```
clc;
close all;
clear;

%% 1. Input Data
% --- Physical Data ---
FLUID = 'T';                % Insert W, T or A for water, Therminol 66 or Air
Di = 0.020;                 % Internal diameter (m)
Do = 0.024;                 % External diameter (m)
L = 20;                     % Tube length (m)
pin = 2e5;                  % Inlet pressure (Pa)
Tin = 95 + 273.15;          % Inlet temperature (K)
Text = 20 + 273.15;         % Ambient temperature (K)
pext = 101325;              % Ambient pressure (Pa)
lambda_t = 36;              % Thermal conductivity of tube (W/m·K)
g = 9.81;
beta = 1 / Text;

% --- Numerical Data ---
N = 100;                    % Number of control volumes
tolerance = 1e-5;           % Convergence criterion
del_x = L / N;              % Control volume length

%% Assign thermophysical property functions based on the fluid type
if strcmp(FLUID, 'W')
    vin = 1;                    % Inlet velocity (m/s)
    density = @density_water;
    viscosity = @viscosity_water;
    conductivity = @conductivity_water;
    SP = @SP_water;
elseif strcmp(FLUID, 'T')
    vin = 1;                    % Inlet velocity (m/s)
    density = @density_therminol;
    viscosity = @viscosity_therminol;
    conductivity = @conductivity_therminol;
    SP = @SP_therminol;
elseif strcmp(FLUID, 'A')
    vin = 30;                    % Inlet velocity (m/s)
    density = @density_air;
    viscosity = @dynamic_visc_air;
    conductivity = @conduct_air;
    SP = @SP_air;
end

%% 2. Geometry and Inlet Properties
% --- Geometry Calculations ---
S_f = pi * (Di / 2)^2;       % Fluid cross-sectional area (m²)
S_t = pi * (Do^2-Di^2)/4;    % Fluid cross-sectional area (m²)
Pi = pi * Di;                % Internal perimeter (m)
Po = pi * Do;                % Internal perimeter (m)
% --- Inlet Properties ---
roin = density(Tin,pin);    % Inlet density (kg/m³)
m_in = roin * vin * S_f;     % Inlet mass flowrate (kg/s)
```

```
%% 2.1 Mesh Definition
Xp_f = linspace(0, L, N + 1);                    % Control volume boundaries
Xp_t = [0, (Xp_f(1:end-1) + Xp_f(2:end)) / 2, L]; % Node positions for the tube

%% 3. Initial Conditions
T_t = linspace(Tin, Text, N + 2); % Initial temperature for tube nodes
v_f = vin * ones(1, N + 1);                    % Initial velocity (m/s)
P_f = pin * ones(1, N + 1);                    % Initial pressure (Pa)
T_f = linspace(Tin, Text, N + 1);                    % Initial temperature (K)
ro_f = roin * ones(1, N + 1);                  % Initial density

%% Start global iteration
convergenza_glo = false;

while ~convergenza_glo

    %% 4. Internal Fluid Evaluation
    convergenza_int = false; % Convergence flag for internal fluid
    alpha_i = zeros(1, N); % Initialize the vector for alpha_i
    while ~convergenza_int
        T_old = T_f; % Store previous iteration values
        P_old = P_f;
        ro_old = ro_f;
        v_old = v_f;

        for i = 1 : N
            % Average properties in control volume
            T_i = (T_f(i+1) + T_f(i)) / 2;
            P_i = (P_f(i+1) + P_f(i)) / 2;
            v_i = (v_f(i+1) + v_f(i)) / 2;
            ro_i = (ro_f(i+1) + ro_f(i)) / 2;

            % Thermodynamic and flow properties
            mu_i = viscosity(T_i, ro_i);
            muw_i = viscosity(T_t(i), ro_i);
            cp_i = SP(T_i);
            lambda_i = conductivity(T_i);
            Re_i = (ro_i * v_i * Di) / mu_i;
            Pr_i = (mu_i * cp_i) / lambda_i;
            Gz_i = Re_i * Pr_i * Di / del_x;
            alpha_i (i) = alpha(Re_i, Pr_i, Gz_i, mu_i, muw_i, lambda_i, Di, ↙
del_x);
            f_i = f(Re_i);

            % Update equations
            P_f(i+1) = P_f (i) - (m_in * (v_f(i+1)-v_f(i)) + f_i * Pi * del_x * ↙
((ro_i * v_i^2) / 2)) / S_f  ;
            T_f(i+1) = T_f(i) + (alpha_i(i) * Pi * del_x * (T_t(i+1) - T_i) - m_in↙
* (v_f(i+1)^2-v_f(i+1)^2)/2)/(m_in*cp_i);
            ro_f(i+1) = density(T_f(i+1),P_f(i+1));
            v_f(i+1) = m_in / (ro_f(i+1) * S_f);
        end
```

```
        % Compute residuals for convergence check
        residual_T = max(abs(T_f - T_old));
        residual_P = max(abs(P_f - P_old));
        residual_ro = max(abs(ro_f - ro_old));
        residual_v = max(abs(v_f - v_old));

        % Check convergence
        if max([residual_T, residual_P, residual_ro, residual_v]) < tolerance
            convergenza_int = true;
        end
    end

   alpha_ext = zeros(1, N); % Save vector

    for i = 1:N
        % All variables evaluated at film temperature Tm = 0.5 * (Text + T_t(i))
        Tm = 0.5 * (Text + T_t(i+1));
        mu = dynamic_visc_air(Tm);
        ro= density_air(Tm, pext);
        lambda= conduct_air(Tm);
        beta= 1/Tm;
        cp= SP_air(Tm);

        Gr = g * beta * ro^2 * abs(T_t(i+1) - Text) * Do^3 / mu^2;
        Pr = mu * cp / lambda;
        Ra = Gr * Pr;

        if Ra < 10^9
            C = 0.47;
            n = 1/4;
            k = 1;
        else
            C = 0.1;
            n = 1/3;
            k = 1;
        end

        Nu_ext = C * Ra^n * k;
        alpha_ext(i) = (lambda * Nu_ext) / Do;
    end

    %% 5. Solve the Tube Temperature
    % Update tube node temperatures iteratively
    % Get the parameter P(i)
    % (i == 1) Internal Boundary condition
    aE = 1;
    aP = 1;
    P(1)= aE / aP;

    for i = 2:(N+1)
        % Calculation of coefficients
        aW = lambda_t * S_t / (Xp_t(i) - Xp_t(i-1));
```

```
        aE = lambda_t * S_t / (Xp_t(i+1) - Xp_t(i));
        aP = aE + aW + alpha_i(i-1)*Pi*(Xp_t(i) - Xp_t(i-1)) ...
                     + alpha_ext(i-1)*Po*(Xp_t(i) - Xp_t(i-1));
        P(i)= aE / ( aP - aW * P(i-1) );
    end

    % (i == N+2) External Boundary condition
    % aE = 0;
    % aW = lambda / (r(N+2) - r(N+1));
    % aP = aW + alpha_b;
    % P(N+2)=aE / (aP - aW * P(N+1)); % This part of the code is useless
    P(N+2)=0;

    % Get the parameter R(i)
    % (i == 1) Internal Boundary condition
    aE = 1;
    aP = aE;
    bP = 0;
    R(1)= bP / aP;

    for i = 2 : (N+1) % Internal nodes
        aW = lambda_t * S_t / (Xp_t(i) - Xp_t(i-1));
        aE = lambda_t * S_t / (Xp_t(i+1) - Xp_t(i));
        aP = aE + aW + alpha_i(i-1)*Pi*(Xp_t(i) - Xp_t(i-1)) ...
                     + alpha_ext(i-1)*Po*(Xp_t(i) - Xp_t(i-1));
        bP = alpha_i(i-1)*0.5*(T_f(i) + T_f(i-1))*Pi*(Xp_t(i) - Xp_t(i-1)) ...
           + alpha_ext(i-1)*Text*Po*(Xp_t(i) - Xp_t(i-1));
        R(i)=(bP + aW * R(i-1) ) / ( aP - aW * P(i-1) );
    end

    % (i == N+2) External Boundary condition
    aW = 1;
    aP = aW;
    bP = 0;
    R(N+2)=( bP + aW * R(N+1) ) / ( aP - aW * P(N+1) );

    T_new(N+2) = R(N+2);  % Initialize the vector

    for i=(N+1):-1:1
        T_new(i)=  P(i) * T_new(i+1) + R(i);
    end

    %% 6. Global Convergence Check
    if max(abs(T_new - T_t)) < tolerance
        convergenza_glo = true;
    end

    T_t = T_new;
end

%% 7a. Final calculations
alpha_i = alpha_i(end);
alpha_o=alpha_ext(end);
```

```
vout=v_f(end);                                                    %final↙
outlet velocity
Pout=P_f(end);                                                    %final↙
outlet pressure
Tout=T_f(end)-273.15;                                             %final↙
outlet temperature

%% Table of Variables and Final Results
% Calculation of key variables and creation of the table

% Final variables
alpha_i_mean = mean (alpha_i);
alpha_o_mean = mean (alpha_o);
T_t_ave = mean(T_t);                          % Average temperature of the tube (K)
T_f_ave = mean(T_f);                          % Average temperature of the fluid (K)
Qw_final = m_in * cp_i * (Tin - T_f(end)); % Total heat exchanged (W)
Re_i_final = Re_i;                            % Final Reynolds number inside the tube
Pr_i_final = Pr_i;                            % Final Prandtl number inside the tube
K_i_final = (mu_i / muw_i)^(0.14);            % Nusselt correction factor
f_i_final = f_i;                              % Final friction factor
alpha_i_final = alpha_i(end);                 % Final internal convection coefficient
Gro_final = Gr;                               % Grashof number
Pr_o_final = Pr;                              % Final Prandtl number outside the tube
alpha_o_final = alpha_ext(end);               % Final external convection coefficient
Rcond = 1 / (2 * pi * lambda_t) * log(Do/Di);
Utot = ( 1 /(alpha_i_mean * Pi * L) + Rcond + (1/(alpha_o_mean * Po * L)));
Uo_final = 1 / (Utot * Po * L); % External heat transfer coefficient
Ui_final = 1 / (Utot * Pi * L); % Internal heat transfer coefficient
vout_final = vout;                            % Final outlet velocity (m/s)
Pout_final = Pout;                            % Final outlet pressure (Pa)
Tout_final = Tout;                            % Final outlet temperature (K)

% Create the table with the results
T = table(Re_i_final, Pr_i_final, K_i_final, f_i_final, alpha_i_final, ...
    Gro_final, Pr_o_final, alpha_o_final, Uo_final, Ui_final, ...
    vout_final, Pout_final, Tout_final, Qw_final, ...
    'VariableNames', {'Re_i', 'Pr_i','K_i','f_i', 'alpha_i', 'Gro', 'Pr_o', ...
    'alpha_o', 'U_o', 'U_i', 'v_out', 'P_out', 'T_out', 'Q_w'});

% Display the table
disp('Table of Final Results:');
disp(T);

%% Figura (1): Temperature T_f e T_t
figure(1);
x = Xp_f;
T_t = T_t(2:end);

% Grafico con un unico asse
plot(x, T_f - 273.15, 'b-', 'LineWidth', 1.5); % T_f in blu
hold on;
plot(x, T_t - 273.15, 'r--', 'LineWidth', 1.5); % T_t in rosso
hold off;
```

```
% Etichette e dettagli
ylabel('Temperature (°C)');
xlabel('Position (m)');
grid on;
title('Figure (1): Temperature of Fluid (T_f) and Tube (T_t)');
legend({'T_f (Fluid Temperature)', 'T_t (Tube Temperature)'}, 'Location',↙
'northeast');

%% Figura (2): Pressione P_f e Velocità v_f
figure(2);

% Asse per P_f (sinistra)
yyaxis left;
plot(x, P_f / 10^5, 'b-.', 'LineWidth', 1.5); % P_f in verde
ylabel('P_f (Pressure, bar)', 'Color', 'b'); % Colore asse coerente con P_f
ylim([0, max(P_f / 10^5) + 1]);
xlabel('Position (m)');
grid on;

% Asse per v_f (destra)
yyaxis right;
plot(x, v_f, 'r-', 'LineWidth', 1.5); % v_f in magenta
ylabel('v_f (Velocity, m/s)', 'Color', 'r'); % Colore asse coerente con v_f
ylim([0, max(v_f) + 5]);

% Titolo e legenda
title('Figure (2): Pressure (P_f) and Velocity (v_f)');
legend({'P_f', 'v_f'}, 'Location', 'northeast');




%% Functions liquid water

% Density calculation liquid water
function sum = density_water(T,~)
    sum = 847.2 + 1.298 * T - (2.657e-3) * T^2;
end

% Viscosity calculation liquid water
function sum = viscosity_water(T,~)
    if T < 353
        sum = 0.9149 - (1.2563e-2) * T + (6.9182e-5) * T^2 - ...
              (1.9067e-7) * T^3 + (2.6275e-10) * T^4 - ...
              (1.4474e-13) * T^5;
    else
        sum = (3.7471e-2) - (3.5636e-4) * T + (1.3725e-6) * T^2 - ...
              (2.6566e-9) * T^3 + (2.5766e-12) * T^4 - ...
              (1e-15) * T^5;
    end
end
```

```
% Thermal conductivity calculation liquid water
function sum = conductivity_water(T)
    sum = -0.722 + 7.168e-3 * T - 9.137e-6 * T^2;
end

% Specific heat calculation
function sum = SP_water(T)
    sum = 5648.8 - 9.140 * T + 14.21e-3 * T^2;
end

%% Therminol 66
% Funzioni per Therminol 66
function sum = density_therminol(T,~)
    sum = 1164.45 - 0.4398 * T - 3.21e-4 * T^2;
end

function sum = viscosity_therminol(T,ro)
    sum = ro * exp(-16.096 + (586.38 / (T - 210.65)));
end

function sum = conductivity_therminol(T)
    sum = 0.116 + 4.9e-5 * T - 1.5e-7 * T^2;
end

function sum = SP_therminol(T)
    sum = 658 + 2.82 * T + 8.97e-4 * T^2;
end

%% Internal heat transfer and friction factor functions

% Heat transfer coefficient calculation
function sum = alpha(Re, Pr, Gz, mui, muwi, lambda, D, del_x)
    if Re < 2000 && Gz > 10
        K = ((D / del_x)^(1/3)) * ((mui / muwi)^(0.14));
        C = 1.86; m = 1/3; n = 1/3;
    elseif Re < 2000 && Gz < 10
        K = 1;
        C = 3.66; m = 0; n = 0;
    elseif Re > 2000 && (0.6 < Pr && Pr < 100)
        K = (mui / muwi)^(0.14);
        C = 0.027; m = 0.8; n = 0.33;
    else
        K = 1;
        C = 0.023; m = 0.8; n = 0.04;
    end
    Nu = C * (Re^m) * (Pr^n) * K;
    sum = (Nu * lambda) / D;
end

% Darcy friction factor calculation
function sum = f(Re)
    if Re <= 2000
        sum = 16 / Re;
```

```
    elseif Re > 5*1e3 && Re < 3*1e4
        sum = 0.079*Re^(-0.25);
    else
        sum = 0.046*Re^(-0.20);
    end
end



%% Functions for external fluid (dry air)

% Density dry air
function sum = density_air(T,p)
    sum = p / (287 * T);
end

% Dynamic viscosity
function sum = dynamic_visc_air(T,~)
    sum = ( 2.5393*1e-5 * (T/273.15)^0.5 ) / (1 + 122/T);
end

% Heat coefficient
function sum = SP_air(T)
    sum =  1031.5 - 0.210 * T + 4.143*1e-4 * T^2;
end

% Conductivity
function sum = conduct_air(T)
    sum = (2.728*1e-3 + 7.776*1e-5 * T);
end
```