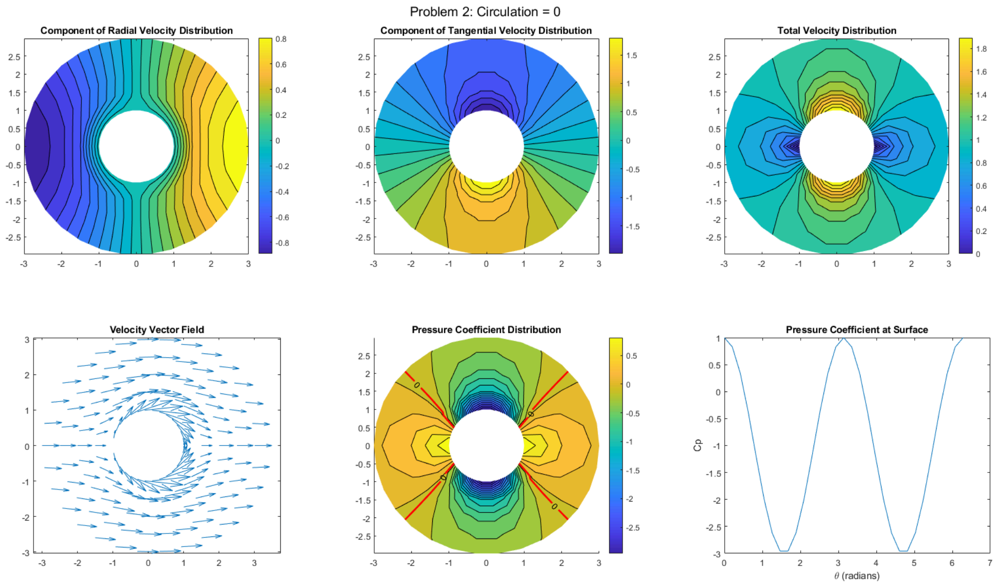
```
close all;
clc, clear;

%{
    |=====ENG 180 Project 9=====|
    File:               ENG126_PJ1_main_Hieu_Bui.m
    Author:             Hieu Bui
    Date:               04/09/2023
    Description:        Compute non-dimensional parameters for flow over
                          2D circular cylinder with uniform, doublet, and
                          vortex flows
%}

[Cl_0,Cd_0] = circularBodyFlow(0,0) % no circulation
[Cl_1,Cd_1] = circularBodyFlow(1,0) % circulation less than critical
[Cl_2,Cd_2] = circularBodyFlow(2,0) % circulation equal critical
[Cl_3,Cd_3] = circularBodyFlow(3,1) % circulation greater than critical

% gamma => gamma values for problem 2
% p3 => compute problem 3 when p3 =1
function [lift_coefficient, drag_coefficient] = circularBodyFlow(gamma,p3)
    cylinder_radius = 1;
    farField_velocity = 1;

    % Polar grid
    r_range = linspace(cylinder_radius, 3, 5);
    theta_range = linspace(0, 2*pi, 31);
    [r, theta] = meshgrid(r_range, theta_range);
    x_polar = r.*cos(theta);
    y_polar = r.*sin(theta);

%===========Problem 2=============%
    % Compute velocity components
    critical_circulation = 4*pi*farField_velocity*cylinder_radius;
    radial_velocity = farField_velocity*cos(theta).*(1-((cylinder_radius.^2)./
(r.^2)));
    if gamma == 0 % no circulation
        angular_velocity = -
farField_velocity*sin(theta).*(1+((cylinder_radius.^2)./(r.^2)));
    elseif gamma == 1 % circulation less than critical
        angular_velocity = -
farField_velocity*sin(theta).*(1+((cylinder_radius.^2)./(r.^2)))-
(critical_circulation*.4)./(2*pi*r);
    elseif gamma == 2 % circulation equal critical
        angular_velocity = -
farField_velocity*sin(theta).*(1+((cylinder_radius.^2)./(r.^2)))-
critical_circulation./(2*pi*r);
    elseif gamma == 3 % circulation greater than critical
        angular_velocity = -
farField_velocity*sin(theta).*(1+((cylinder_radius.^2)./(r.^2)))-
(critical_circulation*1.4)./(2*pi*r);
    end
```

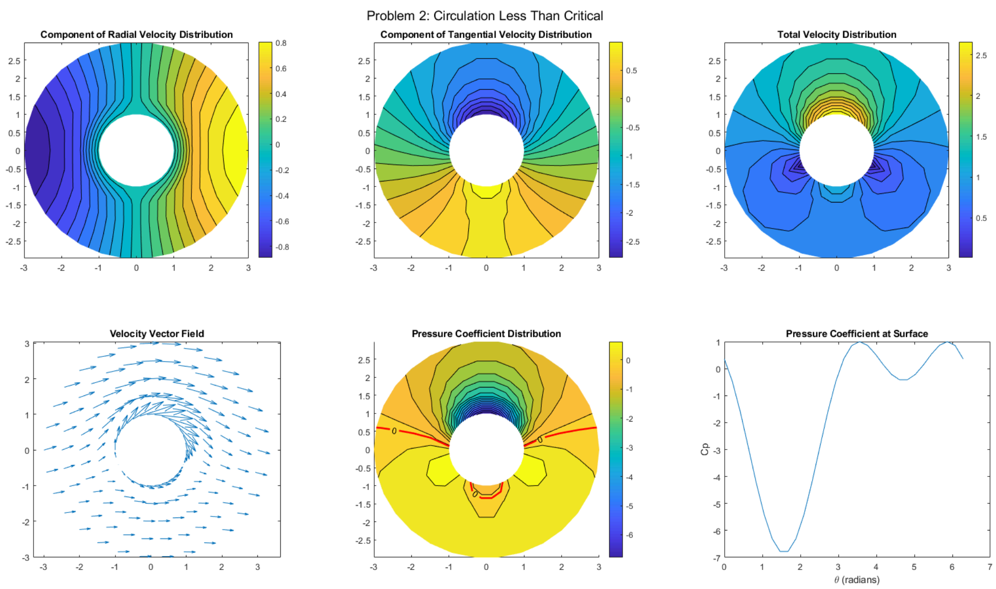
```
    x_velocity = -sin(theta).*angular_velocity + cos(theta).*radial_velocity;
    y_velocity = cos(theta).*angular_velocity + sin(theta).*radial_velocity;
    total_velocity = sqrt(y_velocity.^2+x_velocity.^2);

    % aerodynamic coefficients
    pressure_coefficient = 1 - ((radial_velocity.^2+angular_velocity.^2)./
farField_velocity);
    surface_pressure_coefficient = pressure_coefficient(:,1)';
    lift_coefficient = -.5.*trapz(theta_range,
 (surface_pressure_coefficient.*sin(theta_range)));
    drag_coefficient = -.5.*trapz(theta_range,
 (surface_pressure_coefficient.*cos(theta_range)));

%===========Problem 3=============%
    if p3==1
        circulation_vector = linspace(0,critical_circulation*1.4,20); %
 circulation gamma as a vector
        % declaring vectors
        Cl = zeros(1,20);
        Cl_analytical = zeros(1,20);
        Cd = zeros(1,20);
        lift = zeros(1,20);
        drag = zeros(1,20);
        theta_stagnation_analytical = zeros(1,20);
        theta_stagnation_numerical = zeros(1,20);
        % compute aerodynamics parameters for all circulation values
        for i = 1:length(circulation_vector)
            tangential_velocity = -
farField_velocity*sin(theta).*(1+((cylinder_radius.^2)./(r.^2)))-
circulation_vector(i)./(2*pi*r);
            Cp = 1 - ((radial_velocity.^2+tangential_velocity.^2)./
farField_velocity);
            Cp_surf = Cp(:,1)';
            lift_integration_formula =
 Cp_surf.*cylinder_radius.*sin(theta_range);
            drag_integration_formula =
 Cp_surf.*cylinder_radius.*cos(theta_range);
            Cl(1,i) = -.5.*trapz(theta_range, (Cp_surf.*sin(theta_range)));
            Cl_analytical(1,i) = circulation_vector(i)/
(cylinder_radius*farField_velocity);
            Cd(1,i) = -.5.*trapz(theta_range, (Cp_surf.*cos(theta_range)));
            lift(1,i) = -trapz(theta_range, lift_integration_formula);
            drag(1,i) = -trapz(theta_range, drag_integration_formula);
            surface_velocity =
 sqrt(tangential_velocity(:,1).^2+radial_velocity(:,1).^2);
            [~,X] = min(surface_velocity);
            theta_stagnation_numerical(1,i) = abs(wrapToPi(theta(X,1)));
            theta_stagnation_analytical(1,i) = asin(circulation_vector(i)/
(4*pi*farField_velocity*cylinder_radius));
        end
        % Plotting Cl, Cd, and stagnation angle
        figure('units','normalized','outerposition',[0 0 1 1]);
        subplot(3,1,1)
        hold on
```

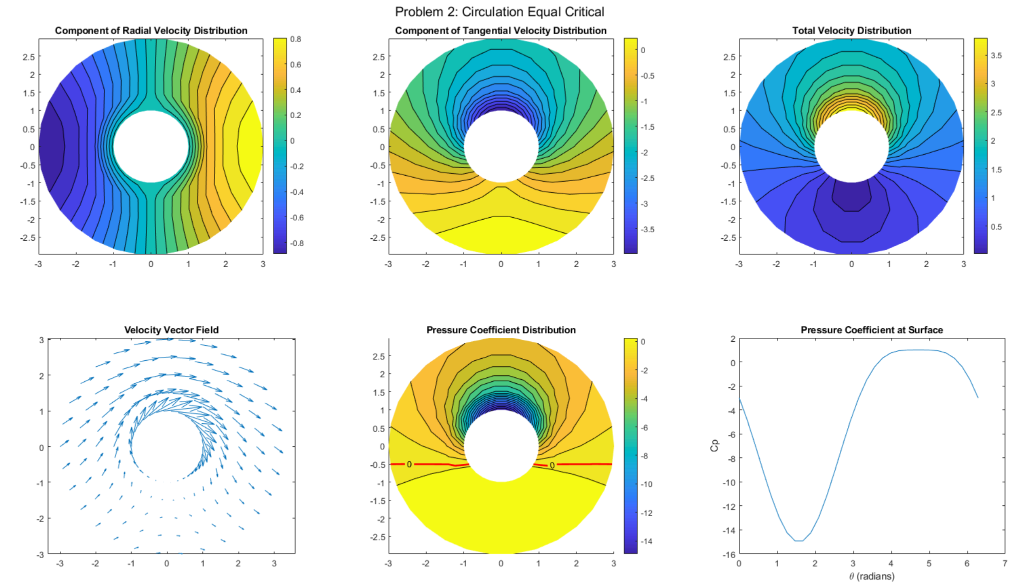
```
        plot(circulation_vector,Cl_analytical);
        plot(circulation_vector,Cl,'ro');
        xlabel('\Gamma',Interpreter='tex')
        ylabel('C_l',Interpreter='tex')
        legend('Analytical Solution','Numerical Solution')
        title('Circulation vs Coefficient of Lift')
        subplot(3,1,2)
        plot(circulation_vector,Cd);
        xlabel('\Gamma',Interpreter='tex')
        ylabel('C_d',Interpreter='tex')
        title('Circulation vs Coefficient of Drag')
        subplot(3,1,3)
        hold on
        plot(circulation_vector,theta_stagnation_analytical)
        plot(circulation_vector,theta_stagnation_numerical)
        xlabel('\Gamma',Interpreter='tex')
        ylabel('\theta_{stag} (radian)',Interpreter='tex')
        legend('Analytical Solution','Numerical Solution')
        title('Circulation vs Stagnation Angle')
        sgtitle('Problem 3: Varying Circulation')
    else
    end
%===========Plotting for problem 2============%

    figure('units','normalized','outerposition',[0 0 1 1]);
    subplot(2,3,1)
    contourf(x_polar,y_polar,radial_velocity,20);
    colorbar;
    title('Component of Radial Velocity Distribution');
    subplot(2,3,2)
    contourf(x_polar,y_polar,angular_velocity,20);
    colorbar;
    title('Component of Tangential Velocity Distribution');
    subplot(2,3,3)
    contourf(x_polar,y_polar,total_velocity,20);
    colorbar;
    title('Total Velocity Distribution');
    subplot(2,3,4)
    quiver(x_polar,y_polar,x_velocity,y_velocity);
    axis equal;
    title('Velocity Vector Field')
    subplot(2,3,5)
    hold on
    contourf(x_polar,y_polar,pressure_coefficient,20);
    contour(x_polar, y_polar, pressure_coefficient,
 [0,0],'r','LineWidth',2,'ShowText','on');
    colorbar;
    title('Pressure Coefficient Distribution');
    subplot(2,3,6)
    plot(theta_range,surface_pressure_coefficient);
    title('Pressure Coefficient at Surface')
    xlabel('\theta (radians)',Interpreter='tex');
    ylabel('Cp');
    if gamma == 0
```

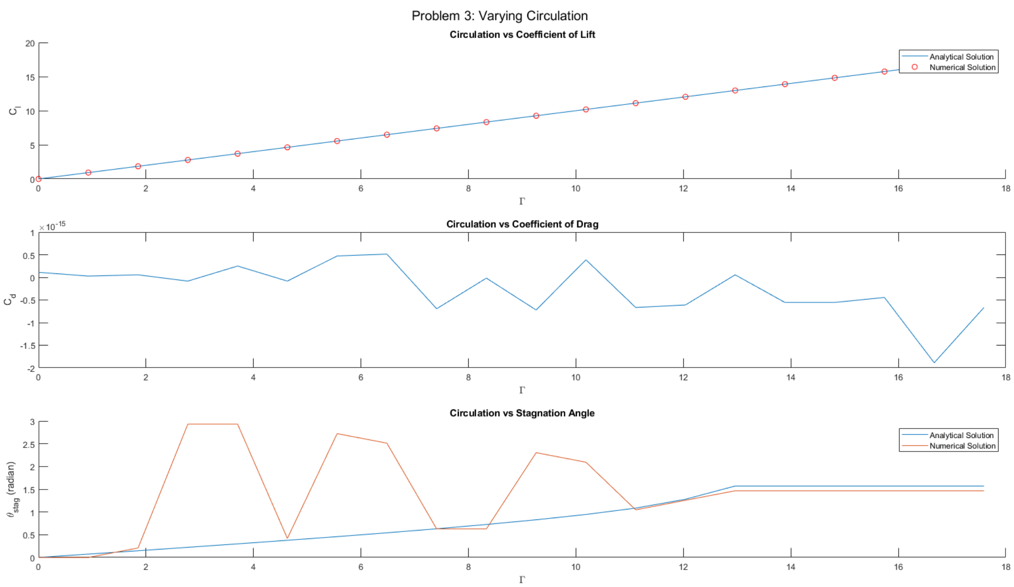
```
            sgtitle('Problem 2: Circulation = 0')
        elseif gamma == 1
            sgtitle('Problem 2: Circulation Less Than Critical')
        elseif gamma == 2
            sgtitle('Problem 2: Circulation Equal Critical')
        else
            sgtitle('Problem 2: Circulation Greater Than Critical')
    end
end
```

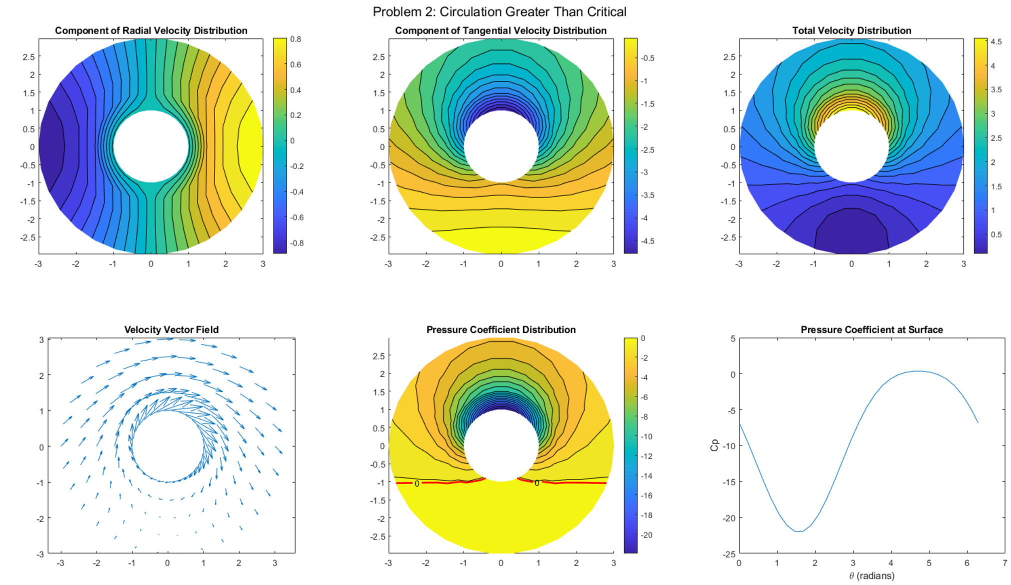
```
Cl_0 =

   1.9429e-16


Cd_0 =

   1.1102e-16


Cl_1 =

    5.0265


Cd_1 =

   1.3878e-16


Cl_2 =

   12.5664


Cd_2 =

  -1.2212e-15

Warning: Imaginary parts of complex X and/or Y arguments ignored.

Cl_3 =

   17.5929


Cd_3 =

  -6.6613e-16
```


Problem 2: Circulation = 0
Component of Radial Velocity Distribution
Component of Tangential Velocity Distribution
Total Velocity Distribution
Velocity Vector Field
Pressure Coefficient Distribution
Pressure Coefficient at Surface
Cp
θ (radians)


Problem 2: Circulation Less Than Critical
Component of Radial Velocity Distribution
Component of Tangential Velocity Distribution
Total Velocity Distribution
Velocity Vector Field
Pressure Coefficient Distribution
Pressure Coefficient at Surface
Cp
θ (radians)


Problem 2: Circulation Equal Critical
Component of Radial Velocity Distribution
Component of Tangential Velocity Distribution
Total Velocity Distribution
Velocity Vector Field
Pressure Coefficient Distribution
Pressure Coefficient at Surface
Cp
θ (radians)


Problem 3: Varying Circulation
Circulation vs Coefficient of Lift
Analytical Solution
Numerical Solution
C_l
Γ
Circulation vs Coefficient of Drag
$\times 10^{-15}$
C_d
Γ
Circulation vs Stagnation Angle
Analytical Solution
Numerical Solution
θ_{stag} (radian)
Γ



Published with MATLAB® R2021b