Report005: Modelo de Mistura Estacionário v3 (Codificação)

Fernando Enrique Castillo Vicencio
Federal University of Technology - Paraná

1 Estrutura do programa

O programa está dividido nos seguintes subprogramas:

- **mainDriftFluxModelSteady_v1.m**: Programa principal do Modelo, baseado no algoritmo 5.1 da Tese de Lima.
- **flowPatternIdentification_v1.m**: Programa para identificar o padrão do escoamento (disperso, separado ou intermitente)
- **AlphaTauDispersed_*v1.m***: Programa para calcular a fração de vazio e a força de atrito para o escoamento disperso.
- **AlphaTauSeparated_*v1.m***: Programa para calcular a fração de vazio e a força de atrito para o escoamento separado.
- **AlphaTau*Intermittent_v1.m***: Programa para calcular a fração de vazio e a força de atrito para o escoamento intermitente.
- ***filmThickSeparated.m***: Programa para calcular a espessura do filme de líquido no escoamento separado.
- ***liquidFractionDispersed.m***: Programa para calcular a fração de líquido no escoamento disperso.
- ***bisectionMethod.m***: Programa para resolver uma equação implícita pelo método da bisseção.

2 Programa Principal (*mainDriftFluxModelSteady_v1.m*):

2.1 Dados de entrada:

- Propriedades da tubulação:
  - Comprimento (L), diâmetro (D), ângulo (θ), rugosidade (ε)

```
% Pipe properties
L = 10.; %length
D = 0.0254; %diameter
angle = 15.;  % angle in degrees
theta = angle * pi / 180.; % angle in radians
rug = 0.001; %rugosity
```

- Dados Termodinâmicos:
  - Temperatura dos fluidos (T_0), aceleração da gravidade (g) , constante do gás (R_0)

```
% Other data
To = 300.; % temperature
g = 9.81; % acceleration of gravity
Ro = 287.; % constant of the gas (air)
```

2.2 Primeiros cálculos:

- Cálculos iniciais das iterações:
  - Número de intervalos (N), variação da posição (Δz), tolerância das iterações

```
% Iteration Calculations
dz0 = 40*D; % approximated delta z, n times diameter
N = floor(L / dz0 + 1); % number of intervales (integer)
dz = L / (N-1) ; % exact delta z
tol = 1e-4; % error tolerance of calculations
```

- Primeiros cálculos geométricos:
  - Perímetro da tubulação: $S = \pi D$, área da tubulação: $A = \frac{\pi D^2}{4}$

```
% First Geometrical calculations
S = pi * D; % perimeter of the pipe
A = pi / 4 * (D^2); % area of the pipe
```

- Propriedades do escoamento

  - Velocidade superficial do líquido (J_L), massa específica do líquido (ρ_L), viscosidade do gás (μ_G), viscosidade do líquido (μ_L), tensão interfacial líquido-gás (γ)

```
% Flow properties
JL = 0.5; % superficial velocity of liquid
DenL = 1000.;  % density of liquid
VisG = 1e-5; % viscosity of gas
VisL = 1e-3;  % viscosity of liquid
surTen = 0.7; % surface tension of liquid in contact with gas
```

2.3 Cálculos na seção N (seção de saída da tubulação)

- Dados na saída:

  - Velocidade superficial do gás, $J_G\,(N)$; pressão na saída, $P\,(N)$:

```
% Outlet properties
JG(N) = 0.5; % Superficial velocity of gas
P(N) = 2e5; % Pressure at outlet
```

- Cálculos do escoamento
  - Velocidade superficial da mistura: $J(N) = J_L + J_G(N)$, massa específica do gás: $\rho_G(N) = \frac{P(N)}{R_0 T_0}$, diferença de massas específicas: $\Delta\rho(N) = \rho_L - \rho_G(N)$
  - Fluxo mássico do gás: $G_G = \rho_G(N)\, J_G(N)$, fluxo mássico do líquido: $G_L = \rho_L J_L$

```
% Second Calculations
J(N)  = JL + JG(N); % Superficial velocity of the mixture
DenG(N) = P(N) / (Ro * To); % Density of the ideal gas
dRho = DenL-DenG(N); % difference of density

GG = DenG(N) * JG (N); % Mass flux of gas
GL = DenL * JL; % Mass flux of liquid
```

- Rotina para a determinação do padrão do escoamento (*flowPatternIdentification_v1.m*)

```
% Routine flowPattern
% Algorithm 5.2
pattern=flowPattern(JG(N), JL, J(N) , DenG(N) , DenL, VisL, VisG, ...
        surTen, D, L, S, A, theta, rug, dRho, g);
```

- Rotina para a determinação da fração de vazio $\alpha(N)$ e a força de atrito $T_W(N)$ na seção de saída do escoamento (AlphaTauDispersed_*v1.m* ; AlphaTauSeparated_*v1.m* ; AlphaTau*Intermittent*_*v1.m*)

```
% Routines for VoidFraction and FrictionForce for each flow pattern
if pattern == 1 % Dispersed Flow
        %% algorithm 5.3
        [alpha(N), TW(N)] = AlphaTauDispersed(JG(N), J(N), ...
                DenG(N), DenL, VisG, VisL, surTen, D, S, theta) ;
elseif pattern == 2 % Separated Flow
        %% algorithm 5.4
        [alpha(N), TW(N)] = AlphaTauSeparated(JG(N), J(N), ...
                DenG(N), DenL, VisG, VisL, surTen, D, S, theta) ;
else % Intermittent Flow
        %% algorithm 5.5
        [alpha(N), TW(N)] = AlphaTauIntermittent(JG(N), J(N), ...
                DenG(N), DenL, VisG, VisL, surTen, D, S, theta) ;
end
```

- Velocidade das fases: $U_G(N) = \frac{J_G(N)}{\alpha(N)}$ e $U_L(N) = \frac{J_L}{1-\alpha(N)}$

```
% actual phase velocities (eq. 3.3 & 3.5) - Line11
UG(N) = JG(N) / alfa(N) ; % actual velocity of gas
UL(N) = JL / ( 1-alfa(N) ) ; % velocity of liquid
```

- Cálculo da função: $\Psi(N) = P(N) + G_G U_G(N) + G_L U_L(N)$

```
% Psi function - Line 12
Psi(N) = P(N) + GG * UG(N) + GL * UL(N);
```

- Massa específica da mistura: $\rho(N) = \alpha(N)\rho_G(N) + [1 - \alpha(N)]\rho_L$
- Função: $\frac{d\Psi}{dz}(N) = -T_W(N) - \rho(N)\ g \sin(\theta)$

```
%Density of the mixture & dPsidz - Line 13
Den(N) = alfa(N) * DenG(N) + ( 1-alfa(N) ) * DenL;
dPsidz(N) = -TW(N) - Den(N) * g * sin(theta);
```

2.4 Cálculos na seção z (desde *z=N-1* até *z=1*)

- Valores prévios das iterações

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculations for i from N-1 to 1
i = N ; % index of iteration
L(i) = (i-1)*dz ; % position at i=N

while i > 0 %Line 14
        i = i-1 ; % refresh index of iteration
        L(i) = (i-1)*dz ; % position at i < N
```

2.4.1 Resolver a EDO para a grandeza $\Psi(z)$ (método de Runge-Kutta)

- EDO : $\frac{d\Psi}{dz}(z) = -T_W(z+1) - \rho(z+1)\ g \sin(\theta) = \frac{d\Psi}{dz}(z+1)$.
  - O termo da direita foi calculado em $i+1$
- Condição de Contorno : $\Psi(L_{z+1}) = P(z+1) + G_G U_G(z+1) + G_L U_L(z+1)$
- Solução pelo Método de Runge-Kutta: $\Psi(z)$

```
%%%%%%%%%%%%%%%%%%%%%%%%
% Runge-Kutta method to solve EDO - Line 16
syms fun(z) % declare symbolic variable
% Define ODE
ode = diff(fun,z) == dPsidz(i+1);
% Boundary condition at z(i+1)
cond = fun( L(i+1) ) == P(z+1) + GG * UG(i+1) + GL * UL(i+1) ;
funSol(z) = dsolve(ode,cond) ; % solve ODE

% Calculate the value of Psi(i)
Psi(i) = vpa(feval(funSol,L(i)));
clear fun funSol cond ode % clear variables
%%%%%%%%%%%%%%%%%%%%%%%%
```

2.4.2 Loop para calcular a pressão na seção z

- O loop é calculado enquanto a condição de tolerância não se cumpre.

1. Massa específica do gás na seção z: $\rho_G(z) = \frac{P(z)^{i+1}}{R_0 T_0}$

```
Pold = P(i+1)    ;
%%%%%%%%%%%%%%%%%%%%%%%%
while fP > tol
        % Density of gas
        DenG(i) = Pold / (Ro*To) ;
```

2. Velocidade superficial do gás: $J_G(z) = \frac{G_G}{\rho_G(z)}$

```
        % Superficial velocity of the gas
        JG(i) = GG / DenG(i) ;
```

3. Velocidade superficial da mistura: $J(z) = J_L + J_G(z)$

```
        % Superficial velocity of the mixture
        J(i) = JL + JG(i) ;
```

4. Algoritmo para calcular a fração de vazio $\alpha(z)$ dependendo do padrão do escoamento: **AlphaTauDispersed_*v1.m*** , **AlphaTauSeparated_*v1.m*** , **AlphaTau*Intermittent*_*v1.m***

```
% Routines for VoidFraction for each flow pattern
if pattern == 1 % Dispersed Flow
        %% algorithm 5.3
        [alpha(i), del] = AlphaTauDispersed...
                (JG(i), J(i), DenG(i), DenL, VisG, ...
                VisL, surTen, D, S, theta) ;
elseif pattern == 2 % Separated Flow
        %% algorithm 5.4
        [alpha(i), del] = AlphaTauSeparated...
                (JG(i), J(i), DenG(i), DenL, VisG, ...
                VisL, surTen, D, S, theta) ;
else % Intermittent Flow
        %% algorithm 5.5
        [alpha(i), del] = AlphaTauIntermittent...
                (JG(i), J(i), DenG(i), DenL, VisG, ...
                VisL, surTen, D, S, theta) ;
end
clear del
% Return alpha(i)
```

5. Velocidade do gás: $U_G(z) = \frac{J_G(N)}{\alpha(z)} \frac{P(N)}{P(z)^{i+1}}$

```
% actual velocity of gas
UG(i) = JG(N) / alfa(i) * P(N) / Pold ;
```

6. Velocidade do líquido: $U_L(z) = \frac{J_L}{1-\alpha(z)}$

```
% actual velocity of liquid
UL(i) = JL / ( 1-alfa(i) ) ;
```

7. Cálculo da função $f(P) = P(z) + G_G U_G(z) + G_L U_L(z) - \Psi(z) >$ tolerância

```
% Calculate the function fP=0
fP = P(i) + GG * UG(i) + GL * UL(i) - Psi(i) ;

% Reassign values of pressure
Pold = P(i) ;

end
% Return P(i); DenG(i), JG(i), J(i), alfa(i), UG(i), UL(i)
%%%%%%%%%%%%%%%%%%%%%%%%
```

8. Algoritmo para calcular a força de atrito $T_W(z)$ dependendo do padrão do escoamento: **AlphaTauDispersed_*v1.m*** , **AlphaTauSeparated_*v1.m*** , **AlphaTau*Intermittent*_*v1.m***

```
% Routines for FrictionFactor for each flow pattern
if pattern == 1 % Dispersed Flow
        %% algorithm 5.3
        [del, TW(i)] = AlphaTauDispersed...
                (JG(i), J(i), DenG(i), DenL, VisG, ...
                VisL, surTen, D, S, theta);
elseif pattern == 2 % Separated Flow
                %% algorithm 5.4
        [del, TW(i)] = AlphaTauSeparated...
                (JG(i), J(i), DenG(i), DenL, VisG, ...
                VisL, surTen, D, S, theta);
else % Intermittent Flow
        %% algorithm 5.5
        [del, TW(i)] = AlphaTauIntermittent...
                (JG(i), J(i), DenG(i), DenL, VisG, ...
                VisL, surTen, D, S, theta);
end
clear del
% Return TW(i)
```

9. Massa específica da mistura: $\rho(z) = \alpha(z)\,\rho_G(z) + [1 - \alpha(z)]\,\rho_L$

```
        % Density of the mixture
        Den(i) = alfa(i) * DenG(i) + ( 1-alfa(i) ) * DenL;
```

10. Cálculo da função $\frac{d\Psi}{dz}(z) = -T_W(z) - \rho(z)\; g \sin(\theta)$

```
        % Calculate dPsidz(i)
        dPsidz(i) = -TW(i) - Den(i) * g * sin(theta);
```

11. Algoritmo para a determinação do padrão do escoamento (***flowPatternIdentification_v1.m***)

```
        %Rotina flowPattern
        pattern=flowPattern...
                (JG(i), JL, J(i) , DenG(i) , DenL, VisL, VisG, ...
                surTen, D, L, S, A, theta, rug, dRho, g) ;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

2.5 Cálculo da queda de pressão ao longo da tubulação

```
%Calculate Pressure Loss through the Pipe
dPL = ( P(1) - P(N) ) / L ;
```

3 Identificação do Padrão de Escoamento (flowPatternIdentification_*v1.m*):

1. Definição da Função:

```
% Flow Pattern Identification - Algorithm 5.2
function pattern=patternIndex(JG, JL, J , DenG , DenL, ...
        VisL, VisG, surTen, D, L, S, A, theta, rug , dRho, g)
```

2. Escoamento quase-horizontal: $0^\circ \leq \theta \leq 10^\circ$ (modelo de Taitel e Dukler, 1976)

```
if 0 <= theta && theta <= 10/180*pi
        % Modelo of Taitel & Dukler (1976)
```

   (a) C: core (gás) , F: film (líquido)

```
            % line2
            ED = 0. ;
            RD = 0. ;
            DenC = DenG ;
            DenF = DenL ;
            VisC = VisG ;
            VisF = VisL ;
```

(b) Algoritmo para calcular a espessura do filme no escoamento separado (***filmThickSeparated.m***)

```
% algoritmo 4.2 - Line3
[alpha,RF,SF,SC,tauWF,tauWC,SI] = ...
        filmThickSeparated(JG, JL, J, DenC, DenF, ...
        VisC, VisF, D, A, theta, rug, surTen);
```

(c) Determinação do padrão do escoamento (disperso, separado ou intermitente)

- Será necessário o uso do algoritmo para determinar a fração de líquido no escoamento disperso (*liquidFractionDispersed.m*).

```
if JG < (1-delta) * ( (1-RF)^3 * A * ...
        dRho * g * cos(theta) / (DenG*SI) )^(1/2)
                pattern =2 % ESTRATIFICADO!
                %%%%%%%%%%%%%%%%%%%%
                %PODE SER ESTRATIFICADO LISO OU ONDULADO

elseif delta <= 0.35
        pattern =2 % ANULAR!
else
        %DETERMINAR CoB, VinfB e RS - ALGORITMO 4.1
        [CoB,VinfB,RS] = liquidFractionDispersed...
                (JG, J, DenL, DenG, VisL, VisG, ...
                surTen, D, theta, dRho, g)
```

```
                Co=CoB;
                VGJ = VinfB;

                alfa = 1-RS ;

                % DenS e VisS 4.4 4.5
                DenS = (1-RS)*DenG + RS*DenL;
                VisS = (1-RS)*VisG + RS*VisL;

                % ReS 5.31, CfS 5.30
                ReS = DenS * J * D / VisS ;
                CfS = ( -3.6 * log ( (rug/(3.7*D))^1.11 + ...
                        6.9/ReS ) )^-2 ;

                if JL >= 2 * RS * ( (1-RS)*A*dRho*g*cos(theta) / ...
                        DenL*Sl*CfS) ) ^(1/2)
                        pattern = 1 % BOLHAS
                else
                        pattern = 3 % GOLFADAS
                end
        end
```

3. Escoamento não-horizontal: $10^\circ < \theta \leq 90^\circ$ (modelo de Barnea et al., 1985)

```
else
        % Modelo de Barnea et al. (1985)
```

(a) Número de Eotvos:

```
        Eo =  dRho * g * D^2 / surTen ;
```

(b) Método iterativo para determinar $J_{L,\text{crítico}}$:

```
        JLcrit = bisectionMethod(@(x)( ...
                ( JG/(JG+x) ) - ...
                ( ( 0.402 * ( (JG+x)/...
                (dRho*g*D/DenL)^(1/2) )^(6/5)  )* Eo^0.1 * ...
                ( -3.6 * log( (rug/(3.7*D))^1.11 + ...
                ( 6.9*VisL./( DenL*(JG+x)*D ) ) ) )^(-4/5) - ...
                - 0.175 )^2 ), 0, 10*JL) ;
```

(c) Determinação do padrão do escoamento (disperso, separado ou intermitente)

```
        if JG >= 3.1 * (dRho*surTen*g*sin(theta))^(1/4) / ...
                (DenG)^(1/2)
                pattern = 2 ; % SEPARADO
        elseif JG/J > 0.52 && JL >= 0.48*JG/0.52
                pattern = 1; % DISPERSO
        elseif JG/J <= 0.52 && JL >= JLcrit
                pattern = 1; % DISPERSO : bolhas uniformes
        elseif theta > 60/180*pi  && JL >= 3*JG-...
                1.15*(dRho*surTen*g/DenL^2)^(1/4)*sin(theta)
                pattern = 1; % DISPERSO : bolhas distorcidas
        elseif JLcrit < JL && JL < 0.48*JG/0.52
                pattern = 2 ; % SEPARADO : Semianular
        elseif theta > 70/180*pi && JL >= sqrt(g*D) * ...
                ((L/D)/40.6-0.22) - JG
                pattern = 3; % INTERMITENTE : golfadas instaveis
        else
                pattern = 3; % INTERMITENTE : golfadas
        end
end
```

4 Força de Atrito e Fração de vazio no escoamento disperso (AlphaTauDispersed_*v1.m)*

- Será necessárias a função para calcular a fração de líquido no escoamento disperso (*liquidFractionDispersed.m*).
- Definição da função

```
[CoB,VinfB,RS] = liquidFractionDispersed...
        (JG, J , DenL, DenG, VisL, VisG, surTen, D, theta, dRho, g);

Co = CoB;
VGJ = VinfB;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
alfa = 1-RS % CONFERIR

DenS = (1-RS)*DenG + RS * DenL;
VisS = (1-RS)*VisG + RS * VisL;

ReS = DenS * J * D / VisS ;
DS = D;
CfS = ( -3.6 * log( (rug / (3.7*DS)).^1.11 + 6.9/ReS )) .^(-2) ;
tauWS =  CfS * DenS * J * abs(J) / 2 ;
TWS =  tauWS * S / A ;
```

```
beta = 0;
TW = TWS;
```

5 Força de Atrito e Fração de vazio no escoamento separado (AlphaTauSeparated_*v1.m)*

- Serão necessárias as funções para calcular a espessura do filme no escoamento separado (*filmThickSeparated.m*) e a fração de líquido no escoamento disperso (*liquidFractionDispersed.m*).

- Definição da função

```
function [alpha, TW] = AlphaTauSeparated...
        (JG, JL, J , DenG, DenL, VisG, VisL, ...
        surTen, D, A, theta,rug)
```

1. Desenvolvimento da função

```
% Wallis (Tabela4.3)
ED = 1- exp ( -0.125 * ( 1e4*JG*VisG* ...
        (DenG/DenL).^(1/2) / surTen - 1.5 ) );

RD = ED * JL / (JG + ED*JL);
```

```
DenF = DenL; VisF = VisL;
DenC = (1-RD)*DenG + RD * DenL;

VisC = (1-RD)*VisG + RD * VisL;
[HF,RF,SI] = filmThickSeparated(JG, JL, J , ...
        DenC , DenF , VisC , VisF, D, A , theta, rug , surTen)

[CoB,VinfB,RS] = liquidFractionDispersed ...
        (JG, J , DenL, DenG, VisL, VisG, surTen, D, theta, dRho, g);

Co = CoB; VGJ = VinfB;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
alfa = 1-RS % CONFERIR

DenS = (1-RS)*DenG + RS * DenL;
VisS = (1-RS)*VisG + RS * VisL;

ReS = DenS * J * D / VisS ;
DS = D;
CfS = ( -3.6 * log( (rug / (3.7*DS)).^1.11 + 6.9/ReS )) .^(-2) ;
tauWS =  CfS * DenS * J * abs(J) / 2 ;
TWS =  tauWS * S / A ;

beta = 0; TW = TWS;
```

6 Força de Atrito e Fração de vazio no escoamento intermitente (AlphaTau*Intermittent_v1.m)*

- Serão necessárias as funções para calcular a espessura do filme no escoamento separado (*filmThickSeparated.m*) e a fração de líquido no escoamento disperso (*liquidFractionDispersed.m*).

- Definição da função

```
function [alpha, TW] = AlphaTauIntermittent...
        (JG, JL, J , DenG, DenL, VisG, VisL, ...
        surTen, D, A, theta,rug)
```

1. Desenvolvimento da função

```
DenC = DenG ;
DenF = DenL ;
VisC = VisG ;
VisF = VisL ;

% Frequency, Gregory & Scott (1969)
f = 0.026 * (JL * (19.75/J + J) * (g*D) )^(6/5) ;

% Liquid fraction in the slug
% Gregory et al (1978)
RS = ( 1 + (J/8.66)^1.39 ) ^(-1) ;
```

```
UT =  1.2 + 0.35 sqrt(g*D) * sin(theta);

CoB = 1 + 0.2 * (sin(theta))^2 ;
VinfB = 1.54 ;
UB = CoB * J + VinfB ;

Co = CoT ;
VGJ = VinfT ;
alpha =  JG / ( Co*J+VGJ ) ;

if z = L
        %%%
        LF = zF;
        LS = UT / f - LF ; %4.27
        TWC = tauWC * SC / A ; % 5.28
        TWS = tauWS * SS / A ;
end

DenS = (1-RS)*DenG + RS * DenL;
VisS = (1-RS)*VisG + RS * VisL;

ReS = DenS * J * D / VisS ;
DS = D;
```

```
CfS = ( -3.6 * log( (rug / (3.7*DS)).^1.11 + 6.9/ReS )) .^(-2) ;
tauWS =  CfS * DenS * J * abs(J) / 2 ;
TWS =  tauWS * S / A ;

%beta <1;
TW = TWS;
```

7 Espessura do filme de líquido no escoamento separado (*filmThickSeparated.m*)

- Definição da função

```
function [alpha,RF,SF,SC,tauWF,tauWC,SI] = ...
        filmThickSeparated(JG, JL, J , DenC , DenF , VisC , VisF, ...
        D, A , theta , rug , surTen)
```

1. Desenvolvimento da função

```
model = 1 ;
tol = 1e-4 ;
interface = 1; %1:planar  2:concentric %%%%como?

dRho = DenF-DenC;
g = 9.81;

f1 = 1;
%line1

if model == 1 % model based on momentum model
        delta = JG / J /4
```

```
        while f1 > tol

                if interface == 1
                        lambda = 2*acos(1-2*delta);
                        SC = D * (pi - lambda/2.);
                        SF = D * lambda/2;
                        SI = D * sin (lambda/2) ;
                        DC = D * ( 1 + ( sin(lambda/2) - sin(lambda)/2 )
                            / ( 2 * (2*pi-lambda+sin(lambda)) ) )^-1;
                        DF = D * (1-sin(lambda)/lambda);
                        RF = (lambda-sin(lambda)) / (2*pi) ;
                else
                        SC = 0.;
                        SF = pi * D;
                        SI = pi * D * ( 1 - 2*delta);
                        DC = D * ( 1 - 2*delta);
                        DF = D * ( 4 * delta * (1-delta)  );
                        RF = 4 * delta * (1-delta);
                end

                % Wallis (Tabela4.3)
                ED = 1- exp ( -0.125 * ( 1e4*JG*VisC* (DenC/DenF).^(1/2)
                    / surTen - 1.5 ) );
                RD = ED * JL / (JG + ED*JL) ;
```

```
% Line 5
UC = ( JG + ED * JL ) / (1-RF) ;
UF = ( JL - ED * JL ) / RF ;

%Line 6
ReC = DenC * UC * DC / VisC ;
ReF = DenF * UF * DF / VisF ;

%Line 7
CfC = ( -3.6 * log( (rug / (3.7*DC )).^1.11 + 6.9/ReC ))
    .^(-2) ;
CfF = ( -3.6 * log( (rug / (3.7*DF )).^1.11 + 6.9/ReF ))
    .^(-2) ;

if interface == 2
        CfI = CfC * (1+300*delta);
elseif JG<=15
        CfI = 0.014;
else
        CfI = 0.0625 * ( log( 15/ReC + 2.3*delta/3.715 )
            ).^-2;
end
```

```
%Line 8
tauWC = CfC * DenC * UC * abs(UC) / 2.;
tauWF = CfF * DenF * UF * abs(UF) / 2.;
taul = Cfl * DenC * (UC-UF) * abs(UC-UF) / 2

if interface == 1

        lambda = bisectionMethod(@(x)( ...
                ( tauWF * (D .* x./2) ./ A ) - ...
                ( ( ((x-sin(x))./(2*pi)) / (1-(x-sin(x))
                   ./(2*pi)) ) .* tauWC * ( D * (pi - x
                   ./2) ) ./ A ) - ...
                ( 1 ./ ( 1 - (x-sin(x))./(2*pi) ) * (
                   taul .* ( D * sin(x./2) ) ./ A) ) +
                   ...
                ( ((x-sin(x))./(2*pi)) * dRho * g * sin(
                   theta) )...
                ), 0.01 , .49 ) ;

        delta = 1/2 * ( 1 - cos(lambda/2) );
        HF = delta * D ;
        SC = D * (pi - lambda/2.);
        SF = D * lambda/2;
        SI = D * sin (lambda/2) ;
```

```
                    RF = 4* (lambda-sin(lambda)) / (2*pi) ;
                    % Return [alpha,RF,SF,SC,tauWF,tauWC,SI]

                else % interface == 2

                    delta = bisectionMethod(@(x)( ...
                        ( tauWF * (pi*D) / A ) - ...
                        ( ( 1 ./ (1-4*x.*(1-x) )) * taul .* ( pi
                            *D*(1-2*x) ) / A ) + ...
                        ( 4*x.*(1-x).*dRho*g*sin(theta) )...
                        ), .05 , .45) ;

                    HF = delta * D
                    SI = pi * D * ( 1 - 2*delta)
                    RF = 4 * delta * (1-delta)

                end
                alfa = (1-RD)*(1-RF)

        end

elseif model == 2 % model based on kinematic law of drift
        Co = 0;          VGJ = 0 ;
end
```

8 Fração de líquido no esocamento disperso (*liquidFractionDispersed.m*)

- Definição da função

```
function [CoB,VinfB,RS] = liquidFractionDispersed...
        (JG, J , DenL, DenG, VisL, VisG, ...
        surTen, D, theta, dRho, g)
```

1. Desenvolvimento da função

```
Eo = dRho * g * D^2 / surTen;
Fr = J / sqrt(dRho*g*D*DenL)^(1/2);

if 0 <= theta && theta <= 10/180*pi
        DBcrit = 0.375*D*CfS* Fr^2 / cos(theta);
else
        DBcrit = 1.265*D * Eo^(-1/2);
end

CoB = 1 + 0.2 * (1-sqrt(DenG/DenL)) * (sin(theta))^2;

% agitado - tabela 4.1
CinfB = sqrt(2) * Eo^(-1/4) * sin(theta); %hipotese inicial
```

```
VinfB = CinfB * ( dRho*g*D/DenL )^(1/2);

RS = bisectionMethod(@(x)( 1-x-JG ./ ( CoB*J + ( sqrt(2)*x.^(7/4)* Eo
    ^(-1/4) * sin(theta) ) ) )...
        , .01 , 0.99)
CinfB = sqrt(2) * RS^(7/4) * Eo^(-1/4) * sin(theta);
VinfB = CinfB* (dRho*g*D/DenL)^(1/2);
```

9 Método da Bisseção (*bisectionMethod.m*)

- Definição da função

```
if f(a)*f(b)>0
        disp('Wrong choice')
        p=0;
else
        p = (a + b)/2;
        err = abs(f(p));
        while err > 1e-7
                if f(a)*f(p)<0
                        b = p;
                else
                        a = p;
                end

                p = (a + b)/2;
                err = abs(f(p));
        end
end
```

10 Programa de Cálculo da Fração de vazio e Tensão de cisalhamento no Escoamento Intermitente

- F: líquido; C: gás

```
DenC = DenG ;   DenF = DenL ;   VisC = VisG ;   VisF = VisL ;
```

- Frequência de Gregory & Scott (69) :

$$f = 0.026\ J_L * \left[(\frac{19.75}{J} + J) * (gD)\right]^{6/5} = 0.0813$$

```
f = 0.026 * (JL * (19.75/J + J) * (g*D) )^(6/5) ;
```

- Fração de Líquido de Gregory et al (78) :

$$R_S = (1 + (J/8.66)^{1.39})^{(-1)} = 0.9526$$

```
RS = ( 1 + (J/8.66)^1.39 ) ^(-1) ;
```

- Velocidade de translação da bolha alongada:

$$U_T = 1.2 + 0.35\sqrt{gD}\ sin(\theta)$$

```
% calculation of UT
Fr = J / ( dRho*g*D/DenL )
Eo = dRho*g*D^2/surTen ;
if Fr<3.5
        CoT = 1 + 0.2 * (sin(theta))^2  ;
        CinfT =  (0.542 - 1.76 / (Eo^0.56)) * cos(theta) + ...
                (0.345*sin(theta) ) / ( 1+3805*(Eo^-3.06) )^0.58 ;
else
        CoT = 1.2  ;
        CinfT = ( 0.345*sin(theta) ) / ( 1+3805*(Eo^-3.06) )^0.58 ;
end
VinfT = CinfT * ( dRho*g*D/DenL )^(1/2) ;
UT =  1.2 + 0.35 * sqrt(g*D) * sin(theta);
```

- Velocidade das bolhas dispersas:

$$U_B = C_{0B}\, J + V_{\infty B} = 1$$

```
regimen = 1 ;
% 1:agitado , 2:distorcido
if regimen==1
        n = 0;
else
        n=7/4;
```

```
end
CoB = 1 + 0.2 * (sin(theta))^2 ;
CinfB = 1.54 * (RS^n) * (Eo^(-1/4)) * sin(theta) ;
VinfB = CinfB * ( dRho*g*D/DenL )^(1/2) ;
UB = CoB * J + VinfB
```

- Cálculo de α

$$\alpha = \frac{J_G}{C_0 J + V_{GJ}} = \frac{J_G}{C_{0T} J + V_{\infty T}} = 0.41$$

```
Co = CoT ;
VGJ = VinfT ;
alpha = JG / ( Co*J+VGJ ) ;
```

- Cálculo da espessura do filme (VER PROGRAMA SEGUINTE)
- Cálculo da Força de atrito no filme

$\rho_S = (1 - R_S)\,\rho_G + R_S \rho_L$	$\mu_S = (1 - R_S)\,\mu_G + R_S \mu_L$	$Re_S = \frac{\rho_S V}{\mu}$
$C_{fS} = \left\{-3.6 * log\left[\frac{\varepsilon}{(3{,}7 D_S)^{1.11}} + \frac{6.9}{Re_S}\right]\right\}^{-2} = 0.0031$	$\tau_{WS} = C_{fS} * Den_S * J * \lvert J \rvert / 2 = 1.47$	$T_{WS} = \frac{\tau_W}{\cdot}$
$\beta = \frac{L_F \frac{P(L)}{L}}{L_F \frac{P(L)}{L} + L_S} = 0.47$	$T_W = \beta\,(T_{WC} + T_{WF}) + (1 - \beta)\,T_{WS} = 365.67$	

```
DenS = (1-RS)*DenG + RS * DenL;
VisS = (1-RS)*VisG + RS * VisL;
ReS = DenS * J * D / VisS ;
DS = D;
CfS = ( -3.6 * log( (rug / (3.7*DS)).^1.11 + 6.9/ReS )) .^(-2) ;
tauWS =  CfS * DenS * J * abs(J) / 2 ;
TWS =  tauWS * S / A

beta = (LF * PL/P) / ( LF *PL/P + LS )
TW = beta * (TWC + TWF ) + (1-beta)*TWS
```

11 Programa de cálculo do perfil da bolha alongada

- Cálculo do $\delta_{INICIAL} = 0.90$:

$$solve(f) = R_F - \frac{\lambda - sin\,(\lambda)}{2\pi} = 0$$

```
interface = 1;
tol = 1e-8;
LF = 0 ;
%%%% zF = 0 ;
dHF = (1e-4)*D ;
a = (1e-8)*D ;
b = (1-1e-8)*D ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if RS < 1
        RF = RS ;       %planar interface
        HF = bisectionHL(RF,D,a,b,tol) ;
else
        HF = D - dHF ;
end
delta0 = HF/D

function HF = bisectionHL(RF,D,a,b,tol)
err = 1 ;
while abs(err) > tol
```

```
        p = (a+b)/2. ;
        fa = funnHF(a, RF, D) ;

        fb = funnHF(b, RF, D) ;
        fp = funnHF(p, RF, D) ;

        %Bisection Method
        if fa*fp<0
                b = p ;
        else
                a = p ;
        end
        err = abs(b-a);
end
HF = p;
end

function
f=funnHF(HF, RF, D)
delta = HF/D ;
lambda = 2 * acos(1 - (2*delta) ) ;
f = RF - ( lambda - sin(lambda) ) / (2*pi) ;
end
```

- Cálculo da derivada: Primeira iteração com $\delta = 0.72$ para o primeiro $\frac{dH}{dz} < 0$

$$\frac{dh}{dz} = \frac{\frac{\tau_f S_f}{A_f} - \frac{\tau_G S_G}{A_G} - \tau_I S_I \left(\frac{1}{A_F} + \frac{1}{A_G}\right) + \Delta\rho\ g\ sin\beta}{\Delta\rho\ g\ cos\beta - \rho_L v_F \frac{(u_t - u_L)}{R_F^2}\frac{dR_F}{dh} - \rho_G v_G \frac{(u_t - u_B)(1-R_F)}{(1-R_F)^2}\frac{dR_F}{dh}} = -69.4$$

```
dhFdzF0=derivada(HF,J,DenC,DenF);
% delta_dH1= dhFdzF/ %
while derivada(HF,J,DenC,DenF)>0
        HF = HF - dHF;
end
delta1 = HF/D
dhFdzF1=derivada(HF,J,DenC,DenF)
h=HF;
dh=dHF;
i=0;
z=0;


function dhdz = derivada(hf,um,DenC,DenF)
```

```
Rf = area(hf)/area(D) ;
uf = (um - vbolha(um)*(1-Rf))/Rf;
ug = (um - Rf*uf)/(1-Rf);
ut = vbolha(um);
dRdh = (4/(pi*D))*sqrt(1-(2*hf/D-1)^2);
dhdz=0.5*fator(rhof,Dh(hf,'l'),uf,mif)*rhof*uf*abs(uf)*perimetro(hf,'l')/
    area(hf) ...
       - 0.5*fator(rhog,Dh(hf,'g'),ug,mig)*rhog*ug*abs(ug)*perimetro(hf,'g')/
          area(D-hf) ...
       - 0.5*fator(rhog,Dh(hf,'g'),ug,mig)*rhog*(ug-uf)*abs(ug-uf)*perimetro(
          hf,'i')*(1/area(hf)+1/area(D-hf)) ...
       + (rhof-rhog)*g*sin(angulo(theta));
dhdz = dhdz/((rhof-rhog)*g*cos(angulo(theta))-rhof*(ut - uf)*(ut-um)*dRdh/(
    Rf^2));
end


function A=area(h)
global D
A = pi-acos(2*h/D-1)+(2*h/D-1)*sqrt(1-(2*h/D-1)^2);
A = A*0.25*D^2;
end

function theta = angulo(theta)
```

```
theta = theta*pi/180;
end

function
f=fator(rho,Dh,v,mi)
Re = rho*Dh*v/mi;
f = max(0.079*Re^(-0.25),16*Re^(-1));
end


function S=perimetro(h,char)
 global D
if (char=='l')
S = D*(pi-acos(2*h/D-1));
elseif (char=='g')
S = D*acos(2*h/D-1);
else
S = D*sqrt(1-(2*h/D-1)^2);
end
end



function [TWC,TWF,UT]=TWCTWF...
```

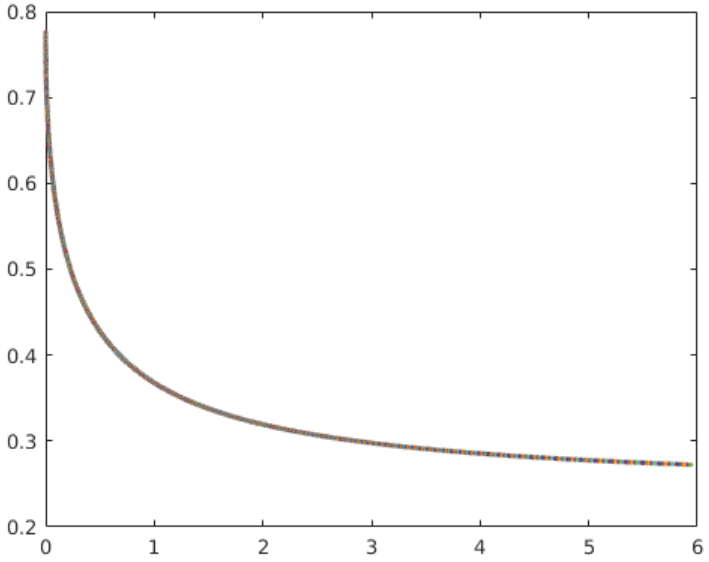
```
(HF,JG, JL, DenC , DenF, VisC, VisF, g, D , A, rug, surTen, dRho,theta,
   interface)

delta = HF/D
lambda = 2 * acos(1 - (2*delta) ) ;

SC = D * (pi-lambda/2) ;
SF = D * lambda/2 ;
SI = D * sin(lambda/2) ;
DC = D * ( 1 + ( sin(lambda/2)-sin(lambda)/2 ) / ...
       ( 2 * ( 2*pi-lambda*sin(lambda) ) )  )^-1 ;
DF = D * ( 1-sin(lambda)/lambda ) ;
RF = ( lambda - sin(lambda) ) / (2*pi)

ED = funED(4, JG, JL , DenC , DenF, VisC, VisF, g, D,  surTen, dRho ) ;
UC =  ( JG + ED*JL ) / (1-RF) ;
UF =  ( JL - ED*JL ) / RF ;
J = JG + JL ;
Fr = J / ( dRho*g*D/DenF )^(1/2) ;
Eo = dRho*g*(D^2)/surTen  ;
if Fr<3.5
        CoT = 1+0.2*(sin(theta))^2 ;
        CinfT = (0.542-1.76/(Eo^.56))*cos(theta) + ...
                  (0.345 / (1+3805*(Eo^3.06))^0.58 ) * sin(theta);
```

```
else
        CoT = 1.2 ;
        CinfT = (0.345 / (1+3805*(Eo^3.06))^0.58 ) * sin(theta);
end

VinfT = CinfT * (dRho*g*D/DenF)^(1/2) ;
UT = CoT * J + VinfT ;
ReC = DenC * UC * DC / VisC ;
ReF = DenF * UF * DF / VisF ;
CfC = ( -3.6 * log( rug/(3.7*DC)^1.11 +  6.9/ReC ) )^-2 ;
CfF = ( -3.6 * log( rug/(3.7*DF)^1.11 +  6.9/ReF ) )^-2 ;
tauWC = CfC * DenC * UC * abs(UC) / 2 ;
tauWF = CfF * DenF * UF * abs(UF) / 2 ;
TWC = tauWC * SC / A ;      TWF = tauWF * SF / A ;
end

function ED = funED(model, JG, JL , DenC , DenF, VisC, VisF, g, D,  surTen,
   dRho )
         % Wallis (Tabela4.3)
ReL = DenF * JL * D / VisF;
We1 = ( DenC * JG^2 * D / surTen ) * ( dRho / DenC )^(1/3);

if model==1
        % Wallis (1969)
```

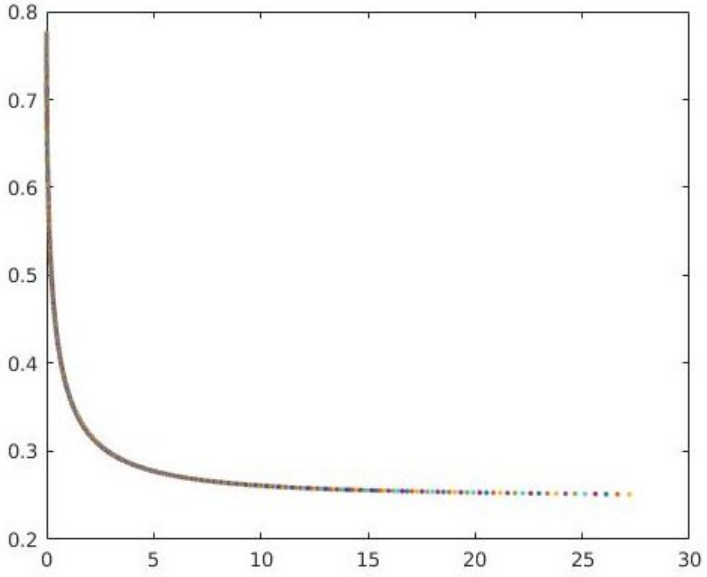
```
        phi = 1e4 * JG * VisC / surTen * (DenC/DenF)^0.5;
        ED = 1 - exp( -0.125 * (phi-1.5) ) ;
elseif model==2
        % Oliemans (1986)
        Coef = (10^-2.52)*(DenC^0.18)*(DenF^1.08)*(VisC^0.28)*...
            (VisF^0.27)*(JG^1.44)*(JL^0.7)*(surTen^-1.8)*(g^0.46)*(D^1.72) ;
        ED = Coef/(Coef+1);
elseif model == 3
        % Ishii & Mishima (1989)
        ED = tanh ( 7.25e-7 * ReL^(1/4) * We1*(5/4) ) ;
else
        % Sawant et al. (2008)
        ReLmin = 250*log(ReL) - 1265 ;
        Em = 1 - ReLmin/ReL ;
        a = 2.31e-4 * ReL^-0.35 ;
        ED = Em * tanh (a * We1^1.25);
end

end
```

- Cálculo da altura de equilíbrio:

  1. Critério de parada: $\%Erro = 0.5\%$ (Vinicius)
     - $\delta_1 = 0.3171$; $L_F = 6m$??????????????????????



2. Critério de parada: Conservação da massa: $JG > (1 - R_S) * U_B + (R_S - \overline{R_F}) * L_F * f$ (Lima)
   - $\delta_2 = 0.299$; $L_F = 26m$; $\frac{dh}{dz} = -69$??????????????????????



- Cálculo das tensões de cisalhamento e força de atrito
  - $\tau_{WC} = C_{fC} * Den_C * U_C * |U_C| /2 = 0.0013\ N/m^2$
  - $\tau_{WF} = C_{fF} * Den_F * U_F * |U_F| /2 = 8.54\ N/m^2$
  - $T_{WC} = \frac{\tau_{WC} * S}{A} = 0.13N$
  - $T_{WF} = \frac{\tau_{WF} * S}{A} = 512N$
- Cálculo do comprimento do pistão:
  - $LS = \frac{U_T}{f} - L_F = 6,52m$??????????????????????

12 Seção do programa para calcular a pressão na em determinada seção

- Resolver a EDO

$U_L(i) =$	$\mu_S = (1 - R_S)\,\mu_G + R_S\mu_L$	$Re_S = \frac{\rho_S V}{\mu}$
$C_{fS} = \left\{-3.6 * log\left[\frac{\varepsilon}{(3{,}7D_S)^{1.11}} + \frac{6.9}{Re_S}\right]\right\}^{-2} = 0.0031$	$\tau_{WS} = C_{fS} * Den_S * J * \lvert J\rvert /2 = 1.47$	$T_{WS} = \frac{\tau_W}{A}$
$\beta = \frac{L_F \frac{P(L)}{L}}{L_F \frac{P(L)}{L} + L_S} = 0.47$	$T_W = \beta\,(T_{WC} + T_{WF}) + (1 - \beta)\,T_{WS} = 365.67$	