2}^0 \quad (68)$$

$$o_{N-1} = [[(\rho\omega)_{N-3/2}, 0]]A_{N-3/2} \quad (69)$$

$$l_{N-1} = o_{N-1} + l_{N-1}^0 - (S_P^p A\Delta x)_{N-1/2} \quad (70)$$

$$I_{N-1} = k_{N-1}^e + (p_{N-1} - p_N)A_{N-1/2} \quad (71)$$

Nesse caso a corrente da célula vizinha a oeste será controlada pelo coeficiente o_{N-1} e será diferente de zero para escoamentos em sentido positivo. A falta do coeficiente l_{N-1}^l satisfaz a hipótese de que as velocidades nas margens direita e esquerda da célula N são iguais.

Como pelas condições de contorno as pressões de entrada e saída são dadas, os coeficientes de correção de pressão na primeira e última célula devem ser nulos. Isso é obtido com:

$$k_1^P = 0 \quad (72)$$

$$L_1^P = 0 \quad (73)$$

$$C_1^P = 1 \quad (74)$$

$$k_N^P = 0 \quad (75)$$

$$O_N^P = 0 \quad (76)$$

$$C_N^P = 1 \quad (77)$$

Como os coeficientes da equação de matrizes do balanço de momento são dependentes do sentido do escoamento nas extremidades dos tubos, para velocidades positivas no início do tubo a entalpia deve ser fornecida da seguinte maneira:

$$k_1^h = h_{COMEÇO\_DO\_TUBO} \quad (78)$$

$$L_1 = 0 \quad (79)$$

$$C_1^h = 1 \quad (80)$$

Caso a velocidade no começo do tubo seja negativa, a entalpia será obtida nos cálculos e os coeficientes da primeira célula de balanço serão:

$$C_1^0 = \frac{(\rho^0 A \Delta x)_1}{\Delta t} \quad (81)$$

$$k_i^h = (S_C^h A\, \Delta x)_1 + (C^0 h^0)_1 \quad (82)$$

$$L_i = [[-(\rho\omega)_{1+1/2}, 0]] A_{1+1/2} \quad (83)$$

$$C_1^h = L_1 + C_1^0 - (S_P^h A\, \Delta x)_1 \quad (84)$$

O coeficiente O_1 seria zero de acordo com a condição UPWIND para w < 0, portanto sua ausência não traz nenhum efeito.

Para o final de tubo, a situação é oposta. Para sentido de escoamento positivo a entalpia na última célula é obtida através de:

$$C_N^0 = \frac{(\rho^0 A\, \Delta x)_N}{\Delta t} \quad (85)$$

$$k_N^h = (S_C^h A\Delta x)_N + (C^0 h^0)_N \quad (86)$$

$$O_N = [[(\rho\omega)_{N-1/2}, 0]]A_{N-1/2} \quad (87)$$

$$C_N^h = O_N + C_N^0 - (S_P^h A\Delta x)_N \quad (88)$$

Análogo ao que foi dito antes a falta do coeficiente L_N não traz nenhum efeito. Para sentido negativo de escoamento, deve-se fornecer a entalpia específica na última célula:

$$k_N^h = h_{FINAL\_DO\_TUBO} \quad (89)$$

$$O_N = 0 \quad (90)$$

$$C_N^h = 1 \quad (91)$$

Condições de contorno:Vazão mássica e entalpia específica na entrada e pressão na saída.

Para que se forneça a vazão mássica no começo do tubo, os coeficientes do balanço de energia devem ser:

$$l_1^l = 0 \quad (92)$$

$$l_1 = 1 \quad (93)$$

$$I_1 = \frac{\dot{m}_{COMEÇO\_DO\_TUBO}}{A_1 \rho_1^*} \quad (94)$$

Essa substituição de coeficientes é válida tanto para uma dada vazão positiva quanto para uma dada vazão negativa.

Para a última célula de balanço de momento (N-1) os coeficientes são determinados pelas equações 67, 68, 69, 70, 71 tanto para escoamentos positivos quanto negativos.

Para a correção de pressão da primeira célula, será admitido que uma célula fictícia 0 tem o mesmo valor da correção da primeira (p'0 = p'1) e que as vazões mássicas sobre as duas margens desta célula são iguais. Com isso obtém-se:

$$k_1^P = \frac{(\rho_1^0 - \rho_1)A_1\Delta x_1}{\Delta t} \quad (95)$$

$$L_1^P = (\rho A)_{1+1/2} m_1^l \quad (96)$$

$$C_1^P = L_1^P \quad (97)$$

Para uma dada pressão no fim do tubo, deve-se escolher os coeficientes da última célula regular de balanço de modo que o valor da correção de pressão nessa célula resulte em zero. Isso é obtido por:

$$k_N^P = 0 \quad (98)$$

$$O_N^P = 0 \quad (99)$$

$$C_N^P = 1 \quad (100)$$

Quando o fluxo mássico for positivo no início do tubo, deve-se fornecer a entalpia específica na primeira célula como uma função do tempo. Os coeficientes para o balanço de energia devem ser dados pelas equações 78, 79 e 80.

Caso o fluxo seja negativo no início do tubo, a entalpia na primeira célula será obtida através de cálculos e serão usados os coeficientes 81, 82, 83 e 84.

No fim do tubo serão usadas as equações 85, 86, 87 e 88 para fluxos positivos e 89, 90 e 91 para fluxos negativos.

Deve-se atentar que para essa condição de contorno, no início do tubo haverá um fluxo negativo apenas para uma vazão pré-escolhida. No entanto no final do tubo, mesmo havendo-se uma vazão positiva no início do tubo, pode-se obter uma vazão negativa devido a uma diminuição da densidade causada por um resfriamento. A situação contrária também poderia acontecer em um fluxo negativo no início e um eventual aquecimento.

6 SOLUÇÃO NUMÉRICA PARA O SISTEMA DE EQUAÇÕES

O método escolhido para inversão de matrizes foi o Método de Crout que é uma variação da técnica de eliminação de Gauss. A diferença é que a matriz é decomposta em duas matrizes triangulares como pode ser visto a seguir:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & : & b_1 \\ a_{21} & a_{22} & a_{23} & : & b_2 \\ a_{31} & a_{32} & a_{33} & : & b_3 \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \cdot \begin{bmatrix} 1 & T_{12} & T_{13} & : & c_1 \\ 0 & 1 & T_{23} & : & c_2 \\ 0 & 0 & 1 & : & c_3 \end{bmatrix} \quad (101)$$

O método foi escolhido por envolver equações extremamente simples. Isso pode ser observado quando se multiplica [L] por [T:c] e iguala-se a [a:b]:

$$a_{11} = L_{11} \quad (102)$$

$$a_{21} = L_{21} \quad (103)$$

$$a_{31} = L_{31} \quad (104)$$

$$a_{12} = L_{11}T_{12} \quad (105)$$

$$a_{22} = L_{21}T_{12} + L_{22} \quad (106)$$

$$a_{32} = L_{31}T_{12} + L_{32} \quad (107)$$

$$a_{13} = L_{11}T_{13} \quad (108)$$

$$a_{23} = L_{21}T_{13} + L_{22}T_{23} \quad (109)$$

$$a_{33} = L_{31}T_{13} + L_{32}T_{23} + L_{33} \quad (110)$$

$$b_1 = L_{11}c_1 \quad (111)$$

$$b_2 = L_{21}c_1 + L_{22}c_2 \quad (112)$$

$$b_3 = L_{31}c_1 + L_{32}c_2 + L_{33}c_3 \quad (113)$$

As equações podem então ser reescritas como:

$$L_{11} = a_{11} \quad (114)$$

$$L_{21} = a_{21} \quad (115)$$

$$L_{31} = a_{31} \quad (116)$$

$$T_{12} = a_{12} / L_{11} \quad (117)$$

$$L_{22} = a_{22} - L_{21}T_{12} \quad (118)$$

$$L_{32} = a_{32} - L_{31}T_{12} \quad (119)$$

$$T_{13} = a_{13} / L_{11} \quad (120)$$

$$T_{23} = (a_{23} - L_{21}T_{13}) / L_{22} \quad (121)$$

$$L_{33} = a_{33} - L_{31}T_{13} - L_{32}T_{23} \quad (122)$$

$$c_1 = b_1 / L_{11} \quad (123)$$

$$c_2 = (b_2 - L_{21}c_1) / L_{22} \quad (124)$$

$$c_3 = (b_3 - L_{31}c_1 - L_{32}c_2) / L_{33} \quad (125)$$

Por fim, obtém-se as incógnitas:

$x_3 = c_3$ (126)

$x_2 = c_2 - T_{23}x_3$ (127)

$x_1 = c_1 - T_{13}x_3 - T_{12}x_2$ (128)

O método de Crout para uma matriz n x n é obtido pelas equações abaixo:

$$L_{ij} = a_{ij} - \sum_{k=1}^{j-1} L_{ik}T_{kj} \qquad \text{para i} \geq \text{j, i} = 1, \ldots,\text{n (129)}$$

$$T_{ij} = \frac{1}{L_{ii}}\left(a_{ij} - \sum_{k=1}^{j-1} L_{ik}T_{kj}\right) \qquad \text{para i} < \text{j, j} = 2, \ldots,\text{n (130)}$$

$$c_i = \frac{1}{L_{ii}}\left(b_i - \sum_{k=1}^{j-1} L_{ik}c_k\right) \qquad \text{para i} = 2, 3, \ldots, \text{n (131)}$$

$T_{ij} = a_{ij} / a_{11}$ para i = 1 (132)

$L_{ij} = a_{i1}$ para j = 1 (133)

$c_1 = b_1 / L_{11}$ (134)

Pode-se então obter as incógnitas:

$x_n = c_n$ (135)

$$x_j = c_j - \sum_{r=j+1}^{n} T_{jr}x_r \qquad \text{para j} = 1, \ldots, \text{n-1 (136)}$$

Outra grande vantagem do método é que ele pode ser bastante simplificado para matrizes tridiagonais. Caso os coletores sejam calculados separadamente e o sistema de

equações seja resolvido apenas para os tubos, cada célula de controle terá a influência apenas de dois vizinhos. Nesse caso, o sistema formará uma matriz tridiagonal que pode ser resolvida pelo Método de Crout. O equacionamento pode ser visto abaixo:

$$\begin{bmatrix} d_1 & t_1 & 0 & 0 \\ l_2 & d_2 & t_2 & 0 \\ 0 & l_3 & d_3 & t_3 \\ 0 & 0 & l_4 & d_4 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{Bmatrix} \quad (137)$$

$$\begin{bmatrix} d_1 & t_1 & 0 & 0 & : & b_1 \\ l_2 & d_2 & t_2 & 0 & : & b_2 \\ 0 & l_3 & d_3 & t_3 & : & b_3 \\ 0 & 0 & l_4 & d_4 & : & b_4 \end{bmatrix} = \begin{bmatrix} D_1 & 0 & 0 & 0 \\ L_2 & D_2 & 0 & 0 \\ 0 & L_3 & D_3 & 0 \\ 0 & 0 & L_4 & D_4 \end{bmatrix} \cdot \begin{bmatrix} 1 & T_1 & 0 & 0 & : & B_1 \\ 0 & 1 & T_2 & 0 & : & B_2 \\ 0 & 0 & 1 & T_3 & : & B_3 \\ 0 & 0 & 0 & 1 & : & B_4 \end{bmatrix} \quad (138)$$

Após a multiplicação obtém-se:

$d_1 = D_1$ (139)

$l_2 = L_2$ (140)

$t_1 = T_1 D_1$ (141)

$d_2 = T_1 L_2 + D_2$ (142)

$l_3 = L_3$ (143)

$t_2 = T_2 D_2$ (144)

$d_3 = T_2 L_3 + D_3$ (145)

$l_4 = L_4$ (146)

$$t_3 = T_3 D_3 \quad (147)$$

$$d_4 = T_3 L_4 + D_4 \quad (148)$$

$$b_1 = B_1 D_1 \quad (149)$$

$$b_2 = B_1 L_2 + B_2 D_2 \quad (150)$$

$$b_3 = B_2 L_3 + B_3 D_3 \quad (151)$$

$$b_4 = B_3 L_4 + B_4 D_4 \quad (152)$$

As equações podem ser reescritas como:

$$D_1 = d_1 \quad (153)$$

$$L_2 = l_2 \quad (154)$$

$$T_1 = \frac{t_1}{D_1} \quad (155)$$

$$D_2 = d_2 - T_1 L_2 \quad (156)$$

$$L_3 = l_3 \quad (157)$$

$$T_2 = \frac{t_2}{D_2} \quad (158)$$

$$D_3 = d_3 - T_2 L_3 \quad (159)$$

$$L_4 = l_4 \quad (160)$$

$$T_3 = \frac{t_3}{D_3} \quad (161)$$

$$D_4 = d_4 - T_3 L_4 \quad (162)$$

$$B_1 = \frac{b_1}{D_1} \quad (163)$$

$$B_2 = \frac{(b_2 - B_1 L_2)}{D_2} \quad (164)$$

$$B_3 = \frac{(b_3 - B_2 L_3)}{D_3} \quad (165)$$

$$B_4 = \frac{(b_4 - B_3 L_4)}{D_4} \quad (166)$$

Portanto, para uma matriz tridiagonal nxn:

$$D_1 = d_1 \quad (167)$$

$$T_i = \frac{t_i}{D_i}, \text{ para i} = 1,2,...,n-1 \quad (168)$$

$$D_i = d_i - T_{i-1} L_i, \text{ para i} = 2,...,n \quad (169)$$

$$B_i = \frac{b_i}{D_i} \quad (170)$$

$$B_i = \frac{(b_i - B_{i-1} L_i)}{D_i}, \text{ para i} = 2,...,n \quad (171)$$

Obtém-se as soluções:

$$x_n = B_i \quad (172)$$

$$x_i = B_i - T_i x_{i+1}, \text{ para i} = n-1,n-2,...,1 \quad (173)$$

7 PROPRIEDADES DA ÁGUA

Conforme já foi verificado, para a execução do método SIMPLE, é necessário que se calcule a densidade da água tendo-se como entrada a pressão e a entalpia específica. A solução utilizada no trabalho é o uso das equações da IAPWS-IF97. Para o desenvolvimento dessas equações, a superfície termodinâmica da água foi dividida em 5 regiões:

Região 1 para toda a fase líquida;

Região 2 para o vapor e para gás ideal;

Região 3 para a área próxima ao ponto crítico;

Região 4 para a curva de saturação;

Região 5 para temperaturas entre 1073,1 K e 2273,1 K e pressões abaixo de 10 MPa

Para a realização da simulação, foram modeladas as regiões de 1 até 4, que envolvem pressões de 0 MPa a 100MPa e temperaturas de 273,15 K até 1073,1 K.

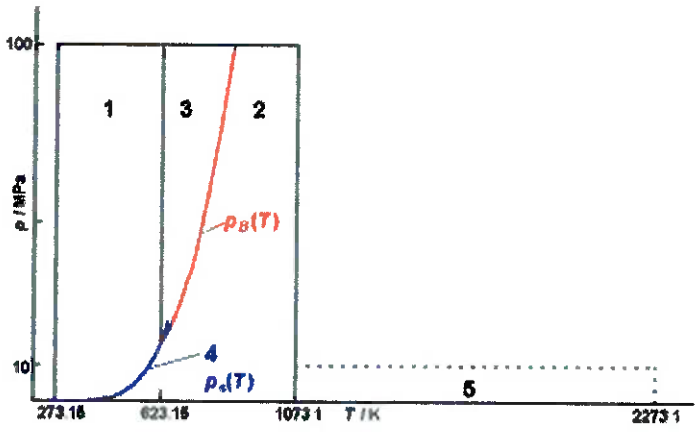

Figura 5. Superfície termodinâmica da água.

Para as regiões 1, 2 e 4, as equações fornecidas pela IAPWS para determinação da entalpia e volume específico têm como entrada a pressão e a temperatura e para a

região 3, são fornecidas equações para determinação da pressão e da entalpia em função da densidade e a temperatura. Desse modo, foi necessária a utilização do método iterativo de Newton-Raphson já que as entradas disponíveis eram outras. Para as regiões 1, 2 e 4, onde apenas uma das entradas era desconhecida (temperatura), foi usada a equação:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad (174)$$

Para a região 3, onde ambas as entradas eram desconhecidas, foi usado:

$$\begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}_{k+1} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}_k - [J]_k^{-1} \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix}_k \quad (175)$$

onde:

$$[J] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \quad (176)$$

e

$$[J]^{-1} = \frac{1}{\det[J]} \left(cof[J]^t \right) \quad (177)$$

O primeiro passo do programa na determinação das propriedades da água é definir a região em que o ponto de operação se encontra. Para isso, calcula-se todas a entalpias especificas que façam margem a duas possíveis regiões para a dada pressão. Para pressões menores que 16,529 MPa as margens estão entre 1 e 4 (líquido saturado) e entre 4 e 2 (vapor saturado). Para pressões entre 16,529 MPa e a pressão crítica as margens estão entre as regiões 1 e 3, entre 3 e 4 (tanto para líquido saturado quanto para vapor saturado) e entre 3 e 2. Como na região 3 a determinação das entalpias específicas em função da pressão e temperatura é feita de maneira iterativa pelo método de Newton-Raphson, obtém-se a entalpia da margem inferior entre 3 e 4 (líquido saturado) usando-se como chute inicial uma densidade maior que a marginal. Por outro lado, para se

determinar a entalpia na margem superior (vapor saturado), usa-se uma densidade menor que a marginal como chute inicial. Para pressões acima da crítica, as margens estão entre 1 e 3 e entre 3 e 2.

Abaixo serão apresentadas todas as equações e coeficientes para determinação de volume especifico e entalpia nas regiões 1 e 2, para determinação de temperatura e pressão da margem entre as regiões 3 e 2, para determinação de entalpia e pressão na região 3 e para a determinação de temperatura de saturação e pressão de saturação (região 4).

Cálculo da entalpia específica e volume específico na região 1:

$$v(T,p)\frac{p}{RT} = \pi\gamma_\pi \quad (178)$$

e

$$\frac{h(T,p)}{RT} = \tau\gamma_\tau \quad (179)$$

Onde:

$$\gamma_\pi = \left(\frac{\partial\gamma}{\partial\pi}\right)_\tau = \sum_{i=1}^{34} -n_i I_i (7{,}1-\pi)^{(I_i-1)}(\tau-1{,}222)^{J_i} \quad (180)$$

$$\gamma_\tau = \left(\frac{\partial\gamma}{\partial\tau}\right)_\pi = \sum_{i=1}^{34} n_i (7{,}1-\pi)^{I_i} J_i (\tau-1{,}222)^{(J_i-1)} \quad (181)$$

$$\tau = \frac{1386K}{T} \quad (182)$$

$$\pi = \frac{p}{16{,}53MPa} \quad (183)$$

Tabela 1. Coeficientes da região1.

i	Ii	Ji	n
1	0	-2	0.14632971213167
2	0	-1	-0.84548187169114
3	0	0	-3.756360367204
4	0	1	3.3855169168385
5	0	2	-0.95791963387872
6	0	3	0.15772038513228
7	0	4	-0.016616417199501
8	0	5	8.1214629983568E-04
9	1	-9	2.8319080123804E-04
10	1	-7	-6.0706301565874E-04
11	1	-1	-0.018990068218419
12	1	0	-0.032529748770505
13	1	1	-0.021841717175414
14	1	3	-5.283835796993E-05
15	2	-3	-4.7184321073267E-04
16	2	0	-3.0001780793026E-04
17	2	1	4.7661393906987E-05
18	2	3	-4.4141845330846E-06
19	2	17	-7.2694996297594E-16
20	3	-4	-3.1679644845054E-05
21	3	0	-2.8270797985312E-06
22	3	6	-8.5205128120103E-10
23	4	-5	-2.2425281908E-06
24	4	-2	-6.5171222895601E-07
25	4	10	-1.4341729937924E-13
26	5	-8	-4.0516996860117E-07
27	8	-11	-1.2734301741641E-09
28	8	-6	-1.7424871230634E-10
29	21	-29	-6.8762131295531E-19
30	23	-31	1.4478307828521E-20
31	29	-38	2.6335781662795E-23
32	30	-39	-1.1947622640071E-23
33	31	-40	1.8228094581404E-24
34	32	-41	-9.3537087292458E-26

Cálculo da pressão e da temperatura na margem entre as regiões 2 e 3:

$$\pi = n_1 + n_2\theta + n_3\theta^2 \quad (184)$$

Onde:

$$\theta = n_4 + \left(\frac{\pi - n_3}{n_3}\right)^{0,5} \quad (185)$$

$$\pi = \frac{p_B}{1MPa} \quad (186)$$

$$\theta = \frac{T_B}{1K} \quad (187)$$

Tabela 2. Coeficientes da região de margem.

i	ni
1	3,4805185628969E+02
2	-1,1671859879975E+00
3	1,0192970039326E-03
4	5,7254459862746E+02
5	1,3918839778870E+01

Cálculo da entalpia específica e volume específico na região 2:

$$v(T,p)\frac{p}{RT} = \pi\left(\gamma_\pi^0 + \gamma_\pi^r\right) \quad (188)$$

$$\frac{h(T,p)}{RT} = \tau\left(\gamma_\tau^0 + \gamma_\tau^r\right) \quad (189)$$

Onde:

$$\gamma_{\pi}^{0} = \left(\frac{\partial \gamma^{0}}{\partial \pi}\right)_{\tau} = \frac{1}{\pi} \quad (190)$$

$$\gamma_{\tau}^{0} = \left(\frac{\partial \gamma^{0}}{\partial \tau}\right)_{\pi} = \sum_{i=1}^{9} n_i^0 J_i^0 \tau^{J_i^0 - 1} \quad (191)$$

$$\gamma_{\pi}^{r} = \left(\frac{\partial \gamma^{r}}{\partial \pi}\right)_{\tau} = \sum_{i=1}^{43} n_i I_i \pi^{J_i - 1} (\tau - 0{,}5)^{J_i} \quad (192)$$

$$\gamma_{\tau}^{r} = \left(\frac{\partial \gamma^{r}}{\partial \tau}\right)_{\pi} = \sum_{i=1}^{43} n_i \pi^{I_i} J_i (\tau - 0{,}5)^{J_i - 1} \quad (193)$$

$$\tau = \frac{540K}{T} \quad (194)$$

$$\pi = \frac{p}{1MPa} \quad (195)$$

Tabela 3. Coeficientes de gás ideal da região 2.

i	Ji0	ni0
1	0	-9.6927686500217
2	1	10.086655968018
3	-5	-0.005608791128302
4	-4	0.071452738081455
5	-3	-0.40710498223928
6	-2	1.4240819171444
7	-1	-4.383951131945
8	2	-0.28408632460772
9	3	0.021268463753307

Tabela 4. Coeficientes da parte residual da região 2.

i	Ii	Ji	ni
1	1	0	-1.7731742473213E-03
2	1	1	-0.017834862292358

3	1	2	-0.045996013696365
4	1	3	-0.057581259083432
5	1	6	-0.05032527872793
6	2	1	-3.3032641670203E-05
7	2	2	-1.8948987516315E-04
8	2	4	-3.9392777243355E-03
9	2	7	-0.043797295650573
10	2	36	-2.6674547914087E-05
11	3	0	2.0481737692309E-08
12	3	1	4.3870667284435E-07
13	3	3	-3.227767723857E-05
14	3	6	-1.5033924542148E-03
15	3	35	-0.040668253562649
16	4	1	-7.8847309559367E-10
17	4	2	1.2790717852285E-08
18	4	3	4.8225372718507E-07
19	5	7	2.2922076337661E-06
20	6	3	-1.6714766451061E-11
21	6	16	-2.1171472321355E-03
22	6	35	-23.895741934104
23	7	0	-5.905956432427E-18
24	7	11	-1.2621808899101E-06
25	7	25	-0.038946842435739
26	8	8	1.1256211360459E-11
27	8	36	-8.2311340897998
28	9	13	1.9809712802088E-08
29	10	4	1.0406965210174E-19
30	10	10	-1.0234747095929E-13
31	10	14	-1.0018179379511E-09
32	16	29	-8.0882908646985E-11
33	16	50	0.10693031879409
34	18	57	-0.33662250574171
35	20	20	8.9185845355421E-25
36	20	35	3.0629316876232E-13
37	20	48	-4.2002467698208E-06
38	21	21	-5.9056029685639E-26
39	22	53	3.7826947613457E-06
40	23	39	-1.2768608934681E-15
41	24	26	7.3087610595061E-29
42	24	40	5.5414715350778E-17
43	24	58	-9.436970724121E-07

Cálculo da pressão e da entalpia específica na região 3

$$\frac{p(T,\rho)}{\rho RT} = \delta\phi_\delta \quad (196)$$

$$\frac{h(T,\rho)}{RT} = \tau\phi_r + \delta\phi_\delta \quad (197)$$

$$\phi_\delta = \left(\frac{\partial\phi}{\partial\delta}\right)_\tau = \frac{n_1}{\delta} + \sum_{i=2}^{40} n_i I_i \delta^{I_i - 1}\tau^{J_i} \quad (198)$$

$$\phi_\tau = \left(\frac{\partial\phi}{\partial\tau}\right)_\delta = \sum_{i=2}^{40} n_i \delta^{I_i} J_i \tau^{J_i - 1} \quad (199)$$

$$\tau = \frac{T_C}{T} = \frac{647{,}096K}{T} \quad (200)$$

$$\delta = \frac{\rho}{\rho_C} = \frac{\rho}{32kg/m^3} \quad (201)$$

Tabela 5. Coeficientes da região 3.

i	Ii	Ji	ni
1	0	0	1.0658070028513
2	0	0	-15.732845290239
3	0	1	20.944396974307
4	0	2	-7.6867707878716
5	0	7	2.6185947787954
6	0	10	-2.808078114862
7	0	12	1.2053369696517
8	0	23	-8.4566812812502E-03
9	1	2	-1.2654315477714
10	1	6	-1.1524407806681
11	1	15	0.88521043984318
12	1	17	-0.64207765181607
13	2	0	0.38493460186671
14	2	2	-0.85214708824206
15	2	6	4.8972281541877

16	2	7	-3.0502617256965
17	2	22	0.039420536879154
18	2	26	0.12558408424308
19	3	0	-0.2799932969871
20	3	2	1.389979956946
21	3	4	-2.018991502357
22	3	16	-8.2147637173963E-03
23	3	26	-0.47596035734923
24	4	0	0.0439840744735
25	4	2	-0.44476435428739
26	4	4	0.90572070719733
27	4	26	0.70522450087967
28	5	1	0.10770512626332
29	5	3	-0.32913623258954
30	5	26	-0.50871062041158
31	6	0	-0.022175400873096
32	6	2	0.094260751665092
33	6	26	0.16436278447961
34	7	2	-0.013503372241348
35	8	26	-0.014834345352472
36	9	2	5.7922953628084E-04
37	9	26	3.2308904703711E-03
38	10	0	8.0964802996215E-05
39	10	1	-1.6557679795037E-04
40	11	26	-4.4923899061815E-05

Cálculo da pressão e da temperatura de saturação (região 4):

$$\frac{p_S(T)}{1MPa} = \left[\frac{2C}{-B + \left(B^2 - 4AC\right)^{0,5}} \right]^4 \quad (202)$$

Onde:

$$A = \vartheta^2 + n_1\vartheta + n_2 \quad (203)$$

$$B = n_3\vartheta^2 + n_4\vartheta + n_5 \quad (204)$$

$$C = n_6\vartheta^2 + n_7\vartheta + n_8 \quad (205)$$

$$\vartheta = \frac{T}{1K} + \frac{n_9}{\frac{T}{1K} - n_{10}} \quad (206)$$

$$\frac{T_S(p)}{1K} = \frac{n_{10} + D - \left[(n_{10} + D)^2 - 4(n_9 + n_{10}D)\right]^{0,5}}{2} \quad (207)$$

Onde:

$$D = \frac{2G}{-F - (F^2 - 4EF)^{0,5}} \quad (208)$$

$$E = \beta^2 + n_3\beta + n_6 \quad (209)$$

$$F = n_1\beta^2 + n_4\beta + n_7 \quad (210)$$

$$G = n_2\beta^2 + n_5\beta + n_8 \quad (211)$$

$$\beta = \left(\frac{p}{1MPa}\right)^{0,25} \quad (212)$$

Tabela 6. Coeficientes da região 4.

i	ni
1	1167.0521452767
2	-724213.16703206
3	-17.073846940092
4	12020.82470247
5	-3232555.0322333
6	14.91510861353
7	-4823.2657361591
8	405113.40542057

9	-0.23855557567849
10	650.17534844798

8 RESULTADOS E DISCUSSÕES

Para que se pudesse estudar o efeito da variação do aquecimento na circulação natural do gerador de vapor, três condições foram simuladas. Para a simulação foi usado um passo de tempo de 0,3 s para um total de 60 s cada condição. O tubo que representa a tubulação de descida da água tem diâmetro externo 63,50 mm e espessura 3,00 mm e os tubos de aquecimento, diâmetro externo 50,80 e espessura 3 mm. Para a simulação será convencionado o sentido positivo do escoamento para baixo (tubulão para tubulão de lama)

Nas três condições, foi mantida a pressão do tubulão a 7 MPa, com água inicialmente saturada. Na condição que será chamada A, foi imposto um aquecimento de 100 kW por trecho no primeiro "riser" e 1000 kW por trecho no segundo "riser". Nas condições B e C foi mantido o mesmo aquecimento no segundo "riser", porém no primeiro ele foi alterado para 500 kW e 20 kW respectivamente.

O resultado da simulação pode ser visto abaixo. As figuras 6, 9, e 12 mostram as curvas da vazão mássica total nos tubos para as três condições. As figuras 7, 10 e 13 mostram as vazões de vapor para os "risers". Por fim, as figuras 8, 11 e 14 trazem os perfis de densidade ao longo dos tubos no instante final da simulação (60s).

Para a condição A, observa-se na figura 6 que o escoamento no primeiro riser, inicialmente ocorre no sentido tubulão-tubulão de lama (para baixo). No entanto esse sentido inverte após aproximadamente 40 s e no final dos 60 s ele ainda não havia estabilizado. Isso pode ser também observado na figura 7, com o surgimento de uma vazão de vapor do primeiro riser para o tubulão nos últimos segundos de simulação (quando há a inversão no sentido de escoamento). Observando-se a figura 8, nota-se que nesse tubo o perfil da densidade não apresenta uma variação linear, e tem uma densidade menor no centro do tubo que em ambas extremidades, confirmando o caráter transitório do escoamento naquele tubo ainda no final da simulação. O perfil de densidades, como era de se esperar, é quase constante no "downcomer" já que não se tem vapor e reduz

linearmente no segundo "riser" a medida que se aproxima do tubulão e que o título da mistura aumenta.

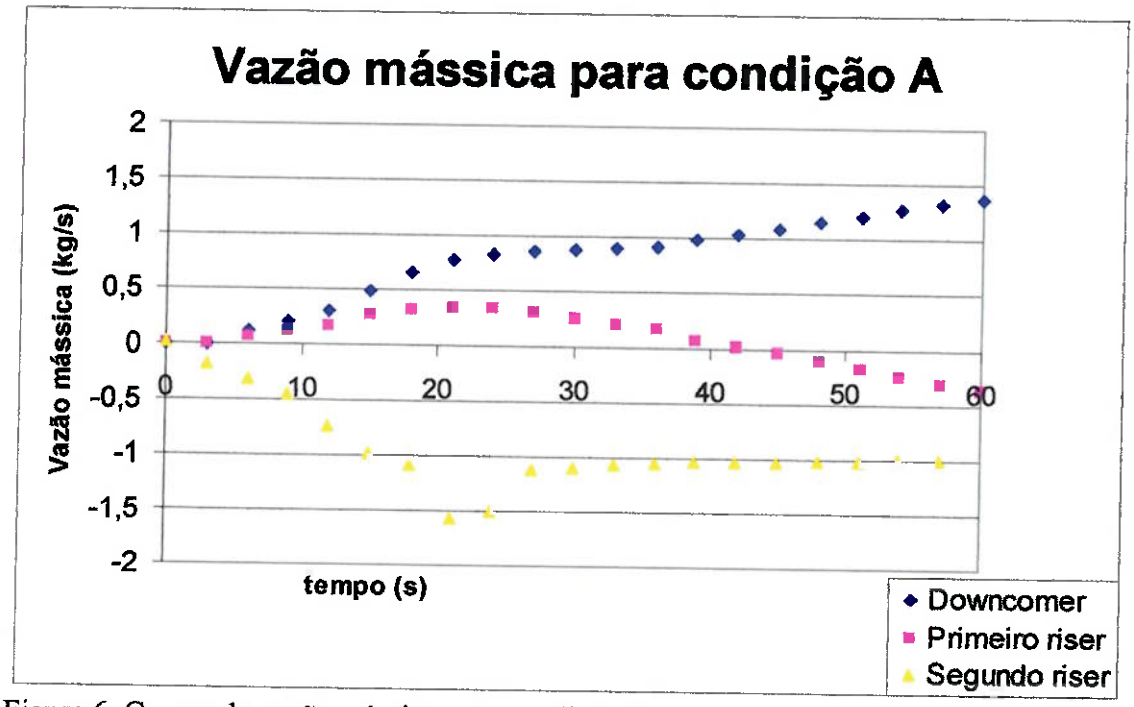

Figura 6. Curvas da vazão mássica para condição A.

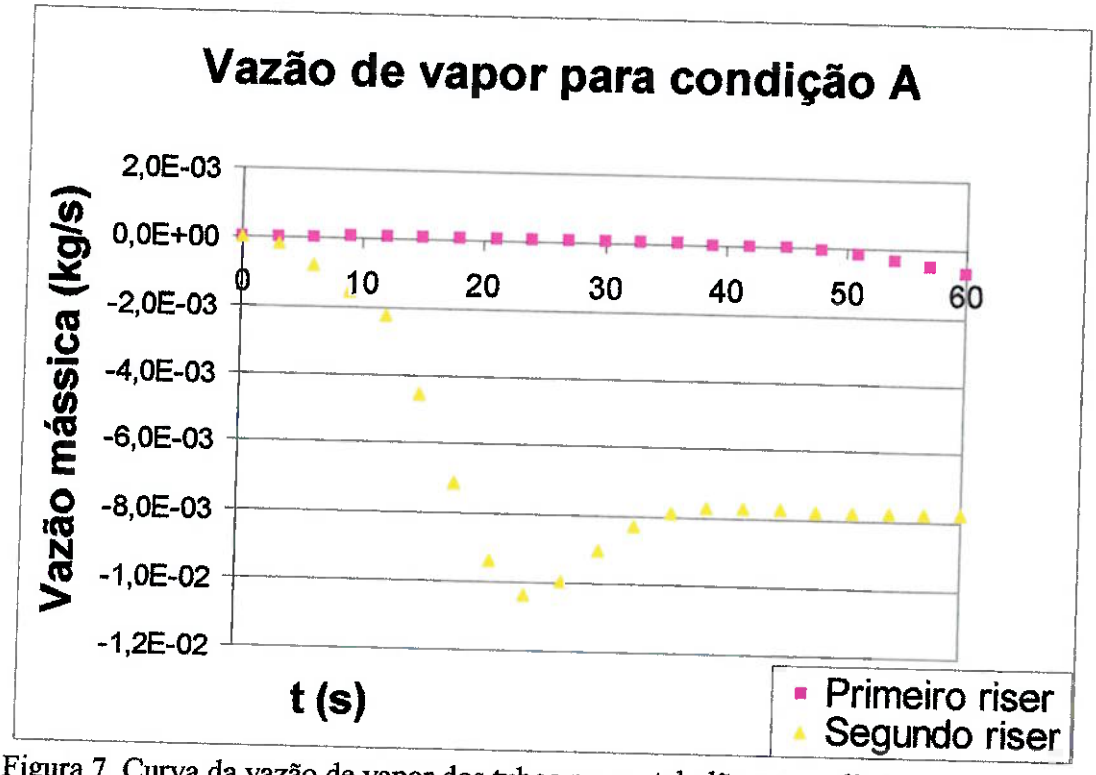

Figura 7. Curva da vazão de vapor dos tubos para o tubulão na condição A.

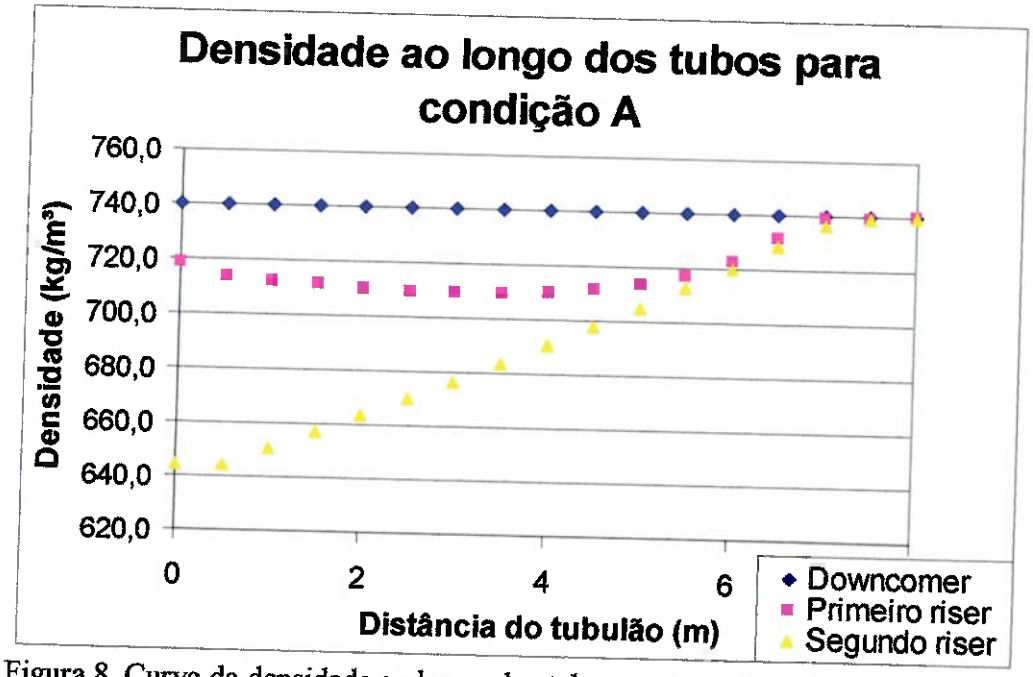

Figura 8. Curva da densidade ao longo dos tubos para condição A

Para a condição B foi imposto um maior aquecimento no primeiro "riser", o que teve como resposta uma vazão no sentido tubulão de lama-tubulão definida mais rapidamente. Com isso todos os escoamentos aparentam estarem estabilizados no instante final da simulação e os perfis das densidades obtidos são lineares.

Observa-se também uma maior vazão no "downcomer" já que nesse caso há uma maior circulação natural devido ao aumento da energia fornecida e ao fato de que esse tubo passou a ser o único a escoar no sentido positivo.

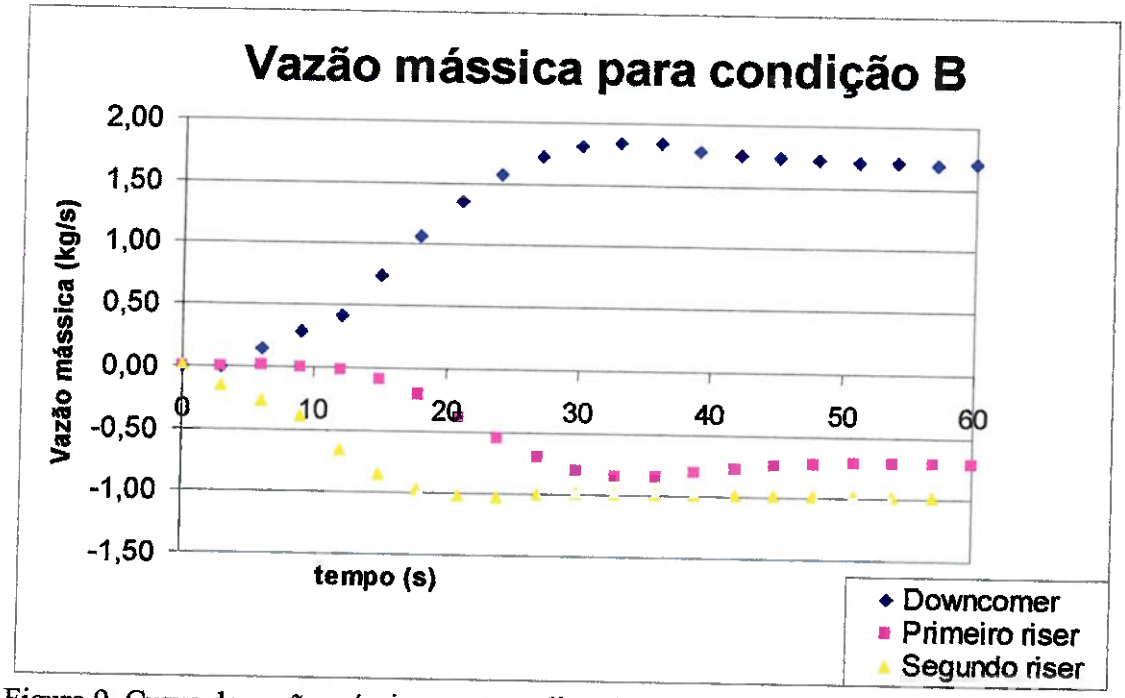

Figura 9. Curva da vazão mássica para condição B.

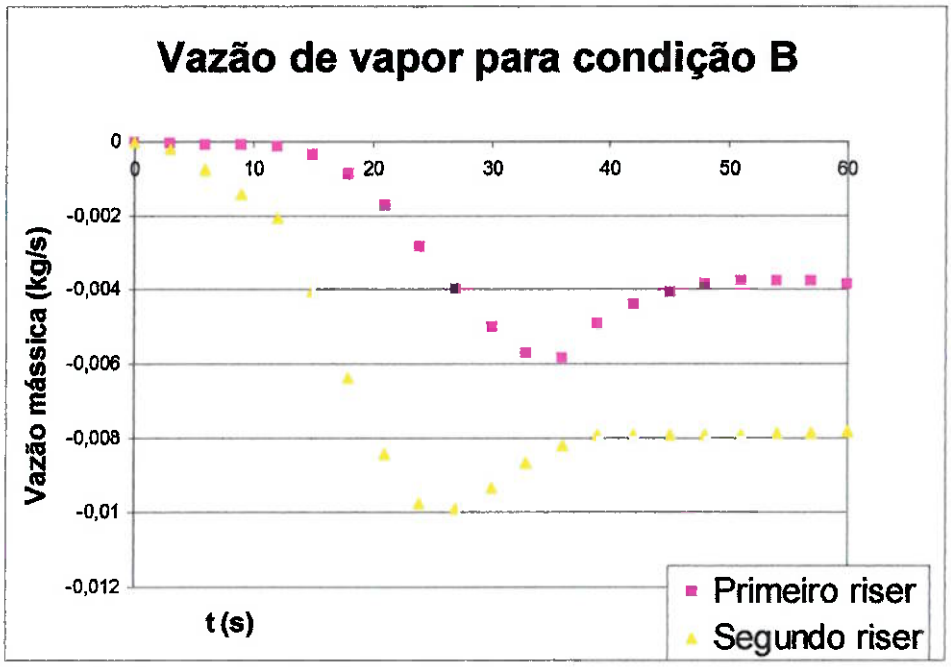

Figura 10. Curva da vazão de vapor dos tubos para o tubulão na condição B.

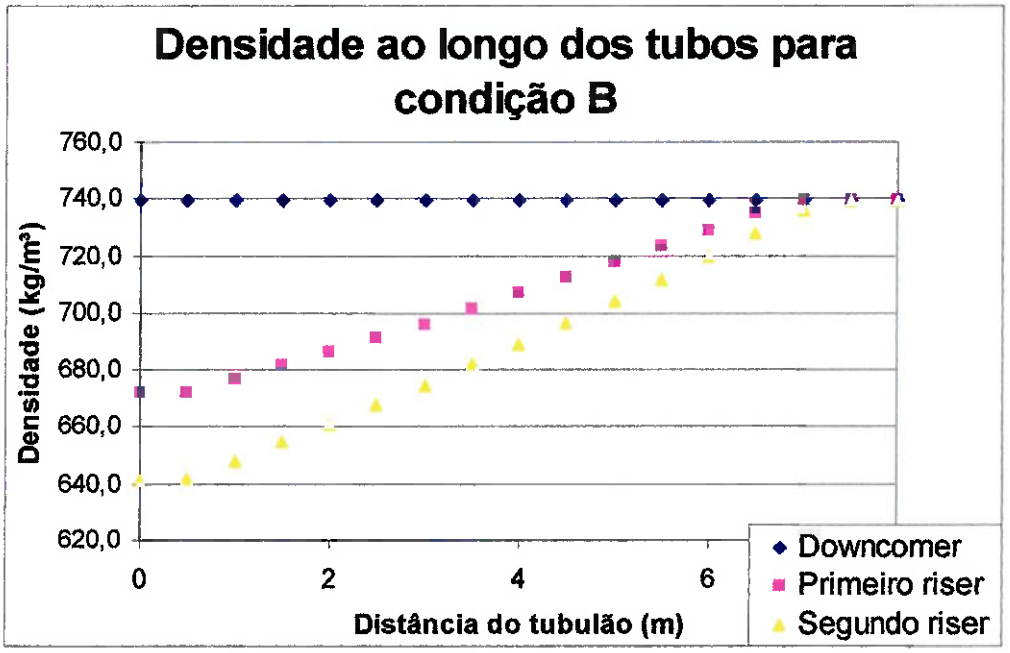

Figura 11. Curva da densidade ao longo dos tubos para condição B.

Na condição C, houve uma diminuição do fornecimento de calor ao primeiro "riser" que gerou, consequentemente, um escoamento positivo bem definido nesse tubo. Observa-se que os escoamentos se estabilizam rapidamente e tem-se como conseqüência um perfil de densidades bem comportado. Observa-se ainda que, como o primeiro riser também está com título muito baixo, ele tem a curva do perfil de densidades muito semelhante à curva do downcomer.

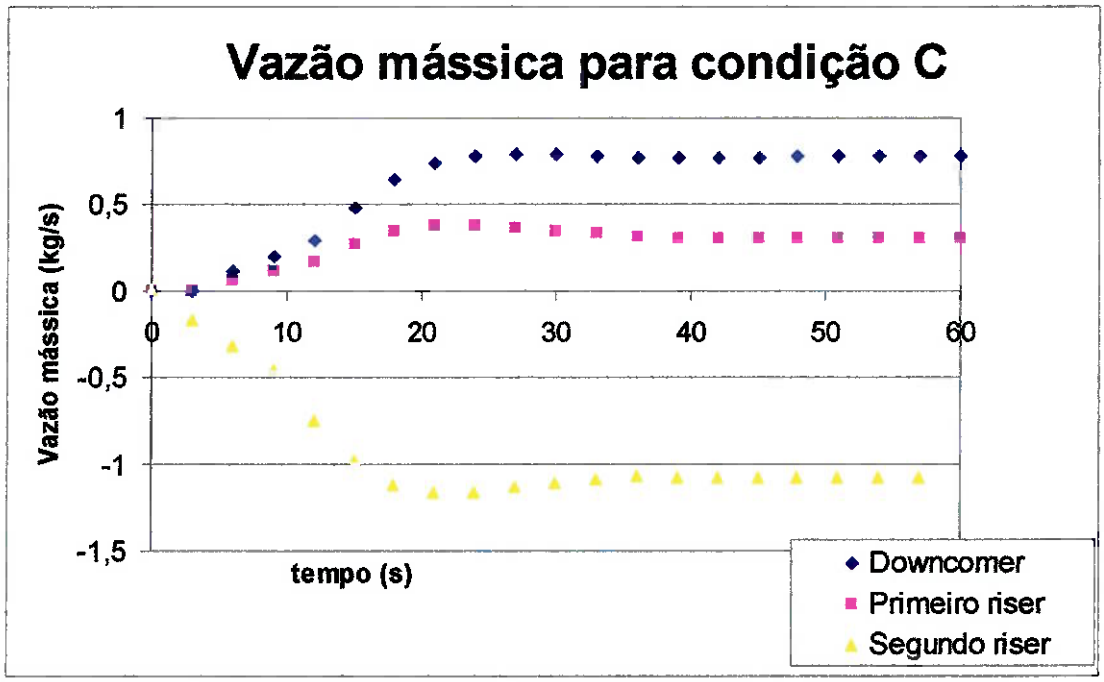

Figura 12. Curva da vazão mássica para a condição C.

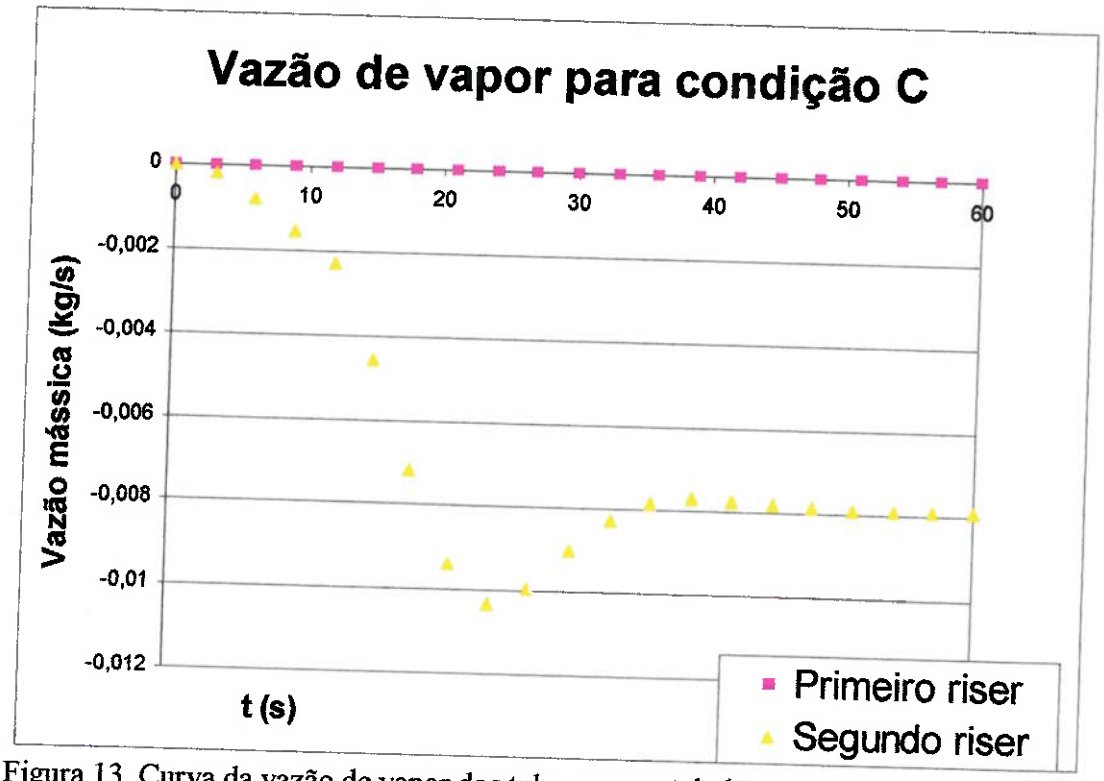

Figura 13. Curva da vazão de vapor dos tubos para o tubulão na condição C.

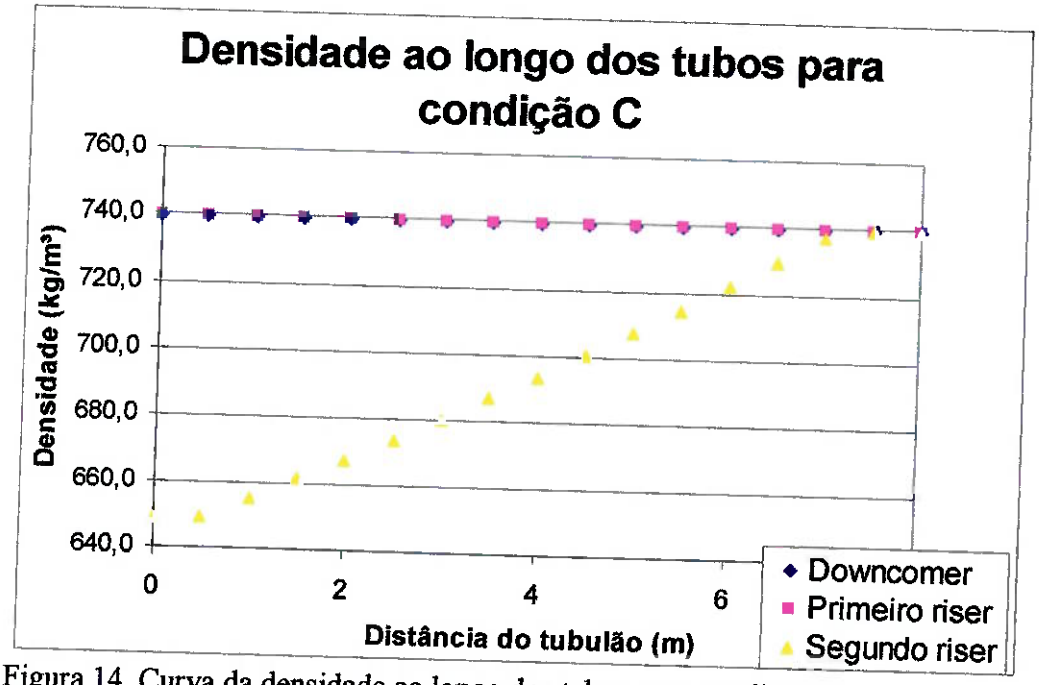

Figura 14. Curva da densidade ao longo dos tubos para condição C.

Foi observado na simulação que a medida que se aumenta o aquecimento dos tubos, obtém-se uma menor pressão no tubulão de lama. Para o aquecimento de 20 kW a pressão obtida foi de 7,02886 MPa. Aumentando-se o aquecimento para 100 kW e para 500 kW essa pressão atingiu 7,0285 MPa e 7,02839 MPa. Considerando-se que a medida que se aumenta o aquecimento, a vazão do downcomer aumenta e que seu perfil de densidades é praticamente o mesmo para todas as condições, acrescentando-se o fato de que a pressão do tubulão é fixa, devemos ter uma queda da pressão com o aumento do aquecimento resultante do aumento da vazão (maior perda de carga). Desse modo a relação obtida na simulação se mostra coerente.

9 CONCLUSÃO

Para o desenvolvimento do projeto, foi feita uma análise de viabilidade onde se discutiu a importância de se simular numericamente o gerador de vapor para seu projeto e controle. Foram enumeradas varias vantagens da simulação numérica em relação à experimental, dentre as quais podemos citar o menor custo e a maior velocidade. Foi traçado um plano para o desenvolvimento do projeto, determinando atividades e impondo um cronograma. O método numérico a ser usado foi escolhido e a modelagem do sistema foi desenvolvida assim como a maneira de se impor as condições de contorno.

Foi escolhida a solução de se modelar o sistema como sendo um conjunto de N pequenos volumes de controle (volume finito) por ser bastante usado em simulação de escoamento e transferência de calor e por estar disponível em uma vasta bibliografia.

Quanto ao processo de resolução foi escolhido o algoritmo iterativo SIMPLE. Ele resolve o problema de não se conhecer os campos de pressão e temperatura através da utilização de um valor inicial aleatório de pressão que converge para a solução.

Na modelagem realizada as células recebem de suas vizinhas as informações necessárias aos balanços (os valores das grandezas de estado e transporte).

Foi desenvolvido todo o equacionamento para as condições de contorno tanto para o caso de escoamento natural quanto para escoamento forçado em caldeira de um passe. No primeiro caso estabeleceu-se as condições a serem fornecidas como sendo pressão na entrada e saída e entalpia na entrada. Para o segundo caso as condições são de entalpia e fluxo mássico na entrada e pressão na saída. Para a simulação da circulação natural foi usado o primeiro caso, porém criou-se a possibilidade de se simular também outros tipos de geradores de vapor.

Escolheu-se o método para solucionar o sistema de equações. Foi usado o Método de Crout pela sua simplicidade quando se lida com matrizes tridiagonais. Foi desenvolvido um programa em linguagem C para resolução dos sistemas.

Desenvolveu-se então um programa em linguagem C para obtenção da densidade, temperatura e título da água para uma determinada pressão e entalpia. Para tanto, foram utilizadas as equações de IAPWS-97.

Com isso pôde-se simular a o escoamento natural em um gerador de vapor para três diferentes condições onde se variava o fornecimento de calor a um dos tubos. Obteve-se um caso (aquele para o qual se forneceu mais calor) onde o escoamento de ambos os tubos aquecidos descia. Em outro o escoamento inicia subindo e inverte seu sentido no decorrer do tempo. No último caso (o menos aquecido) o fluido subia pelo tubo com menor carga térmica.

Os resultados obtidos aparentam ter coerência, entretanto somente a utilização do programa ao longo do tempo é que trará segurança para seu uso e possibilitará eventuais melhorias e possíveis correções. Uma delas diz respeito à correção do fator de atrito devido ao escoamento bifásico. No presente trabalho usou-se o fator 2,3 para sua correção conforme recomendado por Ledinegg em [10] para esse tipo de escoamento. No entanto essa correção não leva em conta o título da mistura, dado disponível no método SIMPLE para cada uma das células, que torna possível uma correção mais refinada. Outra possível melhoria é a inserção do cálculo da viscosidade (usada na determinação do Reynolds para a obtenção do fator de atrito) nas equações das propriedades da água já que no presente programa ela é uma entrada a ser dada pelo usuário e não varia.

10 NOMENCLATURA

Tabela 7. Nomenclatura método SIMPLE.

Simbolo	Unidade	Descrição
L_i	kg/s	Coeficiente-SIMPLE na margem leste da célula i
O_i	kg/s	Coeficiente-SIMPLE na margem oeste de i
l_i^l	kg/s	Coeficiente-SIMPLE na margem leste da vizinha a leste de i
L_i	kg/s	Coeficiente-SIMPLE do vizinho a leste da célula i
C_i	kg/s	Coeficiente-SIMPLE da célula i
O_i	kg/s	Coeficiente-SIMPLE do vizinho a oeste da célula i
L_i^P	Ms	Coeficiente-SIMPLE da correção da pressão no vizinho a leste de i
C_i^P	Ms	Coeficiente-SIMPLE da correção da pressão em i
O_i^P	Ms	Coeficiente-SIMPLE da correção da pressão no vizinho a oeste de i
C_i^h	kg/s	Coeficiente-SIMPLE do balanço de energia em i
A	m^2	Área da secção transversal
K_i		Constante-SIMPLE da célula i
k_i^e	N	Constante-SIMPLE na margem leste da célula i
k_i^h	J/s	Constante-SIMPLE do balanço de energia da célula i
k_i^P	kg/s	Constante-SIMPLE da correção de pressão da célula i
m_i^l	m^2s/kg	Multiplicador-SIMPLE na margem leste da célula i
m_i^o	m^2s/kg	Multiplicador-SIMPLE na margem oeste da célula i
D	kg/m^2s	Fluxo difusivo
F	kg/m^2s	Fluxo convectivo
G_i	m/s^2	Parte da aceleração da gravidade na direção x
H	J/kg	Entalpia específica
p	Pa	Pressão
S		Termo fonte
S_c		Parte constante do termo fonte
S_p		Parte proporcional do termo fonte
S_{Ci}^p	Pa	Parte constante do termo fonte do balanço de momento em i
S_{Pi}^p	Pas/m	Parte proporcional do termo fonte do balanço de momento em i

Simbolo	Unidade	Descrição
S_{Ci}^{h}	W	Parte constante do termo fonte no balanço de energia em i
S_{Pi}^{h}	kg/s	Parte proporcional do termo fonte no balanço de energia em i
T	s	Tempo
T	K	Temperatura
W	m/s	Velocidade na direção do eixo
X	m	Coordenada de espaço
Δp	Pa	Diferença de pressão
Δp_R	Pa	Diferença de pressão resultante da diferença de cota
Δp_H	Pa	Diferença de pressão resultante do atrito
Δt	s	Passo no tempo
Δx	m	Passo no espaço
ϕ		Variavel genérica
Γ		Coeficiente de difusão

Tabela 8. Nomenclatura equações IAPWS-IF97

Simbolo	**Unidade**	**Descrição**
h	kJ/kg	Entalpia específica
I, J	-	Expoente
n	-	Coeficiente
p	MPa	Pressão
ps	MPa	Pressão de saturação
R	kJ/(kgK)	Constante dos gases
T	K	Temperatura
Ts	K	Temperatura de saturação
v	m^3/kg	Volume específico
β	-	Pressão transformada
δ	-	Pressão reduzida
ϕ	-	Adimensional Helmholz
γ	-	Adimensional Gibbs
π	-	Pressão reduzida
θ	-	Temperatura reduzida
ϑ	-	Temperatura transformada
ρ	kg/m^3	Densidade
τ	-	Temperatura reduzida inversa

11 BIBLIOGRAFIA

[1] PERA, H. "Geradores de Vapor". Editora Fama S/C Ltda, São Paulo, 1990.

[2] "Steam: its Generation and Use" The Babcock & Wilcox Company, New York, 1963.

[3] INCROPERA, F. P. DEWITT, D. P. "Fundamentos de Transferência de Calor e de Massa" LTC, Rio de Janeiro, 1998.

[4] PATANKAR, S.V. "Numerical Heat Transfer and Fluid Flow". Series in Computational Methods in Mechanics and Thermal Sciences. Hemisphere Publ. Corp., Washington, New York, London 1980.

[5] PONWEISER, K. "Numerische Simulation von dynamischen Strömungsvorgängen in netzwerkartigen Rohrkonstruktionen". Fortschr.-Ber. VDI, Reihe 6, Nr. 378, VDI Verlag, Düsseldorf 1997.

[6] AL-KHAFAJI, A. W. TOOLEY. J. R. "Numerical Methods in Engineering Practice". Harcourt Brace Jovanovich College Publishers., USA 1986.

[7] SCHWARZ, H.R. KÖCKLER, N. "Numerische Mathematik" B. G. Teubner-Verlag, Stuttgart 2004.

[8] SPANG, B. "Equations of IAPWS-IF97" http://www.cheresources.com/staff.shtml.

[9] FOX, R. W. MCDONALD A. L. "Introdução à Mecânica dos Fluidos", LTC Editora, Rio de Janeiro, 2001.

[10] LEDINEGG, M. "Dampferzeugung : Dampfkessel,Feuerungen einschliesslich Atomreaktoren", Springer, Wien, 1966.

12 ANEXO

```
///---------------------------------------------------------------------------
#include <iostream.h>
#include <conio.h>
#include <math.h>
#pragma hdrstop
#pragma argsused

        double T;
        double titulo;
        int i=1;
        double R=461.526;
        double Pred;
        double Tred;
        //Regiao1
        double CoefI1[38];
        double CoefJ1[38];
        double Coefn1[38];
        //Regiao2
        double CoefJ20[12];
        double Coefn20[12];
        double CoefI2[46];
        double CoefJ2[46];
        double Coefn2[46];
        //Regiao3
        double CoefI3[44];
```

```
double CoefJ3[44];
double Coefn3[44];
double dens3=0;
//Regiao4
double Upsilon;
double Beta;
double CoefA;
double CoefB;
double CoefC;
double CoefD;
double CoefE;
double CoefF;
double CoefG;
double Coefn4[15];
//Regiaob
double CoefnB[7];

double calculaH14(double);
double calculaH24(double);
double calculaH13(double);
double calculaH23(double);
double calculaH34(double);
double calculaTreg1(double, double);
double calculaTreg2(double, double);
double calculaTreg3(double, double);
void coefreg1();
void coefreg2();
void coefreg3();
```

```
        void coefreg4();
        void coefregb();
//-------------------------------------------------------------------------
void coefreg1()
        {

        i = 1;
        CoefI1[i++] = 0;
        CoefI1[i++] = 0;
        CoefI1[i++] = 0;
        CoefI1[i++] = 0;
        CoefI1[i++] = 0;
        CoefI1[i++] = 0;
        CoefI1[i++] = 0;
        CoefI1[i++] = 0;
        CoefI1[i++] = 1;
        CoefI1[i++] = 1;
        CoefI1[i++] = 1;
        CoefI1[i++] = 1;
        CoefI1[i++] = 1;
        CoefI1[i++] = 1;
        CoefI1[i++] = 2;
        CoefI1[i++] = 2;
        CoefI1[i++] = 2;
        CoefI1[i++] = 2;
        CoefI1[i++] = 2;
        CoefI1[i++] = 2;
        CoefI1[i++] = 3;
        CoefI1[i++] = 3;
```

```
CoefI1[i++] = 3;
CoefI1[i++] = 4;
CoefI1[i++] = 4;
CoefI1[i++] = 4;
CoefI1[i++] = 5;
CoefI1[i++] = 8;
CoefI1[i++] = 8;
CoefI1[i++] = 21;
CoefI1[i++] = 23;
CoefI1[i++] = 29;
CoefI1[i++] = 30;
CoefI1[i++] = 31;
CoefI1[i++] = 32;

i = 1;
CoefJ1[i++] = -2;
CoefJ1[i++] = -1;
CoefJ1[i++] = 0;
CoefJ1[i++] = 1;
CoefJ1[i++] = 2;
CoefJ1[i++] = 3;
CoefJ1[i++] = 4;
CoefJ1[i++] = 5;
CoefJ1[i++] = -9;
CoefJ1[i++] = -7;
CoefJ1[i++] = -1;
CoefJ1[i++] = 0;
CoefJ1[i++] = 1;
```

```
CoefJ1[i++] = 3;
CoefJ1[i++] = -3;
CoefJ1[i++] = 0;
CoefJ1[i++] = 1;
CoefJ1[i++] = 3;
CoefJ1[i++] = 17;
CoefJ1[i++] = -4;
CoefJ1[i++] = 0;
CoefJ1[i++] = 6;
CoefJ1[i++] = -5;
CoefJ1[i++] = -2;
CoefJ1[i++] = 10;
CoefJ1[i++] = -8;
CoefJ1[i++] = -11;
CoefJ1[i++] = -6;
CoefJ1[i++] = -29;
CoefJ1[i++] = -31;
CoefJ1[i++] = -38;
CoefJ1[i++] = -39;
CoefJ1[i++] = -40;
CoefJ1[i++] = -41;

i = 1;
Coefn1[i++] = 0.14632971213167;
Coefn1[i++] = -0.84548187169114;
Coefn1[i++] = -3.756360367204;
Coefn1[i++] = 3.3855169168385;
Coefn1[i++] = -0.95791963387872;
```

```
Coefn1[i++] = 0.15772038513228;
Coefn1[i++] = -0.016616417199501;
Coefn1[i++] = 8.1214629983568E-04;
Coefn1[i++] = 2.8319080123804E-04;
Coefn1[i++] = -6.0706301565874E-04;
Coefn1[i++] = -0.018990068218419;
Coefn1[i++] = -0.032529748770505;
Coefn1[i++] = -0.021841717175414;
Coefn1[i++] = -5.283835796993E-05;
Coefn1[i++] = -4.7184321073267E-04;
Coefn1[i++] = -3.0001780793026E-04;
Coefn1[i++] = 4.7661393906987E-05;
Coefn1[i++] = -4.4141845330846E-06;
Coefn1[i++] = -7.2694996297594E-16;
Coefn1[i++] = -3.1679644845054E-05;
Coefn1[i++] = -2.8270797985312E-06;
Coefn1[i++] = -8.5205128120103E-10;
Coefn1[i++] = -2.2425281908E-06;
Coefn1[i++] = -6.5171222895601E-07;
Coefn1[i++] = -1.4341729937924E-13;
Coefn1[i++] = -4.0516996860117E-07;
Coefn1[i++] = -1.2734301741641E-09;
Coefn1[i++] = -1.7424871230634E-10;
Coefn1[i++] = -6.8762131295531E-19;
Coefn1[i++] = 1.4478307828521E-20;
Coefn1[i++] = 2.6335781662795E-23;
Coefn1[i++] = -1.1947622640071E-23;
Coefn1[i++] = 1.8228094581404E-24;
```

```
        Coefn1[i++] = -9.3537087292458E-26;


        }
//---------------------------------------------------------------------------
void coefreg2()
{
        i=1;
        CoefJ20[i++] = 0;
        CoefJ20[i++] = 1;
        CoefJ20[i++] = -5;
        CoefJ20[i++] = -4;
        CoefJ20[i++] = -3;
        CoefJ20[i++] = -2;
        CoefJ20[i++] = -1;
        CoefJ20[i++] = 2;
        CoefJ20[i++] = 3;

        i=1;
        Coefn20[i++] = -9.6927686500217;
        Coefn20[i++] = 10.086655968018;
        Coefn20[i++] = -0.005608791128302;
        Coefn20[i++] = 0.071452738081455;
        Coefn20[i++] = -0.40710498223928;
        Coefn20[i++] = 1.4240819171444;
        Coefn20[i++] = -4.383951131945;
        Coefn20[i++] = -0.28408632460772;
        Coefn20[i++] = 0.021268463753307;
```

```
i=1;
CoefI2[i++] = 1;
CoefI2[i++] = 1;
CoefI2[i++] = 1;
CoefI2[i++] = 1;
CoefI2[i++] = 1;
CoefI2[i++] = 2;
CoefI2[i++] = 2;
CoefI2[i++] = 2;
CoefI2[i++] = 2;
CoefI2[i++] = 2;
CoefI2[i++] = 3;
CoefI2[i++] = 3;
CoefI2[i++] = 3;
CoefI2[i++] = 3;
CoefI2[i++] = 4;
CoefI2[i++] = 4;
CoefI2[i++] = 4;
CoefI2[i++] = 5;
CoefI2[i++] = 6;
CoefI2[i++] = 6;
CoefI2[i++] = 6;
CoefI2[i++] = 7;
CoefI2[i++] = 7;
CoefI2[i++] = 7;
CoefI2[i++] = 8;
```

```
CoefI2[i++] = 8;
CoefI2[i++] = 9;
CoefI2[i++] = 10;
CoefI2[i++] = 10;
CoefI2[i++] = 10;
CoefI2[i++] = 16;
CoefI2[i++] = 16;
CoefI2[i++] = 18;
CoefI2[i++] = 20;
CoefI2[i++] = 20;
CoefI2[i++] = 20;
CoefI2[i++] = 21;
CoefI2[i++] = 22;
CoefI2[i++] = 23;
CoefI2[i++] = 24;
CoefI2[i++] = 24;
CoefI2[i++] = 24;

i=1;
CoefJ2[i++] = 0;
CoefJ2[i++] = 1;
CoefJ2[i++] = 2;
CoefJ2[i++] = 3;
CoefJ2[i++] = 6;
CoefJ2[i++] = 1;
CoefJ2[i++] = 2;
CoefJ2[i++] = 4;
CoefJ2[i++] = 7;
```

```
CoefJ2[i++] = 36;
CoefJ2[i++] = 0;
CoefJ2[i++] = 1;
CoefJ2[i++] = 3;
CoefJ2[i++] = 6;
CoefJ2[i++] = 35;
CoefJ2[i++] = 1;
CoefJ2[i++] = 2;
CoefJ2[i++] = 3;
CoefJ2[i++] = 7;
CoefJ2[i++] = 3;
CoefJ2[i++] = 16;
CoefJ2[i++] = 35;
CoefJ2[i++] = 0;
CoefJ2[i++] = 11;
CoefJ2[i++] = 25;
CoefJ2[i++] = 8;
CoefJ2[i++] = 36;
CoefJ2[i++] = 13;
CoefJ2[i++] = 4;
CoefJ2[i++] = 10;
CoefJ2[i++] = 14;
CoefJ2[i++] = 29;
CoefJ2[i++] = 50;
CoefJ2[i++] = 57;
CoefJ2[i++] = 20;
CoefJ2[i++] = 35;
CoefJ2[i++] = 48;
```

```
CoefJ2[i++] = 21;
CoefJ2[i++] = 53;
CoefJ2[i++] = 39;
CoefJ2[i++] = 26;
CoefJ2[i++] = 40;
CoefJ2[i++] = 58;

i=1;
Coefn2[i++] = -1.7731742473213E-03;
Coefn2[i++] = -0.017834862292358;
Coefn2[i++] = -0.045996013696365;
Coefn2[i++] = -0.057581259083432;
Coefn2[i++] = -0.05032527872793;
Coefn2[i++] = -3.3032641670203E-05;
Coefn2[i++] = -1.8948987516315E-04;
Coefn2[i++] = -3.9392777243355E-03;
Coefn2[i++] = -0.043797295650573;
Coefn2[i++] = -2.6674547914087E-05;
Coefn2[i++] = 2.0481737692309E-08;
Coefn2[i++] = 4.3870667284435E-07;
Coefn2[i++] = -3.227767723857E-05;
Coefn2[i++] = -1.5033924542148E-03;
Coefn2[i++] = -0.040668253562649;
Coefn2[i++] = -7.8847309559367E-10;
Coefn2[i++] = 1.2790717852285E-08;
Coefn2[i++] = 4.8225372718507E-07;
Coefn2[i++] = 2.2922076337661E-06;
Coefn2[i++] = -1.6714766451061E-11;
```

```
        Coefn2[i++] = -2.1171472321355E-03;
        Coefn2[i++] = -23.895741934104;
        Coefn2[i++] = -5.905956432427E-18;
        Coefn2[i++] = -1.2621808899101E-06;
        Coefn2[i++] = -0.038946842435739;
        Coefn2[i++] = 1.1256211360459E-11;
        Coefn2[i++] = -8.2311340897998;
        Coefn2[i++] = 1.9809712802088E-08;
        Coefn2[i++] = 1.0406965210174E-19;
        Coefn2[i++] = -1.0234747095929E-13;
        Coefn2[i++] = -1.0018179379511E-09;
        Coefn2[i++] = -8.0882908646985E-11;
        Coefn2[i++] = 0.10693031879409;
        Coefn2[i++] = -0.33662250574171;
        Coefn2[i++] = 8.9185845355421E-25;
        Coefn2[i++] = 3.0629316876232E-13;
        Coefn2[i++] = -4.2002467698208E-06;
        Coefn2[i++] = -5.9056029685639E-26;
        Coefn2[i++] = 3.7826947613457E-06;
        Coefn2[i++] = -1.2768608934681E-15;
        Coefn2[i++] = 7.3087610595061E-29;
        Coefn2[i++] = 5.5414715350778E-17;
        Coefn2[i++] = -9.436970724121E-07;

}
//---------------------------------------------------------------------------
void coefreg3()
{
```

```
i=1;
CoefI3[i++] = 0;
CoefI3[i++] = 0;
CoefI3[i++] = 0;
CoefI3[i++] = 0;
CoefI3[i++] = 0;
CoefI3[i++] = 0;
CoefI3[i++] = 0;
CoefI3[i++] = 0;
CoefI3[i++] = 1;
CoefI3[i++] = 1;
CoefI3[i++] = 1;
CoefI3[i++] = 1;
CoefI3[i++] = 2;
CoefI3[i++] = 2;
CoefI3[i++] = 2;
CoefI3[i++] = 2;
CoefI3[i++] = 2;
CoefI3[i++] = 2;
CoefI3[i++] = 3;
CoefI3[i++] = 3;
CoefI3[i++] = 3;
CoefI3[i++] = 3;
CoefI3[i++] = 3;
CoefI3[i++] = 4;
CoefI3[i++] = 4;
CoefI3[i++] = 4;
CoefI3[i++] = 4;
```

```
CoefI3[i++] = 5;
CoefI3[i++] = 5;
CoefI3[i++] = 5;
CoefI3[i++] = 6;
CoefI3[i++] = 6;
CoefI3[i++] = 6;
CoefI3[i++] = 7;
CoefI3[i++] = 8;
CoefI3[i++] = 9;
CoefI3[i++] = 9;
CoefI3[i++] = 10;
CoefI3[i++] = 10;
CoefI3[i++] = 11;

i=1;
CoefJ3[i++] = 0;
CoefJ3[i++] = 0;
CoefJ3[i++] = 1;
CoefJ3[i++] = 2;
CoefJ3[i++] = 7;
CoefJ3[i++] = 10;
CoefJ3[i++] = 12;
CoefJ3[i++] = 23;
CoefJ3[i++] = 2;
CoefJ3[i++] = 6;
CoefJ3[i++] = 15;
CoefJ3[i++] = 17;
CoefJ3[i++] = 0;
```

```
CoefJ3[i++] = 2;
CoefJ3[i++] = 6;
CoefJ3[i++] = 7;
CoefJ3[i++] = 22;
CoefJ3[i++] = 26;
CoefJ3[i++] = 0;
CoefJ3[i++] = 2;
CoefJ3[i++] = 4;
CoefJ3[i++] = 16;
CoefJ3[i++] = 26;
CoefJ3[i++] = 0;
CoefJ3[i++] = 2;
CoefJ3[i++] = 4;
CoefJ3[i++] = 26;
CoefJ3[i++] = 1;
CoefJ3[i++] = 3;
CoefJ3[i++] = 26;
CoefJ3[i++] = 0;
CoefJ3[i++] = 2;
CoefJ3[i++] = 26;
CoefJ3[i++] = 2;
CoefJ3[i++] = 26;
CoefJ3[i++] = 2;
CoefJ3[i++] = 26;
CoefJ3[i++] = 0;
CoefJ3[i++] = 1;
CoefJ3[i++] = 26;
```

```
i=1;
Coefn3[i++] = 1.0658070028513;
Coefn3[i++] = -15.732845290239;
Coefn3[i++] = 20.944396974307;
Coefn3[i++] = -7.6867707878716;
Coefn3[i++] = 2.6185947787954;
Coefn3[i++] = -2.808078114862;
Coefn3[i++] = 1.2053369696517;
Coefn3[i++] = -8.4566812812502E-03;
Coefn3[i++] = -1.2654315477714;
Coefn3[i++] = -1.1524407806681;
Coefn3[i++] = 0.88521043984318;
Coefn3[i++] = -0.64207765181607;
Coefn3[i++] = 0.38493460186671;
Coefn3[i++] = -0.85214708824206;
Coefn3[i++] = 4.8972281541877;
Coefn3[i++] = -3.0502617256965;
Coefn3[i++] = 0.039420536879154;
Coefn3[i++] = 0.12558408424308;
Coefn3[i++] = -0.2799932969871;
Coefn3[i++] = 1.389979956946;
Coefn3[i++] = -2.018991502357;
Coefn3[i++] = -8.2147637173963E-03;
Coefn3[i++] = -0.47596035734923;
Coefn3[i++] = 0.0439840744735;
Coefn3[i++] = -0.44476435428739;
Coefn3[i++] = 0.90572070719733;
Coefn3[i++] = 0.70522450087967;
```

```
        Coefn3[i++] = 0.10770512626332;
        Coefn3[i++] = -0.32913623258954;
        Coefn3[i++] = -0.50871062041158;
        Coefn3[i++] = -0.022175400873096;
        Coefn3[i++] = 0.094260751665092;
        Coefn3[i++] = 0.16436278447961;
        Coefn3[i++] = -0.013503372241348;
        Coefn3[i++] = -0.014834345352472;
        Coefn3[i++] = 5.7922953628084E-04;
        Coefn3[i++] = 3.2308904703711E-03;
        Coefn3[i++] = 8.0964802996215E-05;
        Coefn3[i++] = -1.6557679795037E-04;
        Coefn3[i++] = -4.4923899061815E-05;

}
//---------------------------------------------------------------------------
void coefreg4()
{

        i=1;
        Coefn4[i++] = 1167.0521452767;
        Coefn4[i++] = -724213.16703206;
        Coefn4[i++] = -17.073846940092;
        Coefn4[i++] = 12020.82470247;
        Coefn4[i++] = -3232555.0322333;
        Coefn4[i++] = 14.91510861353;
        Coefn4[i++] = -4823.2657361591;
        Coefn4[i++] = 405113.40542057;
```

```
        Coefn4[i++] = -0.23855557567849;
        Coefn4[i++] = 650.17534844798;

}
//---------------------------------------------------------------------------
void coefregb()
        {
        i=1;
        CoefnB[i++] = 348.05185628969;
        CoefnB[i++] = -1.1671859879975;
        CoefnB[i++] = 1.0192970039326E-03;
        CoefnB[i++] = 572.54459862746;
        CoefnB[i++] = 13.91883977887;
        }

//---------------------------------------------------------------------------
        double calculaH14(double P14)
        {
        double T14;
        double H14;

        coefreg4();

        Beta=pow(P14,0.25);
        CoefG=Coefn4[2]*pow(Beta,2)+Coefn4[5]*Beta+Coefn4[8];
        CoefF=Coefn4[1]*pow(Beta,2)+Coefn4[4]*Beta+Coefn4[7];
        CoefE=pow(Beta,2)+Coefn4[3]*Beta+Coefn4[6];
        CoefD=(2*CoefG)/(-CoefF-pow((pow(CoefF,2)-4*CoefE*CoefG),0.5));
```

```
        T14=(Coefn4[10]+CoefD-pow((pow(Coefn4[10]+CoefD,2)-
4*(Coefn4[9]+Coefn4[10]*CoefD)),0.5))/2;

        coefreg1();

        Pred=P14/16.53;
        Tred=1386/T14;

        double Gib_ad_tau=0;

       for (int i=1;i<35;i++)
          {
          Gib_ad_tau=Gib_ad_tau+Coefn1[i]*pow((7.1-
Pred),CoefI1[i])*CoefJ1[i]*pow((Tred-1.222),(CoefJ1[i]-1));
          }
        H14=Tred*Gib_ad_tau*R*T14;

        return H14;
        }
//---------------------------------------------------------------------------
        double calculaTreg4(double P14)
        {
        double T14;

        coefreg4();

        Beta=pow(P14,0.25);
        CoefG=Coefn4[2]*pow(Beta,2)+Coefn4[5]*Beta+Coefn4[8];
```

```
        CoefF=Coefn4[1]*pow(Beta,2)+Coefn4[4]*Beta+Coefn4[7];
        CoefE=pow(Beta,2)+Coefn4[3]*Beta+Coefn4[6];
        CoefD=(2*CoefG)/(-CoefF-pow((pow(CoefF,2)-4*CoefE*CoefG),0.5));
        T14=(Coefn4[10]+CoefD-pow((pow(Coefn4[10]+CoefD,2)-
4*(Coefn4[9]+Coefn4[10]*CoefD)),0.5))/2;
        return T14;
        }
//---------------------------------------------------------------------------
        double calculaH24(double P24)
        {
        double T24;
        double H24;

        coefreg4();

        Beta=pow(P24,0.25);
        CoefG=Coefn4[2]*pow(Beta,2)+Coefn4[5]*Beta+Coefn4[8];
        CoefF=Coefn4[1]*pow(Beta,2)+Coefn4[4]*Beta+Coefn4[7];
        CoefE=pow(Beta,2)+Coefn4[3]*Beta+Coefn4[6];
        CoefD=(2*CoefG)/(-CoefF-pow((pow(CoefF,2)-4*CoefE*CoefG),0.5));
        T24=(Coefn4[10]+CoefD-pow((pow(Coefn4[10]+CoefD,2)-
4*(Coefn4[9]+Coefn4[10]*CoefD)),0.5))/2;

        coefreg2();

        double Gib_ad_pi0=0;
        double Gib_ad_tau0=0;
        double Gib_ad_piR=0;
```

```
        double Gib_ad_tauR=0;

        Tred=540/T24;
        Pred=P24;
        Gib_ad_pi0=1/Pred;

        for (i=1;i<10;i++)
         {
         Gib_ad_tau0=Gib_ad_tau0+Coefn20[i]*CoefJ20[i]*pow(Tred,CoefJ20[i]-1);
         }

        for (i=1;i<44;i++)
         {

Gib_ad_tauR=Gib_ad_tauR+Coefn2[i]*pow(Pred,CoefI2[i])*CoefJ2[i]*pow(Tred-
0.5,CoefJ2[i]-1);
         }

         H24=R*T24*Tred*(Gib_ad_tauR+Gib_ad_tau0);


        return H24;
        }
//---------------------------------------------------------------------------
        double calculaH13(double P13)
        {
        double T13;
        double H13;
```

```
        T13=623.15;
        coefreg1();

        Pred=P13/16.53;
        Tred=1386/T13;

        double Gib_ad_tau=0;

       for (int i=1;i<35;i++)
          {
          Gib_ad_tau=Gib_ad_tau+Coefn1[i]*pow((7.1-
Pred),CoefI1[i])*CoefJ1[i]*pow((Tred-1.222),(CoefJ1[i]-1));
          }
        H13=Tred*Gib_ad_tau*R*T13;

        return H13;
        }
//---------------------------------------------------------------------------
        double calculaH23(double P23)
        {
        double T23;
        double H23;

        coefregb();

        T23=CoefnB[4]+pow((P23-CoefnB[5])/CoefnB[3],0.5);
```

```
coefreg2();

double Gib_ad_pi0=0;
double Gib_ad_tau0=0;
double Gib_ad_piR=0;
double Gib_ad_tauR=0;

Tred=540/T23;
Pred=P23;
Gib_ad_pi0=1/Pred;

for (i=1;i<10;i++)
 {
 Gib_ad_tau0=Gib_ad_tau0+Coefn20[i]*CoefJ20[i]*pow(Tred,CoefJ20[i]-1);
 }

for (i=1;i<44;i++)
 {

Gib_ad_tauR=Gib_ad_tauR+Coefn2[i]*pow(Pred,CoefI2[i])*CoefJ2[i]*pow(Tred-
0.5,CoefJ2[i]-1);
 }

 H23=R*T23*Tred*(Gib_ad_tauR+Gib_ad_tau0);


return H23;
}
```

```
//---------------------------------------------------------------------------
        double calculaH34n(double P34)
        {
        double T34;
        double H34;
        double Ps;

        coefreg4();

        Beta=pow(P34,0.25);
        CoefG=Coefn4[2]*pow(Beta,2)+Coefn4[5]*Beta+Coefn4[8];
        CoefF=Coefn4[1]*pow(Beta,2)+Coefn4[4]*Beta+Coefn4[7];
        CoefE=pow(Beta,2)+Coefn4[3]*Beta+Coefn4[6];
        CoefD=(2*CoefG)/(-CoefF-pow((pow(CoefF,2)-4*CoefE*CoefG),0.5));
        T34=(Coefn4[10]+CoefD-pow((pow(Coefn4[10]+CoefD,2)-
4*(Coefn4[9]+Coefn4[10]*CoefD)),0.5))/2;
        T34=T34-0.01;

        coefreg3();
         double densred34=0;
         double dens34=0;
         double densant34=0;
         double delpdelrho=0;
         double Helm_ad_delta=0;
         double Helm_ad_deltadelta=0;
         double Helm_ad_tau=0;
         double Helm_ad_tautau=0;
         double Edensreg3=0;
```

```
Tred=647.096/T34;

densant34=600;
for (int idensreg3=1;idensreg3<1000;idensreg3++)
  {
  densred34=densant34/322;
  Helm_ad_delta=Coefn3[1]/densred34;
  for (i=2;i<41;i++)
    {

Helm_ad_delta=Helm_ad_delta+Coefn3[i]*CoefI3[i]*pow(densred34,CoefI3[i]-
1)*pow(Tred,CoefJ3[i]);
    }

  Helm_ad_deltadelta=-Coefn3[1]/pow(densred34,2);
  for (i=2;i<41;i++)
    {
    Helm_ad_deltadelta=Helm_ad_deltadelta+Coefn3[i]*CoefI3[i]*(CoefI3[i]-
1)*pow(densred34,CoefI3[i]-2)*pow(Tred,CoefJ3[i]);
    }


delpdelrho=(R*T34/322)*(2*densant34*Helm_ad_delta+pow(densant34,2)*Helm_ad_d
eltadelta/322);
    dens34=densant34+(1000000*P34-
densant34*R*T34*Helm_ad_delta*densred34)/delpdelrho;
    Edensreg3=pow(pow(dens34-densant34,2),0.5);
```

```
        if (Edensreg3<0.000005){
           idensreg3=1000;}

        densant34=dens34;
        }

      densred34=dens34/322;

      for (i=2;i<41;i++)
          {

Helm_ad_tau=Helm_ad_tau+Coefn3[i]*pow(densred34,CoefI3[i])*CoefJ3[i]*pow(Tred
,CoefJ3[i]-1);
          }
      H34=R*T34*(Tred*Helm_ad_tau+densred34*Helm_ad_delta);


    return H34;
    }

//---------------------------------------------------------------------

double calculaH34p(double P34)
      {
      double T34;
      double H34;
```

```
        coefreg4();

        Beta=pow(P34,0.25);
        CoefG=Coefn4[2]*pow(Beta,2)+Coefn4[5]*Beta+Coefn4[8];
        CoefF=Coefn4[1]*pow(Beta,2)+Coefn4[4]*Beta+Coefn4[7];
        CoefE=pow(Beta,2)+Coefn4[3]*Beta+Coefn4[6];
        CoefD=(2*CoefG)/(-CoefF-pow((pow(CoefF,2)-4*CoefE*CoefG),0.5));
        T34=(Coefn4[10]+CoefD-pow((pow(Coefn4[10]+CoefD,2)-
4*(Coefn4[9]+Coefn4[10]*CoefD)),0.5))/2;
        T34=T34+0.01;

        coefreg3();
         double densred34=0;
         double dens34=0;
         double densant34=0;
         double delpdelrho=0;
         double Helm_ad_delta=0;
         double Helm_ad_deltadelta=0;
         double Helm_ad_tau=0;
         double Helm_ad_tautau=0;
         double Edensreg3=0;

         Tred=647.096/T34;
         densant34=100;
         for (int idensreg3=1;idensreg3<1000;idensreg3++)
           {
           densred34=densant34/322;
           Helm_ad_delta=Coefn3[1]/densred34;
```

```
            for (i=2;i<41;i++)
              {

Helm_ad_delta=Helm_ad_delta+Coefn3[i]*CoefI3[i]*pow(densred34,CoefI3[i]-
1)*pow(Tred,CoefJ3[i]);
              }

            Helm_ad_deltadelta=-Coefn3[1]/pow(densred34,2);
            for (i=2;i<41;i++)
              {
              Helm_ad_deltadelta=Helm_ad_deltadelta+Coefn3[i]*CoefI3[i]*(CoefI3[i]-
1)*pow(densred34,CoefI3[i]-2)*pow(Tred,CoefJ3[i]);
              }


delpdelrho=(R*T34/322)*(2*densant34*Helm_ad_delta+pow(densant34,2)*Helm_ad_d
eltadelta/322);
            dens34=densant34+(1000000*P34-
densant34*R*T34*Helm_ad_delta*densred34)/delpdelrho;
            Edensreg3=pow(pow(dens34-densant34,2),0.5);

            if (Edensreg3<0.000005){
              idensreg3=1000;}

            densant34=dens34;
            }

          densred34=dens34/322;
```

```
        for (i=2;i<41;i++)
            {

Helm_ad_tau=Helm_ad_tau+Coefn3[i]*pow(densred34,CoefI3[i])*CoefJ3[i]*pow(Tred
,CoefJ3[i]-1);
            }
        H34=R*T34*(Tred*Helm_ad_tau+densred34*Helm_ad_delta);


    return H34;
    }
//---------------------------------------------------------------------------
    double calculaTreg1(double Pct1, double Hct1)
     {
     double Tct1=0;
     double Tct1ant=450;
     double Tct1red=0;
     double Pct1red=0;
     double Gib_ad_tau=0;
     double delhdelT1=0;
     double ETct1=0;
     double Tct1reddeltap=0;
     double Tct1reddeltan=0;
     double Hct1deltap=0;
     double Hct1deltan=0;
     double Gib_ad_taup=0;
     double Gib_ad_taun=0;
```

```
coefreg1();

Pct1red=Pct1/16.53;

for (int iTct1=1;iTct1<1000;iTct1++)
{
Tct1red=1386/Tct1ant;
Tct1reddeltap=1386/(Tct1ant+1);
Tct1reddeltan=1386/(Tct1ant-1);
Gib_ad_tau=0;
Gib_ad_taup=0;
Gib_ad_taun=0;
for (int i=1;i<35;i++)
  {
  Gib_ad_tau=Gib_ad_tau+Coefn1[i]*pow((7.1-
Pct1red),CoefI1[i])*CoefJ1[i]*pow((Tct1red-1.222),(CoefJ1[i]-1));
  }
for (int i=1;i<35;i++)
  {
  Gib_ad_taup=Gib_ad_taup+Coefn1[i]*pow((7.1-
Pct1red),CoefI1[i])*CoefJ1[i]*pow((Tct1reddeltap-1.222),(CoefJ1[i]-1));
  }
for (int i=1;i<35;i++)
  {
  Gib_ad_taun=Gib_ad_taun+Coefn1[i]*pow((7.1-
Pct1red),CoefI1[i])*CoefJ1[i]*pow((Tct1reddeltan-1.222),(CoefJ1[i]-1));
```

```
        }

      Tct1reddeltap=1386/(Tct1ant+1);
      Tct1reddeltan=1386/(Tct1ant-1);
      Hct1deltap=Tct1reddeltap*Gib_ad_taup*R*(Tct1ant+1);
      Hct1deltan=Tct1reddeltan*Gib_ad_taun*R*(Tct1ant-1);
      delhdelT1=(Hct1deltap-Hct1deltan)/2;

      Tct1=Tct1ant+(Hct1-Tct1red*Gib_ad_tau*R*Tct1ant)/delhdelT1;
      ETct1=pow(pow(Tct1-Tct1ant,2),0.5);
      if (ETct1<0.0005)
        {
        iTct1=10000;
        }
      Tct1ant=Tct1;
      }

      return Tct1;
       }
//-------------------------------------------------------------------------
     double calculaTreg2(double Pct2, double Hct2)
      {
      double Tct2=0;
      double Tct2ant=800;
      double Tct2red=0;
      double Pct2red=0;
      double Gib_ad_tau0=0;
      double Gib_ad_tauR=0;
```

```
double Gib_ad_tau0p=0;
double Gib_ad_tauRp=0;
double Gib_ad_tau0n=0;
double Gib_ad_tauRn=0;
double delhdelT2=0;
double ETct2=0;
double Tct2reddeltap=0;
double Tct2reddeltan=0;
double Hct2deltap=0;
double Hct2deltan=0;


coefreg2();

Pct2red=Pct2/1;

for (int iTct2=1;iTct2<1000;iTct2++)
{

 Tct2red=540/Tct2ant;
 Tct2reddeltap=540/(Tct2ant+0.1);
 Tct2reddeltan=540/(Tct2ant-0.1);

 Gib_ad_tau0=0;
 Gib_ad_tauR=0;
 Gib_ad_tau0p=0;
 Gib_ad_tauRp=0;
 Gib_ad_tau0n=0;
```

```
Gib_ad_tauRn=0;

for (i=1;i<10;i++)
 {
 Gib_ad_tau0=Gib_ad_tau0+Coefn20[i]*CoefJ20[i]*pow(Tct2red,CoefJ20[i]-
1);
 }

for (i=1;i<44;i++)
 {

Gib_ad_tauR=Gib_ad_tauR+Coefn2[i]*pow(Pct2red,CoefI2[i])*CoefJ2[i]*pow(Tct2red
-0.5,CoefJ2[i]-1);
 }

for (i=1;i<10;i++)
 {

Gib_ad_tau0p=Gib_ad_tau0p+Coefn20[i]*CoefJ20[i]*pow(Tct2reddeltap,CoefJ20[i]-
1);
 }

for (i=1;i<44;i++)
 {

Gib_ad_tauRp=Gib_ad_tauRp+Coefn2[i]*pow(Pct2red,CoefI2[i])*CoefJ2[i]*pow(Tct2r
eddeltap-0.5,CoefJ2[i]-1);
```

```
        }

        for (i=1;i<10;i++)
        {

Gib_ad_tau0n=Gib_ad_tau0n+Coefn20[i]*CoefJ20[i]*pow(Tct2reddeltan,CoefJ20[i]-
1);
        }

        for (i=1;i<44;i++)
        {

Gib_ad_tauRn=Gib_ad_tauRn+Coefn2[i]*pow(Pct2red,CoefI2[i])*CoefJ2[i]*pow(Tct2r
eddeltan-0.5,CoefJ2[i]-1);
        }


        Hct2deltap=R*(Tct2ant+0.1)*Tct2reddeltap*(Gib_ad_tauRp+Gib_ad_tau0p);
        Hct2deltan=R*(Tct2ant-0.1)*Tct2reddeltan*(Gib_ad_tauRn+Gib_ad_tau0n);
        delhdelT2=(Hct2deltap-Hct2deltan)/0.2;

        Tct2=Tct2ant+(Hct2-
R*Tct2ant*Tct2red*(Gib_ad_tauR+Gib_ad_tau0))/delhdelT2;
        ETct2=pow(pow(Tct2-Tct2ant,2),0.5);

        if (ETct2<0.0005)
          {
          iTct2=10000;
```

```
    }
 Tct2ant=Tct2;
 }

 return Tct2;
 }

//-------------------------------------------------------------------------
 double calculaTreg3(double Pct3, double Hct3)
 {
 coefreg3();
 double dens3antigo=0;
 double Hf=0;
 Hf=calculaH34p(Pct3);
 if (Hct3>=Hf&&Pct3<22.064)
  {
  dens3antigo=100;
  }
 else
  {
  dens3antigo=600;
  }

 double Tct3=0;
 double Tct3antigo=630;
 double Tct3red=0;
 double Tct3reddeltap=0;
 double Tct3reddeltan=0;
```

```
dens3=0;
double dens3red=0;
double dens3reddeltap=0;
double dens3reddeltan=0;
double delhdelt3=0;
double delhdeldens3=0;
double delpdelt3=0;
double delpdeldens3=0;
double Hct3p_T=0;
double Hct3n_T=0;
double Hct3p_dens=0;
double Hct3n_dens=0;
double Pct3p_T=0;
double Pct3n_T=0;
double Pct3p_dens=0;
double Pct3n_dens=0;
double ETct3=0;
double Edens3=0;
double Helm_ad_delta=0;
double Helm_ad_delta_Tp=0;
double Helm_ad_delta_Tn=0;
double Helm_ad_delta_densp=0;
double Helm_ad_delta_densn=0;
double Helm_ad_tau=0;
double Helm_ad_tau_Tp=0;
double Helm_ad_tau_Tn=0;
double Helm_ad_tau_densp=0;
double Helm_ad_tau_densn=0;
```

```
  double Hct3calc=0;
  double Pct3calc=0;
  double eeCoef[4][4];
  double eeCoefcof[4][4];
  double eeCoefcoftrans[4][4];
  double eeCoefinv[4][4];
  double invdet;

for (int iTct3=1;iTct3<1000;iTct3++)
{

  Tct3red=647.096/Tct3antigo;
  Tct3reddeltap=647.096/(Tct3antigo+0.1);
  Tct3reddeltan=647.096/(Tct3antigo-0.1);

  dens3red=dens3antigo/322;
  dens3reddeltap=(dens3antigo+0.1)/322;
  dens3reddeltan=(dens3antigo-0.1)/322;

  Helm_ad_delta=0;
  Helm_ad_delta_Tp=0;
  Helm_ad_delta_Tn=0;
  Helm_ad_delta_densp=0;
  Helm_ad_delta_densn=0;
  Helm_ad_tau=0;
  Helm_ad_tau_Tp=0;
  Helm_ad_tau_Tn=0;
  Helm_ad_tau_densp=0;
```

```
        Helm_ad_tau_densn=0;

        Helm_ad_delta=Coefn3[1]/dens3red;
            for (i=2;i<41;i++)
               {

Helm_ad_delta=Helm_ad_delta+Coefn3[i]*CoefI3[i]*pow(dens3red,CoefI3[i]-
1)*pow(Tct3red,CoefJ3[i]);
               }
        //dens fixa e T variando
        Helm_ad_delta_Tp=Coefn3[1]/dens3red;
            for (i=2;i<41;i++)
               {

Helm_ad_delta_Tp=Helm_ad_delta_Tp+Coefn3[i]*CoefI3[i]*pow(dens3red,CoefI3[i]-
1)*pow(Tct3reddeltap,CoefJ3[i]);
               }

        Helm_ad_delta_Tn=Coefn3[1]/dens3red;
            for (i=2;i<41;i++)
               {

Helm_ad_delta_Tn=Helm_ad_delta_Tn+Coefn3[i]*CoefI3[i]*pow(dens3red,CoefI3[i]-
1)*pow(Tct3reddeltan,CoefJ3[i]);
               }

        //T fixa e dens variando
        Helm_ad_delta_densp=Coefn3[1]/dens3reddeltap;
```

```
        for (i=2;i<41;i++)
          {

Helm_ad_delta_densp=Helm_ad_delta_densp+Coefn3[i]*CoefI3[i]*pow(dens3reddelta
p,CoefI3[i]-1)*pow(Tct3red,CoefJ3[i]);
          }

      Helm_ad_delta_densn=Coefn3[1]/dens3reddeltan;
        for (i=2;i<41;i++)
          {

Helm_ad_delta_densn=Helm_ad_delta_densn+Coefn3[i]*CoefI3[i]*pow(dens3reddelta
n,CoefI3[i]-1)*pow(Tct3red,CoefJ3[i]);
          }

      for (i=2;i<41;i++)
          {

Helm_ad_tau=Helm_ad_tau+Coefn3[i]*pow(dens3red,CoefI3[i])*CoefJ3[i]*pow(Tct3r
ed,CoefJ3[i]-1);
          }

      //dens fixa e T variando

      for (i=2;i<41;i++)
          {
```

```
Helm_ad_tau_Tp=Helm_ad_tau_Tp+Coefn3[i]*pow(dens3red,CoefI3[i])*CoefJ3[i]*po
w(Tct3reddeltap,CoefJ3[i]-1);
                }

        for (i=2;i<41;i++)
                {

Helm_ad_tau_Tn=Helm_ad_tau_Tn+Coefn3[i]*pow(dens3red,CoefI3[i])*CoefJ3[i]*po
w(Tct3reddeltan,CoefJ3[i]-1);
                }

        //T fixa e dens variando

        for (i=2;i<41;i++)
                {

Helm_ad_tau_densp=Helm_ad_tau_densp+Coefn3[i]*pow(dens3reddeltap,CoefI3[i])*C
oefJ3[i]*pow(Tct3red,CoefJ3[i]-1);
                }

        for (i=2;i<41;i++)
                {

Helm_ad_tau_densn=Helm_ad_tau_densn+Coefn3[i]*pow(dens3reddeltan,CoefI3[i])*C
oefJ3[i]*pow(Tct3red,CoefJ3[i]-1);
                }
```

```
Hct3p_T=R*(Tct3antigo+0.1)*(Tct3reddeltap*Helm_ad_tau_Tp+dens3red*Helm_ad_d
elta_Tp);
        Hct3n_T=R*(Tct3antigo-
0.1)*(Tct3reddeltan*Helm_ad_tau_Tn+dens3red*Helm_ad_delta_Tn);

Hct3p_dens=R*Tct3antigo*(Tct3red*Helm_ad_tau_densp+dens3reddeltap*Helm_ad_d
elta_densp);

Hct3n_dens=R*Tct3antigo*(Tct3red*Helm_ad_tau_densn+dens3reddeltan*Helm_ad_d
elta_densn);
        Pct3p_T=dens3antigo*R*(Tct3antigo+0.1)*dens3red*Helm_ad_delta_Tp;
        Pct3n_T=dens3antigo*R*(Tct3antigo-0.1)*dens3red*Helm_ad_delta_Tn;

Pct3p_dens=(dens3antigo+0.1)*R*Tct3antigo*dens3reddeltap*Helm_ad_delta_densp;
        Pct3n_dens=(dens3antigo-
0.1)*R*Tct3antigo*dens3reddeltan*Helm_ad_delta_densn;

        delhdelt3=(Hct3p_T-Hct3n_T)/0.2;
        delhdeldens3=(Hct3p_dens-Hct3n_dens)/0.2;
        delpdelt3=(Pct3p_T-Pct3n_T)/0.2;
        delpdeldens3=(Pct3p_dens-Pct3n_dens)/0.2;

        Hct3calc=R*Tct3antigo*(Tct3red*Helm_ad_tau+dens3red*Helm_ad_delta)-
Hct3;
        Pct3calc=dens3antigo*R*Tct3antigo*dens3red*Helm_ad_delta-Pct3*1000000;

        eeCoef[0][0]=delhdelt3;
```

```
eeCoef[0][1]=delhdeldens3;
eeCoef[1][0]=delpdelt3;
eeCoef[1][1]=delpdeldens3;

invdet=1/(eeCoef[0][0]*eeCoef[1][1]-eeCoef[1][0]*eeCoef[0][1]);
eeCoefcof[0][0]=eeCoef[1][1];
eeCoefcof[0][1]=-eeCoef[1][0];
eeCoefcof[1][0]=-eeCoef[0][1];
eeCoefcof[1][1]=eeCoef[0][0];

eeCoefcoftrans[0][0]=eeCoefcof[0][0];
eeCoefcoftrans[0][1]=eeCoefcof[1][0];
eeCoefcoftrans[1][0]=eeCoefcof[0][1];
eeCoefcoftrans[1][1]=eeCoefcof[1][1];

eeCoefinv[0][0]=eeCoefcoftrans[0][0]*invdet;
eeCoefinv[0][1]=eeCoefcoftrans[0][1]*invdet;
eeCoefinv[1][0]=eeCoefcoftrans[1][0]*invdet;
eeCoefinv[1][1]=eeCoefcoftrans[1][1]*invdet;

Tct3=Tct3antigo-(eeCoefinv[0][0]*Hct3calc+eeCoefinv[0][1]*Pct3calc);
dens3=dens3antigo-(eeCoefinv[1][0]*Hct3calc+eeCoefinv[1][1]*Pct3calc);

ETct3=pow(pow(Tct3-Tct3antigo,2),0.5);
Edens3=pow(pow(dens3-dens3antigo,2),0.5);

if (ETct3<0.0005&&Edens3<0.0005)
   {
```

```
            iTct3=10000;
            }
          Tct3antigo=Tct3;
          dens3antigo=dens3;
         }

    return Tct3;
    }
//---------------------------------------------------------------------------
          double calculadens1(double T, double P)
          {

          coefreg1();

          double vol_esp=0;
          double densreg1=0;

          Pred=P/16.53;
          Tred=1386/T;

          double Gib_ad_pi=0;

          for (int a=1;a<35;a++)
           {
            Gib_ad_pi=Gib_ad_pi-1*Coefn1[a]*CoefI1[a]*pow((7.1-Pred),(CoefI1[a]-
1))*pow((Tred-1.222),(CoefJ1[a]));
            }
          vol_esp=Pred*Gib_ad_pi*R*T/(P*1000000);
```

```
densreg1=1/vol_esp;

return densreg1;
}

//-------------------------------------------------------------------------
double calculadens2(double T, double P)
{

coefreg2();

double vol_esp=0;
double densreg2=0;

Tred=540/T;
Pred=P;

double Gib_ad_piR=0;
double Gib_ad_pi0=0;

Gib_ad_pi0=1/Pred;

for (i=1;i<44;i++)
 {
 Gib_ad_piR=Gib_ad_piR+Coefn2[i]*CoefI2[i]*pow(Pred,CoefI2[i]-
1)*pow(Tred-0.5,CoefJ2[i]);
 }
```

```
        vol_esp=Pred*R*T*(Gib_ad_pi0+Gib_ad_piR)/(P*1000000);
        densreg2=1/vol_esp;

      return densreg2;
      }

//---------------------------------------------------------------------------
double calculadens(double P, double entalpia_esp)
{

    dens3=0;
    T=0;
    titulo=0;
    double densidade=0;
    double Tmax=0;
    double Tmin=0;
    double densmax=0;
    double densmin=0;


    if (P<=16.529164)
    {
    double Hfront14;
    Hfront14=calculaH14(P);
    //cout << "(<=16.529)Hfront14 :" << Hfront14 <<endl;

    double Hfront24;
    Hfront24=calculaH24(P);
```

```
//cout << "(<=16.529)Hfront24 :" << Hfront24 <<endl;

    if (entalpia_esp<=Hfront14)
    {
    T=calculaTreg1(P,entalpia_esp);
    titulo=0;
    densidade=calculadens1(T,P);
    //cout << "(<=16.529)T :" << T <<endl;
    //cout << "titulo :" << titulo <<endl;
    //cout << "densidade :" << densidade <<endl;
    }

    if (entalpia_esp>Hfront14&&entalpia_esp<Hfront24)
    {
    T=calculaTreg4(P);
    titulo=(entalpia_esp-Hfront14)/(Hfront24-Hfront14);
    Tmin=calculaTreg1(P,Hfront14);
    densmin=calculadens1(T,P);
    Tmax=calculaTreg2(P,Hfront24);
    densmax=calculadens2(T,P);
    densidade=1/(titulo*(1/densmax)+(1-titulo)*(1/densmin));
    //cout << "(<=16.529)T :" << T <<endl;
    //cout << "titulo :" << titulo <<endl;
    //cout << "densidade :" << densidade <<endl;
    }

    if (entalpia_esp>=Hfront24)
    {
```

```
        T=calculaTreg2(P,entalpia_esp);
        titulo=1;
        densidade=calculadens2(T,P);
        //cout << "(<=16.529)T :" << T <<endl;
        //cout << "titulo :" << titulo <<endl;
        //cout << "densidade :" << densidade <<endl;
        }
}


if (P>16.529164&&P<22.064)
{
double Hfront13;
Hfront13=calculaH13(P);
//cout << "(>16.529e<22.064)Hfront13 :" << Hfront13 <<endl;

double Hfront23;
Hfront23=calculaH23(P);
//cout << "(>16.529e<22.064)Hfront23 :" << Hfront23 <<endl;

double Hfront34n;
Hfront34n=calculaH34n(P);
//cout << "(>16.529e<22.064)Hfront34n :" << Hfront34n <<endl;

double Hfront34p;
Hfront34p=calculaH34p(P);
//cout << "(>16.529e<22.064)Hfront34p :" << Hfront34p <<endl;
```

```
if (entalpia_esp<=Hfront13)
    {
    T=calculaTreg1(P,entalpia_esp);
    titulo=0;
    densidade=calculadens1(T,P);
    //cout << "(>16.529e<22.064)T :" << T <<endl;
    //cout << "titulo :" << titulo <<endl;
    //cout << "densidade :" << densidade <<endl;
    }

if (entalpia_esp<=Hfront34n&&entalpia_esp>Hfront13)
    {
    T=calculaTreg3(P,entalpia_esp);
    titulo=0;
    densidade=dens3;
    //cout << "(>16.529e<22.064)T :" << T <<endl;
    //cout << "titulo :" << titulo <<endl;
    //cout << "densidade :" << densidade <<endl;
    }

if (entalpia_esp>Hfront34n&&entalpia_esp<Hfront34p)
    {
    T=calculaTreg4(P);
    Tmin=calculaTreg3(P,Hfront34n);
    densmin=dens3;
    Tmax=calculaTreg3(P,Hfront34p);
    densmax=dens3;
    titulo=(entalpia_esp-Hfront34n)/(Hfront34p-Hfront34n);
```

```
        densidade=1/(titulo*(1/densmax)+(1-titulo)*(1/densmin));
        //cout << "(<=16.529)T :" << T <<endl;
        //cout << "titulo :" << titulo <<endl;
        //cout << "densidade :" << densidade <<endl;
        }

   if (entalpia_esp>=Hfront34p&&entalpia_esp<Hfront23)
        {
        T=calculaTreg3(P,entalpia_esp);
        titulo=1;
        densidade=dens3;
        //cout << "(>16.529e<22.064)T :" << T <<endl;
        //cout << "titulo :" << titulo <<endl;
        //cout << "densidade :" << densidade <<endl;
        }

  if (entalpia_esp>=Hfront23)
        {
        T=calculaTreg2(P,entalpia_esp);
        titulo=1;
        densidade=calculadens2(T,P);
        //cout << "(>16.529e<22.064)T :" << T <<endl;
        //cout << "titulo :" << titulo <<endl;
        //cout << "densidade :" << densidade <<endl;
        }

}
```

```
if (P>=22.064)
{
double Hfront13;
Hfront13=calculaH13(P);
//cout << "(>=22.064)Hfront13 :" << Hfront13 <<endl;

double Hfront23;
Hfront23=calculaH23(P);
//cout << "(>=22.064)Hfront23 :" << Hfront23 <<endl;

if (entalpia_esp<=Hfront13)
    {
    T=calculaTreg1(P,entalpia_esp);
    titulo=-1;
    densidade=calculadens1(T,P);
    //cout << "(>=22.064)T :" << T <<endl;
    //cout << "titulo :" << titulo <<endl;
    //cout << "densidade :" << densidade <<endl;
    }

if (entalpia_esp>=Hfront23)
    {
    T=calculaTreg2(P,entalpia_esp);
    titulo=-1;
    densidade=calculadens2(T,P);
    //cout << "(>=22.064)T :" << T <<endl;
    //cout << "titulo :" << titulo <<endl;
    //cout << "densidade :" << densidade <<endl;
```

```
            }

        if (entalpia_esp<Hfront23&&&entalpia_esp>Hfront13)
            {
            T=calculaTreg3(P,entalpia_esp);
            titulo=-1;
            densidade=dens3;
            //cout << "(>=22.064)T :" << T <<endl;
            //cout << "titulo :" << titulo <<endl;
            //cout << "densidade :" << densidade <<endl;
            }
        }

        return densidade;
}
//---------------------------------------------------------------------------

int main(int argc, char* argv[])
{
        double deltat=0.3;   //s
        double deltax=0.5;   //m
        double Dtubo[4];  //m
        double Atubo[4];  //m3
        double Ptubulao[4];   //MPa
        double Ptublama[4];   //MPa
        double Htubulao[4]; //J/kg
        double Htublama[4];
        double Hchute[4];    //J/kg
```

```
double Pchute[4];
double vchute[4];
double perfilP[25][4];
double perfilH[25][4];
double perfilT[25][4];
double perfildens[25][4];
double perfiltitulo[25][4];
double perfilv[24][4];
double perfilvant[24][4];
double perfilvmed[24][4];
double P=0;
double entalpia_esp=0;
double densidade=0;
//BALANCO DE MOMENTO
double perfillzero[24][4];
double perfilke[24][4];
double perfilo[24][4];
double perfill[24][4];
double perfillleste[24][4];
double perfildensmed[24][4];
double perfildensmedant[24][4];
double Spc[24][4];
double Spp[24][4];
double dPr[24][4];
double fatrito=0;
double fatritoant=0;
double efat=100;
double Reynolds=0;
```

```
double rugos=0.000046;
double katrito[4];
//Inversão
int linhas;
int colunas;
// LINHAS E COLUNAS TEM QUE SER MENOR QUE 59
double Coef[60][60];
double Result[60];
double MatL[60][60];
double MatT[60][60];
double Matc[60];
double Sol[60];
//Correção da pressão
double perfilmo[25][4];
double perfilml[25][4];
double perfilkp[25][4];
double perfilOp[25][4];
double perfilLp[25][4];
double perfilCp[25][4];
double perfildensant[25][4];
double perfilPl[25][4];
//Balanço de energia
double perfilChzero[25][4];
double perfilkh[25][4];
double perfilOh[25][4];
double perfilLh[25][4];
double perfilCh[25][4];
double perfilSch[25][4];
```

```
double perfilSph[25][4];
double perfilHant[25][4];
double Q[4];
double Vtublama;

double tempo;
double vazaom0[50];
double vazaom1[50];
double vazaom2[50];
double vazaovap1[50];
double vazaovap2[50];
int it=0;

tempo=0;

katrito[0]=1;
katrito[1]=2.3;
katrito[2]=2.3;

Q[0]=0;
Q[1]=100000;
Q[2]=1000000;

Vtublama=0.015;

Dtubo[0]=0.0575;
Dtubo[1]=0.0448;
Dtubo[2]=0.0448;
```

```
Atubo[0]=0.002597;
Atubo[1]=0.001576;
Atubo[2]=0.001576;

Ptubulao[0]=7;
Ptubulao[1]=7;
Ptubulao[2]=7;

Htubulao[0]=1267437.2139;
Htubulao[1]=1267437.2139;
Htubulao[2]=1267437.2139;

Htublama[0]=1000132.8;
Htublama[1]=1000132.8;
Htublama[2]=1000132.8;

Hchute[0]=1267437.2139;
Hchute[1]=1267437.2139;
Hchute[2]=1267437.2139;

Pchute[0]=7;
Pchute[1]=7;
Pchute[2]=7;

vchute[0]=0.0001;
vchute[1]=0.0001;
vchute[2]=0.0001;
```

```
//CHUTE
    for (int iaa=0;iaa<3;iaa++)
    {
     perfilP[16][iaa]=Pchute[iaa];
     perfilP[0][iaa]=Ptubulao[iaa];

     for (int Pi=1;Pi<16;Pi++)
     {
     perfilP[Pi][iaa]=Pchute[iaa];
     }

     perfilH[0][iaa]=Htubulao[iaa];
     perfilHant[0][iaa]=perfilH[0][iaa];

     for (int Hi=1;Hi<17;Hi++)
     {
     perfilH[Hi][iaa]=Hchute[iaa];
     perfilHant[Hi][iaa]=perfilH[Hi][iaa];
     }

    for (int vi=0;vi<16;vi++)
    {
    perfilv[vi][iaa]=vchute[iaa];
    perfilvant[vi][iaa]=perfilv[vi][iaa];
    }

   //for (int Pi=0;Pi<17;Pi++)
```

```
        //cout << perfilv[Pi][iaa];


        for (int densi=0;densi<17;densi++)
        {
        perfildens[densi][iaa]=calculadens(perfilP[densi][iaa],perfilH[densi][iaa]);
        perfiltitulo[densi][iaa]=titulo;
        perfilT[densi][iaa]=T;
        perfildensant[densi][iaa]=perfildens[densi][iaa];
        }

        for (int idma=0;idma<16;idma++)
        {
        perfildensmedant[idma][iaa]=(perfildens[idma][iaa]+perfildens[idma+1][iaa])/2;
        }

        for (int iha=0;iha<17;iha++)
        {
        perfilHant[iha][iaa]=perfilH[iha][iaa];
        }

        for (int ikp=0;ikp<17;ikp++)
        {
        perfilkp[ikp][iaa]=30;
        }
    }
//FIM DO CHUTE
```

```
  while
(pow(pow(perfilkp[0][0],2),0.5)>0.1||pow(pow(perfilkp[1][0],2),0.5)>0.1||pow(pow(perf
ilkp[2][0],2),0.5)>0.1||pow(pow(perfilkp[3][0],2),0.5)>0.1||pow(pow(perfilkp[4][0],2),0.
5)>0.1||pow(pow(perfilkp[5][0],2),0.5)>0.1||pow(pow(perfilkp[6][0],2),0.5)>0.1||pow(po
w(perfilkp[7][0],2),0.5)>0.1||pow(pow(perfilkp[8][0],2),0.5)>0.1||pow(pow(perfilkp[9][
0],2),0.5)>0.1||pow(pow(perfilkp[10][0],2),0.5)>0.1||pow(pow(perfilkp[11][0],2),0.5)>0.
1||pow(pow(perfilkp[12][0],2),0.5)>0.1||pow(pow(perfilkp[13][0],2),0.5)>0.1||pow(pow(
perfilkp[14][0],2),0.5)>0.1||pow(pow(perfilkp[15][0],2),0.5)>0.1||pow(pow(perfilkp[16][
0],2),0.5)>0.1||
pow(pow(perfilkp[0][1],2),0.5)>0.1||pow(pow(perfilkp[1][1],2),0.5)>0.1||pow(pow(perfi
lkp[2][1],2),0.5)>0.1||pow(pow(perfilkp[3][1],2),0.5)>0.1||pow(pow(perfilkp[4][1],2),0.
5)>0.1||pow(pow(perfilkp[5][1],2),0.5)>0.1||pow(pow(perfilkp[6][1],2),0.5)>0.1||pow(po
w(perfilkp[7][1],2),0.5)>0.1||pow(pow(perfilkp[8][1],2),0.5)>0.1||pow(pow(perfilkp[9][
1],2),0.5)>0.1||pow(pow(perfilkp[10][1],2),0.5)>0.1||pow(pow(perfilkp[11][1],2),0.5)>0.
1||pow(pow(perfilkp[12][1],2),0.5)>0.1||pow(pow(perfilkp[13][1],2),0.5)>0.1||pow(pow(
perfilkp[14][1],2),0.5)>0.1||pow(pow(perfilkp[15][1],2),0.5)>0.1||
pow(pow(perfilkp[0][2],2),0.5)>0.1||pow(pow(perfilkp[1][2],2),0.5)>0.1||pow(pow(perfi
lkp[2][2],2),0.5)>0.1||pow(pow(perfilkp[3][2],2),0.5)>0.1||pow(pow(perfilkp[4][2],2),0.
5)>0.1||pow(pow(perfilkp[5][2],2),0.5)>0.1||pow(pow(perfilkp[6][2],2),0.5)>0.1||pow(po
w(perfilkp[7][2],2),0.5)>0.1||pow(pow(perfilkp[8][2],2),0.5)>0.1||pow(pow(perfilkp[9][
2],2),0.5)>0.1||pow(pow(perfilkp[10][2],2),0.5)>0.1||pow(pow(perfilkp[11][2],2),0.5)>0.
1||pow(pow(perfilkp[12][2],2),0.5)>0.1||pow(pow(perfilkp[13][2],2),0.5)>0.1||pow(pow(
perfilkp[14][2],2),0.5)>0.1||pow(pow(perfilkp[15][2],2),0.5)>0.1||tempo<59.7)

{
tempo=tempo+deltat;
//BALANCO DE MOMENTO
```

```
for (int iaa=0;iaa<3;iaa++)
{
 for (int idm=0;idm<16;idm++)
 {
 perfildensmed[idm][iaa]=(perfildens[idm][iaa]+perfildens[idm+1][iaa])/2;
 }

 for (int ivm=1;ivm<16;ivm++)
 {
 perfilvmed[ivm][iaa]=(perfilv[ivm-1][iaa]+perfilv[ivm][iaa])/2;
 }

 //tubulão-tubos
 if (perfilv[0][iaa]>=0)
 {

Reynolds=perfildensmed[0][iaa]*pow(pow(perfilv[0][iaa],2),0.5)*Dtubo[iaa]/0.000096;
 //cout << Reynolds << endl;

fatritoant=0.25*pow((log10(((rugos/Dtubo[iaa])/3.7)+(5.74/pow(Reynolds,0.9)))),-2);
 //cout << fatritoant << endl;
 efat=10;
 while (efat>0.001)
 {
  fatrito=pow((1/(-
2*log10(((rugos/Dtubo[iaa])/3.7)+2.51/(Reynolds*pow(fatritoant,0.5))))),2);
  efat=pow(pow(fatrito-fatritoant,2),0.5);
  fatritoant=fatrito;
```

```
            //cout << fatrito << endl;
            //cout << "efat  " << efat << endl;
          }
          fatrito=katrito[iaa]*fatrito;
          //cout << perfilv[0][iaa] << "  fatrito:  " << fatrito << endl;

dPr[0][iaa]=(fatrito)*(deltax/Dtubo[iaa])*(perfildensmed[0][iaa]/2)*pow(perfilv[0][iaa],
2)+1.5*pow(perfilv[0][iaa],2)*(perfildensmed[0][iaa]/2);
        }
        else
        {

Reynolds=perfildensmed[0][iaa]*pow(pow(perfilv[0][iaa],2),0.5)*Dtubo[iaa]/0.000096;
        fatritoant=0.25*pow((log10(((rugos/Dtubo[iaa])/3.7)+(5.74/pow(Reynolds,0.9)))),-
2);
        efat=10;
        while (efat>0.001)
         {
            fatrito=pow((1/(-
2*log10(((rugos/Dtubo[iaa])/3.7)+2.51/(Reynolds*pow(fatritoant,0.5))))),2);
            efat=pow(pow(fatrito-fatritoant,2),0.5);
            fatritoant=fatrito;
         }
        fatrito=katrito[iaa]*fatrito;
        //cout << perfilv[0][iaa] << "  fatrito:  " << fatrito << endl;

dPr[0][iaa]=(fatrito)*(deltax/Dtubo[iaa])*(perfildensmed[0][iaa]/2)*pow(perfilv[0][iaa],
2);
```

```
}
//cout << "dpr " << dPr[0][iaa] << endl;

//tubos-tubulão de lama
if (perfilv[15][iaa]<0)
{

Reynolds=perfildensmed[15][iaa]*pow(pow(perfilv[15][iaa],2),0.5)*Dtubo[iaa]/0.00009
6;
fatritoant=0.25*pow((log10(((rugos/Dtubo[iaa])/3.7)+(5.74/pow(Reynolds,0.9)))),-
2);
efat=10;
while (efat>0.001)
 {
   fatrito=pow((1/(-
2*log10(((rugos/Dtubo[iaa])/3.7)+2.51/(Reynolds*pow(fatritoant,0.5))))),2);
   efat=pow(pow(fatrito-fatritoant,2),0.5);
   fatritoant=fatrito;
 }
fatrito=katrito[iaa]*fatrito;
//cout << perfilv[15][iaa] << "  fatrito:  " << fatrito << endl;

dPr[15][iaa]=(fatrito)*(deltax/Dtubo[iaa])*(perfildensmed[15][iaa]/2)*pow(perfilv[15][i
aa],2)+1.5*pow(perfilv[15][iaa],2)*(perfildensmed[15][iaa]/2);
}
else
{
```

```
Reynolds=perfildensmed[15][iaa]*pow(pow(perfilv[15][iaa],2),0.5)*Dtubo[iaa]/0.00009
6;
      fatritoant=0.25*pow((log10(((rugos/Dtubo[iaa])/3.7)+(5.74/pow(Reynolds,0.9)))),-
2);
      efat=10;
      while (efat>0.001)
       {
          fatrito=pow((1/(-
2*log10(((rugos/Dtubo[iaa])/3.7)+2.51/(Reynolds*pow(fatritoant,0.5))))),2);
          efat=pow(pow(fatrito-fatritoant,2),0.5);
          fatritoant=fatrito;
       }
      fatrito=katrito[iaa]*fatrito;
      //cout << perfilv[15][iaa] << "  fatrito:  " << fatrito << endl;

dPr[15][iaa]=(fatrito)*(deltax/Dtubo[iaa])*(perfildensmed[15][iaa]/2)*pow(perfilv[15][i
aa],2);
      }
      for (int ipr=1;ipr<15;ipr++)
      {

Reynolds=perfildensmed[ipr][iaa]*pow(pow(perfilv[ipr][iaa],2),0.5)*Dtubo[iaa]/0.0000
96;
      fatritoant=0.25*pow((log10(((rugos/Dtubo[iaa])/3.7)+(5.74/pow(Reynolds,0.9)))),-
2);
      efat=10;
      while (efat>0.001)
```

```
        {
          fatrito=pow((1/(-
2*log10(((rugos/Dtubo[iaa])/3.7)+2.51/(Reynolds*pow(fatritoant,0.5)))))),2);
          efat=pow(pow(fatrito-fatritoant,2),0.5);
          fatritoant=fatrito;
        }
      fatrito=katrito[iaa]*fatrito;
      //cout << perfilv[ipr][iaa] << "  fatrito:  " << fatrito << endl;

dPr[ipr][iaa]=((fatrito)*(deltax/Dtubo[iaa])*(perfildensmed[ipr][iaa]/2)*pow(perfilv[ipr]
[iaa],2));//+(perfilv[ipr]*perfildensmed[ipr]*(perfilvmed[ipr+1]-perfilvmed[ipr]));
      //cout << "dpr " << dPr[ipr][iaa] << endl;
      }
      //cout << "dpr " << dPr[15][iaa] << endl;
      for (int iscp=0;iscp<16;iscp++)
      {
      Spp[iscp][iaa]=-
pow(pow(dPr[iscp][iaa],2),0.5)/pow(pow(perfilv[iscp][iaa],2),0.5);
      }

      for (int ispc=0;ispc<16;ispc++)
      {
      //if (perfilv[ispc]>0)
      Spc[ispc][iaa]=deltax*perfildens[ispc][iaa]*9.8;

      //if (perfilv[ispc]<0)
      //Spc[ispc]=-deltax*perfildensmed[ispc]*9.8;
```

```
        //if (perfilv[ispc]=0)
        //Spc[ispc]=0;

        //cout << "dph  " << Spc[ispc][iaa] << endl;
        }

        for (int ilz=0;ilz<16;ilz++)
        {
        perfillzero[ilz][iaa]=perfildensmedant[ilz][iaa]*Atubo[iaa]*deltax/deltat;

perfilke[ilz][iaa]=Spc[ilz][iaa]*Atubo[iaa]*deltax+perfillzero[ilz][iaa]*perfilvant[ilz][ia
a];
        }

        perfilo[0][iaa]=0;
        for (int io=1;io<16;io++)
          {
          if (perfilv[io-1][iaa]>0)
            perfilo[io][iaa]=perfildensmed[io-1][iaa]*perfilv[io-1][iaa]*Atubo[iaa];
          else
            perfilo[io][iaa]=0;
          //cout <<"perfilo  " << perfilo[io][iaa] << endl;
          }

      perfillleste[15][iaa]=0;
      for (int ill=0;ill<15;ill++)
        {
        if (perfilv[ill+1][iaa]<0)
```

```
            perfillleste[ill][iaa]=-perfildensmed[ill+1][iaa]*perfilv[ill+1][iaa]*Atubo[iaa];
          else
            perfillleste[ill][iaa]=0;
          //cout <<"perfillleste  " << perfillleste[ill][iaa] << endl;
          }

      for (int ili=0;ili<16;ili++)
      {
      perfill[ili][iaa]=perfilo[ili][iaa]+perfillleste[ili][iaa]+perfillzero[ili][iaa]-
Atubo[iaa]*deltax*Spp[ili][iaa];
      //cout << "perfill  " << perfill[ili][iaa] << endl;
      }
       //for (int Pi=0;Pi<17;Pi++)
       //{
       //cout << "  " << Pi;
       //cout << "  " << perfilv[Pi][iaa];
       //cout << "  " << -perfilo[Pi][iaa];
       //cout << endl;
       //cout << "  " << perfill[Pi][iaa];
       //cout << endl;
       //cout << "  " << -perfillleste[Pi][iaa];
       //cout << endl;
       //}
   }

//INVERSÃO DE MATRIZ
      linhas = 48;
      colunas = 48;
```

```
        // LINHAS E COLUNAS TEM QUE SER MENOR QUE 59

         for (int iii=0;iii<linhas;iii++)
        {
         for (int jjj=0;jjj<colunas;jjj++)
         {
           Coef[iii][jjj]=0;
           MatL[iii][jjj]=0;
           MatT[iii][jjj]=0;
         }

         Result[iii]=0;
         Matc[iii]=0;
         Sol[iii]=0;
        }
//PERFILL
        for (int iii=0;iii<16;iii++)
        {
            Coef[iii][iii]=perfill[iii][0];
        }

        for (int iii=16;iii<32;iii++)
        {
            Coef[iii][iii]=perfill[iii-16][1];
        }

        for (int iii=32;iii<48;iii++)
        {
```

```
            Coef[iii][iii]=perfill[iii-32][2];
        }
    //PERFILO
        for (int iii=1;iii<16;iii++)
        {
            Coef[iii][iii-1]=-perfilo[iii][0];
        }

        for (int iii=17;iii<32;iii++)
        {
            Coef[iii][iii-1]=-perfilo[iii-16][1];
        }

        for (int iii=33;iii<48;iii++)
        {
            Coef[iii][iii-1]=-perfilo[iii-32][2];
        }
    //PERFILLLESTE
        for (int iii=0;iii<15;iii++)
        {
            Coef[iii][iii+1]=-perfillleste[iii][0];
        }

        for (int iii=16;iii<31;iii++)
        {
            Coef[iii][iii+1]=-perfillleste[iii-16][1];
        }
```

```
        for (int iii=32;iii<47;iii++)
        {
            Coef[iii][iii+1]=-perfillleste[iii-32][2];
        }
 //RESULT
        for (int iii=0;iii<16;iii++)
        {
            Result[iii]=perfilke[iii][0]+(perfilP[iii][0]-
perfilP[iii+1][0])*1000000*Atubo[0];
        }

        for (int iii=16;iii<32;iii++)
        {
            Result[iii]=perfilke[iii-16][1]+(perfilP[iii-16][1]-perfilP[iii+1-
16][1])*1000000*Atubo[1];
        }

        for (int iii=32;iii<48;iii++)
        {
            Result[iii]=perfilke[iii-32][2]+(perfilP[iii-32][2]-perfilP[iii+1-
32][2])*1000000*Atubo[2];
        }

        //for (int iii=0;iii<linhas;iii++)
        //{
         //for (int jjj=0;jjj<colunas;jjj++)
          //cout << Coef[iii][jjj] << "  ";
```

```
  //cout << Result[iii];
  //cout << endl;
 //}

 //cout << endl;

 for (int lll=0;lll<colunas;lll++)
 {
  for (int kkk=0;kkk<linhas;kkk++)
  {
   if (lll==0)
   {
    MatL[kkk][lll]=Coef[kkk][lll];
    //cout << "L" << kkk << lll << ": " << MatL[kkk][lll] << endl;
   }

   if (kkk==0)
   {
    MatT[kkk][lll]=Coef[kkk][lll]/Coef[0][0];
    //cout << "T" << kkk << lll << ": " << MatT[kkk][lll] << endl;
   }

   if (kkk<lll&&lll!=0&&kkk!=0)
   {
    MatT[kkk][lll]=Coef[kkk][lll]/MatL[kkk][kkk];
    for (int qqq=0;qqq<kkk;qqq++)
     MatT[kkk][lll]=MatT[kkk][lll]-
MatL[kkk][qqq]*MatT[qqq][lll]/MatL[kkk][kkk];
```

```
        //cout << "T" << kkk << lll << ": " << MatT[kkk][lll] << endl;
      }


      if (kkk>=lll&&lll!=0)
      {
        MatL[kkk][lll]=Coef[kkk][lll];
        for (int ppp=0;ppp<lll;ppp++)
          MatL[kkk][lll]=MatL[kkk][lll]-MatL[kkk][ppp]*MatT[ppp][lll];
        //cout << "L" << kkk << lll << ": " << MatL[kkk][lll] << endl;
      }

    }
  }


  Matc[0]=Result[0]/MatL[0][0];
  //cout << "c0: " << Matc[0] << endl;


  for (int mmm=1;mmm<linhas;mmm++)
  {
    Matc[mmm]=Result[mmm]/MatL[mmm][mmm];
    for (int nnn=0;nnn<mmm;nnn++)
      Matc[mmm]=Matc[mmm]-
MatL[mmm][nnn]*Matc[nnn]/MatL[mmm][mmm];
    //cout << "c" << mmm << ": " << Matc[mmm] << endl;
  }
```

```
int hhh=linhas-1;
Sol[hhh]=Matc[hhh];
//cout << "x" << hhh << ": " << Sol[hhh] << endl;

for (int uuu=hhh-1;uuu>=0;uuu--)
{
   Sol[uuu]=Matc[uuu];
   for (int ggg=hhh;ggg>uuu;ggg--)
      Sol[uuu]=Sol[uuu]-MatT[uuu][ggg]*Sol[ggg];
   //cout << "x" << uuu << ": " << Sol[uuu] << endl;
}

for (int iii=0;iii<16;iii++)
{
   perfilvant[iii][0]=perfilv[iii][0];
   perfilv[iii][0]=Sol[iii];
   //cout << iii << " v1 " << perfilv[iii][0] << endl;
}
//cout << endl;

for (int iii=16;iii<32;iii++)
{
   perfilvant[iii-16][1]=perfilv[iii-16][1];
   perfilv[iii-16][1]=Sol[iii];
   //cout << iii-16 << " v1 " << perfilv[iii-16][1] << endl;
}
//cout << endl;
```

```
    for (int iii=32;iii<48;iii++)
    {
       perfilvant[iii-32][2]=perfilv[iii-32][2];
       perfilv[iii-32][2]=Sol[iii];
       //cout << iii-32 << " v1 " << perfilv[iii-32][2] << endl;
    }
    //cout << endl;

//FIM DA INVERSÃO DE MATRIZ

//CORREÇÃO DE PRESSÃO

   for (int iaa=0;iaa<3;iaa++)
    {
    perfilOp[0][iaa]=0;


    for (int iii=1;iii<17;iii++)
    {
        perfilmo[iii][iaa]=Atubo[iaa]/perfill[iii-1][iaa];
        perfilOp[iii][iaa]=perfildensmed[iii-1][iaa]*Atubo[iaa]*perfilmo[iii][iaa];
    //cout << "  " << perfilOp[iii][iaa];
    }
    //cout << endl;

    perfilLp[16][iaa]=0;//ok
```

```
perfilLp[0][iaa]=0; //CC
perfilml[0][iaa]=Atubo[iaa]/perfill[0][iaa];
for (int iii=1;iii<16;iii++)
{
    perfilml[iii][iaa]=Atubo[iaa]/perfill[iii][iaa];
    perfilLp[iii][iaa]=perfildensmed[iii][iaa]*Atubo[iaa]*perfilml[iii][iaa];
//cout << "  " << perfilLp[iii][iaa];
}

perfilCp[0][iaa]=1; //CC
perfilCp[16][0]=perfilOp[16][0]+perfilOp[16][1]+perfilOp[16][2];
for (int iii=1;iii<16;iii++)
{
    perfilCp[iii][iaa]=perfilOp[iii][iaa]+perfilLp[iii][iaa];
    //cout << "  " << perfilCp[iii][iaa];
}

perfilkp[0][iaa]=0; //CC

for (int iii=1;iii<16;iii++)
{
    perfilkp[iii][iaa]=((perfildensant[iii][iaa]-
perfildens[iii][iaa])*Atubo[iaa]*deltax/deltat)+perfildensmed[iii-1][iaa]*perfilv[iii-
1][iaa]*Atubo[iaa]-perfildensmed[iii][iaa]*perfilv[iii][iaa]*Atubo[iaa];
    //cout << "kp " << perfilkp[iii][iaa] << endl;
}
perfilkp[16][0]=((perfildensant[16][0]-
perfildens[16][0])*Vtublama/deltat)+perfildensmed[15][0]*perfilv[15][0]*Atubo[0]+per
```

```
fildensmed[15][1]*perfilv[15][1]*Atubo[1]+perfildensmed[15][2]*perfilv[15][2]*Atubo
[2];
      //cout << "kp " << perfilkp[16][0] << endl;
      }

 //INVERSÃO DE MATRIZ
      linhas = 49;
      colunas = 49;
      // LINHAS E COLUNAS TEM QUE SER MENOR QUE 59

      for (int iii=0;iii<linhas;iii++)
      {
       for (int jjj=0;jjj<colunas;jjj++)
       {
         Coef[iii][jjj]=0;
         MatL[iii][jjj]=0;
         MatT[iii][jjj]=0;
       }

       Result[iii]=0;
       Matc[iii]=0;
       Sol[iii]=0;
     }
//PERFILCP
     for (int iii=0;iii<17;iii++)
     {
        Coef[iii][iii]=perfilCp[iii][0];
     }
```

```
        for (int iii=17;iii<33;iii++)
        {
            Coef[iii][iii]=perfilCp[iii-17][1];
        }

        for (int iii=33;iii<49;iii++)
        {
            Coef[iii][iii]=perfilCp[iii-33][2];
        }
    //PERFILOP
        for (int iii=1;iii<17;iii++)
        {
            Coef[iii][iii-1]=-perfilOp[iii][0];
        }

        for (int iii=18;iii<33;iii++)
        {
            Coef[iii][iii-1]=-perfilOp[iii-17][1];
        }

        for (int iii=34;iii<49;iii++)
        {
            Coef[iii][iii-1]=-perfilOp[iii-33][2];
        }
        Coef[16][32]=-perfilOp[16][1];
        Coef[16][48]=-perfilOp[16][2];
    //PERFILLP
```

```
        for (int iii=0;iii<16;iii++)
        {
            Coef[iii][iii+1]=-perfilLp[iii][0];
        }

        for (int iii=17;iii<32;iii++)
        {
            Coef[iii][iii+1]=-perfilLp[iii-17][1];
        }

        for (int iii=33;iii<48;iii++)
        {
            Coef[iii][iii+1]=-perfilLp[iii-33][2];
        }

        Coef[32][16]=-perfilLp[15][1];
        Coef[48][16]=-perfilLp[15][2];
    //PERFILKP
        for (int iii=0;iii<17;iii++)
        {
            Result[iii]=perfilkp[iii][0];
        }

        for (int iii=17;iii<33;iii++)
        {
            Result[iii]=perfilkp[iii-17][1];
        }
```

```
for (int iii=33;iii<49;iii++)
{
    Result[iii]=perfilkp[iii-33][2];
}

//for (int iii=0;iii<linhas;iii++)
//{
 //for (int jjj=0;jjj<colunas;jjj++)
  //cout << Coef[iii][jjj] << "  ";

 //cout << Result[iii];
 //cout << endl;
//}

//cout << endl;

for (int lll=0;lll<colunas;lll++)
{
  for (int kkk=0;kkk<linhas;kkk++)
  {
   if (lll==0)
   {
     MatL[kkk][lll]=Coef[kkk][lll];
     //cout << "L" << kkk << lll << ": " << MatL[kkk][lll] << endl;
   }

  if (kkk==0)
  {
```

```
            MatT[kkk][lll]=Coef[kkk][lll]/Coef[0][0];
            //cout << "T" << kkk << lll << ": " << MatT[kkk][lll] << endl;
          }

          if (kkk<lll&&lll!=0&&kkk!=0)
          {
            MatT[kkk][lll]=Coef[kkk][lll]/MatL[kkk][kkk];
            for (int qqq=0;qqq<kkk;qqq++)
              MatT[kkk][lll]=MatT[kkk][lll]-
MatL[kkk][qqq]*MatT[qqq][lll]/MatL[kkk][kkk];
            //cout << "T" << kkk << lll << ": " << MatT[kkk][lll] << endl;
          }


          if (kkk>=lll&&lll!=0)
          {
            MatL[kkk][lll]=Coef[kkk][lll];
            for (int ppp=0;ppp<lll;ppp++)
              MatL[kkk][lll]=MatL[kkk][lll]-MatL[kkk][ppp]*MatT[ppp][lll];
            //cout << "L" << kkk << lll << ": " << MatL[kkk][lll] << endl;
          }

        }
      }


      Matc[0]=Result[0]/MatL[0][0];
      //cout << "c0: " << Matc[0] << endl;
```

```
for (int mmm=1;mmm<linhas;mmm++)
{
   Matc[mmm]=Result[mmm]/MatL[mmm][mmm];
   for (int nnn=0;nnn<mmm;nnn++)
      Matc[mmm]=Matc[mmm]-
MatL[mmm][nnn]*Matc[nnn]/MatL[mmm][mmm];
   //cout << "c" << mmm << ": " << Matc[mmm] << endl;
}

int hhh2=linhas-1;
Sol[hhh2]=Matc[hhh2];
//cout << "x" << hhh2 << ": " << Sol[hhh2] << endl;

for (int uuu=hhh2-1;uuu>=0;uuu--)
{
   Sol[uuu]=Matc[uuu];
   for (int ggg=hhh2;ggg>uuu;ggg--)
      Sol[uuu]=Sol[uuu]-MatT[uuu][ggg]*Sol[ggg];
   //cout << "x" << uuu << ": " << Sol[uuu] << endl;
}

for (int iii=0;iii<17;iii++)
{
   perfilPl[iii][0]=Sol[iii];
   //cout << iii << "  " << perfilPl[iii][0] << endl;
}
```

```
        //cout << endl;

        for (int iii=17;iii<33;iii++)
        {
           perfilPl[iii-17][1]=Sol[iii];
           //cout << iii-17 << "  " << perfilPl[iii-17][1] << endl;
        }
        //cout << endl;

        for (int iii=33;iii<49;iii++)
        {
           perfilPl[iii-33][2]=Sol[iii];
           //cout << iii-33 << "  " << perfilPl[iii-33][2] << endl;
        }
        //cout << endl;

        perfilPl[16][1]=perfilPl[16][0];
        perfilPl[16][2]=perfilPl[16][0];

//FIM DA INVERSÃO DE MATRIZ

//CORREÇÃO DE VELOCIDADE

      for (int iaa=0;iaa<3;iaa++)
      {
        for (int ippl=0;ippl<17;ippl++)
          {
          perfilP[ippl][iaa]=perfilP[ippl][iaa]+perfilPl[ippl][iaa]/1000000;
```

```
        cout << ippl << " P " << perfilP[ippl][iaa] << endl;
        }

      for (int idma2=0;idma2<16;idma2++)
       {

perfilv[idma2][iaa]=perfilv[idma2][iaa]+perfilml[idma2][iaa]*(perfilPl[idma2][iaa]-
perfilPl[idma2+1][iaa]);
       cout << idma2 << " V " << perfilv[idma2][iaa] << endl;
       }
    }
//BALANÇO DE ENERGIA

  for (int iaa=0;iaa<3;iaa++)
  {
    for (int iCz=0;iCz<17;iCz++)
      {
      if (Q[iaa]>0)
      perfilSch[iCz][iaa]=Q[iaa];
      else
      perfilSch[iCz][iaa]=0;

      if (Q[iaa]<0)
      perfilSph[iCz][iaa]=Q[iaa]/perfilH[iCz][iaa];
      else
      perfilSph[iCz][iaa]=0;

      perfilSch[0][iaa]=0;
```

```
        perfilSph[0][iaa]=0;
        perfilSch[16][iaa]=0;
        perfilSph[16][iaa]=0;

        perfilChzero[iCz][iaa]=perfildensant[iCz][iaa]*Atubo[iaa]*deltax/deltat;


perfilkh[iCz][iaa]=perfilSch[iCz][iaa]*Atubo[iaa]*deltax+perfilChzero[iCz][iaa]*perfil
Hant[iCz][iaa];
        }

      perfilOh[0][iaa]=0;
      for (int ioh=1;ioh<17;ioh++)
         {
         if (perfilv[ioh-1][iaa]>0)
         perfilOh[ioh][iaa]=perfildensmed[ioh-1][iaa]*perfilv[ioh-1][iaa]*Atubo[iaa];
         else
         perfilOh[ioh][iaa]=0;
         }

      perfilLh[16][iaa]=0;
      for (int ilh=0;ilh<16;ilh++)
         {
         if (perfilv[ilh][iaa]<0)
         perfilLh[ilh][iaa]=-perfildensmed[ilh][iaa]*perfilv[ilh][iaa]*Atubo[iaa];
         else
         perfilLh[ilh][iaa]=0;
         }
```

```
      for (int ich=0;ich<16;ich++)
          {
          perfilCh[ich][iaa]=perfilOh[ich][iaa]+perfilLh[ich][iaa]+perfilChzero[ich][iaa]-
perfilSph[ich][iaa]*Atubo[iaa]*deltax;
          }

perfilCh[16][0]=perfilOh[16][0]+perfilOh[16][1]+perfilOh[16][2]+perfilChzero[16][0];

      if (perfilv[0][iaa]>0)
          {
          perfilkh[0][iaa]=Htubulao[iaa];
          perfilLh[0][iaa]=0;
          perfilCh[0][iaa]=1;
          }


      for (int idma2=0;idma2<16;idma2++)
       {
       perfilHant[idma2][iaa]=perfilH[idma2][iaa];
       perfildensmedant[idma2][iaa]=perfildensmed[idma2][iaa];
       perfildensant[idma2][iaa]=perfildens[idma2][iaa];
       }
     }
//INVERSÃO DE MATRIZ
      linhas = 49;
      colunas = 49;
// LINHAS E COLUNAS TEM QUE SER MENOR QUE 59
```

```
        for (int iii=0;iii<linhas;iii++)
        {
         for (int jjj=0;jjj<colunas;jjj++)
         {
           Coef[iii][jjj]=0;
           MatL[iii][jjj]=0;
           MatT[iii][jjj]=0;
         }

         Result[iii]=0;
         Matc[iii]=0;
         Sol[iii]=0;
        }
//PERFILCH
      for (int iii=0;iii<17;iii++)
      {
          Coef[iii][iii]=perfilCh[iii][0];
      }

      for (int iii=17;iii<33;iii++)
      {
          Coef[iii][iii]=perfilCh[iii-17][1];
      }

      for (int iii=33;iii<49;iii++)
      {
          Coef[iii][iii]=perfilCh[iii-33][2];
```

```
    }
//PERFILOH
    for (int iii=1;iii<17;iii++)
    {
        Coef[iii][iii-1]=-perfilOh[iii][0];
    }

    for (int iii=18;iii<33;iii++)
    {
        Coef[iii][iii-1]=-perfilOh[iii-17][1];
    }

    for (int iii=34;iii<49;iii++)
    {
        Coef[iii][iii-1]=-perfilOh[iii-33][2];
    }

    Coef[16][32]=-perfilOh[16][1];
    Coef[16][48]=-perfilOh[16][2];

//PERFILLH
    for (int iii=0;iii<16;iii++)
    {
        Coef[iii][iii+1]=-perfilLh[iii][0];
    }

    for (int iii=17;iii<32;iii++)
    {
```

```
        Coef[iii][iii+1]=-perfilLh[iii-17][1];
    }

    for (int iii=33;iii<48;iii++)
    {
        Coef[iii][iii+1]=-perfilLh[iii-33][2];
    }

    Coef[32][16]=-perfilLh[15][1];
    Coef[48][16]=-perfilLh[15][2];

//PERFILKH
    for (int iii=0;iii<17;iii++)
    {
        Result[iii]=perfilkh[iii][0];
    }

    for (int iii=17;iii<33;iii++)
    {
        Result[iii]=perfilkh[iii-17][1];
    }

    for (int iii=33;iii<49;iii++)
    {
        Result[iii]=perfilkh[iii-33][2];
    }

    //for (int iii=0;iii<linhas;iii++)
```

```
//{
 //for (int jjj=0;jjj<colunas;jjj++)
  //cout << Coef[iii][jjj] << "  ";

 //cout << Result[iii];
 //cout << endl;
//}

//cout << endl;

for (int lll=0;lll<colunas;lll++)
{
  for (int kkk=0;kkk<linhas;kkk++)
  {
   if (lll==0)
   {
     MatL[kkk][lll]=Coef[kkk][lll];
     //cout << "L" << kkk << lll << ": " << MatL[kkk][lll] << endl;
   }

   if (kkk==0)
   {
     MatT[kkk][lll]=Coef[kkk][lll]/Coef[0][0];
     //cout << "T" << kkk << lll << ": " << MatT[kkk][lll] << endl;
   }

   if (kkk<lll&&lll!=0&&kkk!=0)
   {
```

```
            MatT[kkk][lll]=Coef[kkk][lll]/MatL[kkk][kkk];
            for (int qqq=0;qqq<kkk;qqq++)
              MatT[kkk][lll]=MatT[kkk][lll]-
MatL[kkk][qqq]*MatT[qqq][lll]/MatL[kkk][kkk];
            //cout << "T" << kkk << lll << ": " << MatT[kkk][lll] << endl;
          }


          if (kkk>=lll&&lll!=0)
          {
            MatL[kkk][lll]=Coef[kkk][lll];
            for (int ppp=0;ppp<lll;ppp++)
              MatL[kkk][lll]=MatL[kkk][lll]-MatL[kkk][ppp]*MatT[ppp][lll];
            //cout << "L" << kkk << lll << ": " << MatL[kkk][lll] << endl;
          }

        }
      }


      Matc[0]=Result[0]/MatL[0][0];
      //cout << "c0: " << Matc[0] << endl;


      for (int mmm=1;mmm<linhas;mmm++)
      {
        Matc[mmm]=Result[mmm]/MatL[mmm][mmm];
        for (int nnn=0;nnn<mmm;nnn++)
```

```
            Matc[mmm]=Matc[mmm]-
MatL[mmm][nnn]*Matc[nnn]/MatL[mmm][mmm];
        //cout << "c" << mmm << ": " << Matc[mmm] << endl;
    }

    int hhh3=linhas-1;
    Sol[hhh3]=Matc[hhh3];
    //cout << "x" << hhh3 << ": " << Sol[hhh3] << endl;

    for (int uuu=hhh3-1;uuu>=0;uuu--)
    {
      Sol[uuu]=Matc[uuu];
      for (int ggg=hhh3;ggg>uuu;ggg--)
        Sol[uuu]=Sol[uuu]-MatT[uuu][ggg]*Sol[ggg];
      //cout << "x" << uuu << ": " << Sol[uuu] << endl;
    }

    for (int iii=0;iii<17;iii++)
    {
      perfilH[iii][0]=Sol[iii];
      cout << iii << " H " << perfilH[iii][0] << endl;
    }

    for (int iii=17;iii<33;iii++)
    {
      perfilH[iii-17][1]=Sol[iii];
      cout << iii-17 << " H " << perfilH[iii-17][1] << endl;
    }
```

```
      for (int iii=33;iii<49;iii++)
      {
        perfilH[iii-33][2]=Sol[iii];
        cout << iii-33 << " H " << perfilH[iii-33][2] << endl;
      }

      perfilH[16][1]=perfilH[16][0];
      perfilH[16][2]=perfilH[16][0];
//FIM DA INVERSÃO DE MATRIZ

  for (int iaa=0;iaa<3;iaa++)
  {
   for (int densi=0;densi<17;densi++)
      {
      perfildens[densi][iaa]=calculadens(perfilP[densi][iaa],perfilH[densi][iaa]);
      perfiltitulo[densi][iaa]=titulo;
      perfilT[densi][iaa]=T;

      if
(((tempo>2.9)&&(tempo<3.1))||((tempo>5.9)&&(tempo<6.1))||((tempo>8.9)&&(tempo
<9.1))||((tempo>11.9)&&(tempo<12.1))||((tempo>14.9)&&(tempo<15.1))||((tempo>17.9
)&&(tempo<18.1))||(((tempo>20.9)&&(tempo<21.1)))||((tempo>23.9)&&(tempo<24.1))
||((tempo>26.9)&&(tempo<27.1))||((tempo>29.9)&&(tempo<30.1))||((tempo>32.9)&&(t
empo<33.1))||((tempo>35.9)&&(tempo<36.1))||((tempo>38.9)&&(tempo<39.1))||((temp
o>41.9)&&(tempo<42.1))||((tempo>44.9)&&(tempo<45.1))||((tempo>47.9)&&(tempo<
48.1))||((tempo>50.9)&&(tempo<51.1))||((tempo>53.9)&&(tempo<54.1))||((tempo>56.9
)&&(tempo<57.1))||((tempo>59.9)&&(tempo<60.1)))
```

```
        {
        cout << " dens " << perfildens[densi][iaa] << "    x " << perfiltitulo[densi][iaa]
<< "   T " << perfilT[densi][iaa] << endl;
        }
      }
  cout << endl;
  }

 cout << "t: " << tempo << " s" << endl;
 if
(((tempo>2.9)&&(tempo<3.1))||((tempo>5.9)&&(tempo<6.1))||((tempo>8.9)&&(tempo
<9.1))||((tempo>11.9)&&(tempo<12.1))||((tempo>14.9)&&(tempo<15.1))||((tempo>17.9
)&&(tempo<18.1))||(((tempo>20.9)&&(tempo<21.1)))||((tempo>23.9)&&(tempo<24.1))
||((tempo>26.9)&&(tempo<27.1))||((tempo>29.9)&&(tempo<30.1))||((tempo>32.9)&&(t
empo<33.1))||((tempo>35.9)&&(tempo<36.1))||((tempo>38.9)&&(tempo<39.1))||((temp
o>41.9)&&(tempo<42.1))||((tempo>44.9)&&(tempo<45.1))||((tempo>47.9)&&(tempo<
48.1))||((tempo>50.9)&&(tempo<51.1))||((tempo>53.9)&&(tempo<54.1))||((tempo>56.9
)&&(tempo<57.1))||((tempo>59.9)&&(tempo<60.1)))
   {
   it=tempo/3;
   vazaom0[it]=perfilv[15][0]*Atubo[0]*perfildensmed[15][0];
   vazaom1[it]=perfilv[15][1]*Atubo[1]*perfildensmed[15][1];
   vazaom2[it]=perfilv[15][2]*Atubo[2]*perfildensmed[15][2];
   vazaovap1[it]=perfilv[1][1]*Atubo[1]*perfildensmed[1][1]*perfiltitulo[1][1];
   vazaovap2[it]=perfilv[1][2]*Atubo[2]*perfildensmed[1][2]*perfiltitulo[1][2];
   cout << "ok" << endl;
   }
```

```
}
    for (int icout=0;icout<21;icout++)
    {
     cout << vazaom0[icout] << "   " << vazaom1[icout] << "   " << vazaom2[icout] << "
" << vazaovap1[icout] << "   " << vazaovap2[icout] << endl;
    }

      getch();
      return 0;
}
```