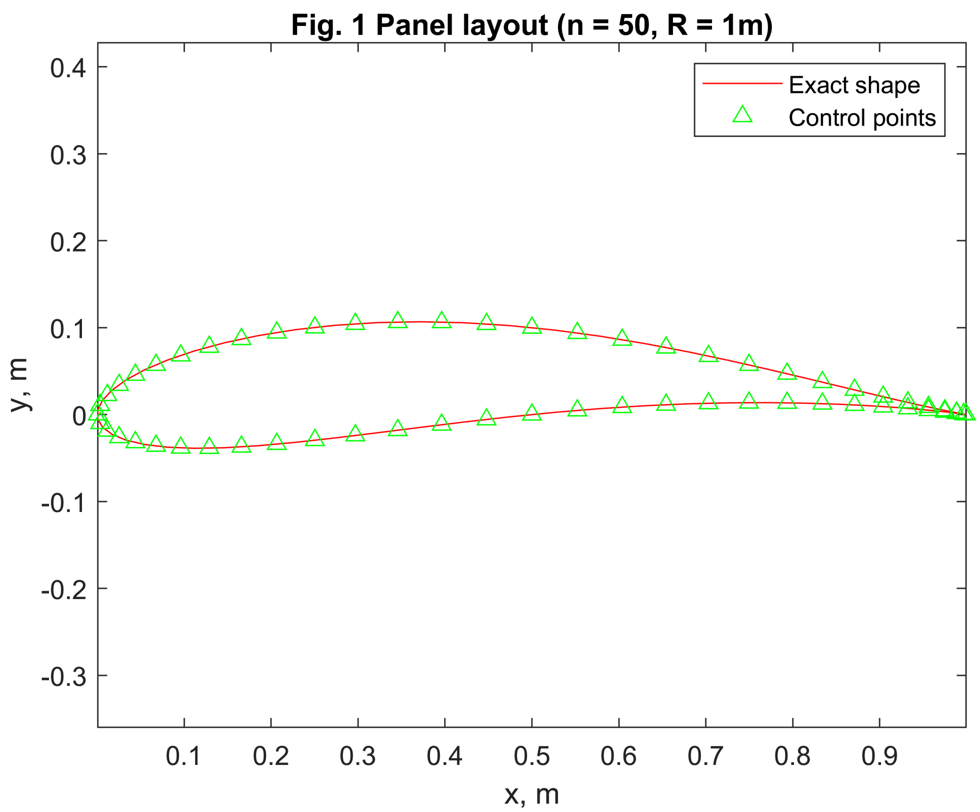
```
clc;clear;close all;
V_inf = 2;                          % freestream velocity
R = 1;                              % cylinder radius
n = 60;                             % number of panels
d_theta = 2*pi/n;                   % resolution of angles
alpha = 0;                          % angle of attack
theta = pi+pi/n:-d_theta:-pi+pi/n;  % angles of boundary points of panels

b=2;
t=0.2;
c=0.05;                             %Initalizing Constants
p=1;
e=1;
r=0;

%Initialising X and Y arrays which contain the coordinate points of the
%airfoil
X = 0.5+0.5.*(abs(cos(theta)).^b)./cos(theta);
Y = (1-X.^p).*(t/2).*((abs(sin(theta))).^b)./sin(theta) + c.*sin((X.^e).*pi) + r.*sin(X.*2.*pi)

Phi =   zeros(n,1);                 % angle from Vinf to bottom of panel
beta =  zeros(n,1);                 % angle from Vinf to outward normal of panel
conX =  zeros(n,1);                 % X coordinates of control points
conY =  zeros(n,1);                 % Y coordinates of control points
S =     zeros(n,1);                 % panel length
for i = 1:n
    Phi(i) = -alpha + atan2((Y(i+1)-Y(i)),(X(i+1)-X(i)));
    beta(i) = Phi(i)+pi/2;
    if beta(i)>2*pi, beta(i)=beta(i)-2*pi;
    elseif beta(i)<0, beta(i)=beta(i)+2*pi; end
    conX(i) = (X(i+1)+X(i))/2;
    conY(i) = (Y(i+1)+Y(i))/2;
    S(i) = sqrt((X(i+1)-X(i))^2 + (Y(i+1)-Y(i))^2);
end

close all
figure(1)
plot(X,Y,'r',conX,conY,'g^');
axis equal; legend('Exact shape','Control points')
xlabel('x, m'); ylabel('y, m'); title('Fig. 1 Panel layout (n = 50, R = 1m)');

In = zeros(n,n);        % integral, normal
Is = zeros(n,n);        % integral, tangent
V_n = zeros(n,1);       % normal velocity by Vinf
effJ = zeros(n-1,n);    % effective j that exclude i
for i=1:n
    effJ(:,i)=[1:i-1 i+1:n]; xi=conX(i); yi=conY(i);
    for k = 1:n-1
        j = effJ(k,i); Xj = X(j); Yj = Y(j);
        A = -(xi-Xj)*cos(Phi(j))-(yi-Yj)*sin(Phi(j));
        B = (xi-Xj)^2+(yi-Yj)^2;
        C = sin(Phi(i)-Phi(j));
        D = (yi-Yj)*cos(Phi(i))-(xi-Xj)*sin(Phi(i));
```

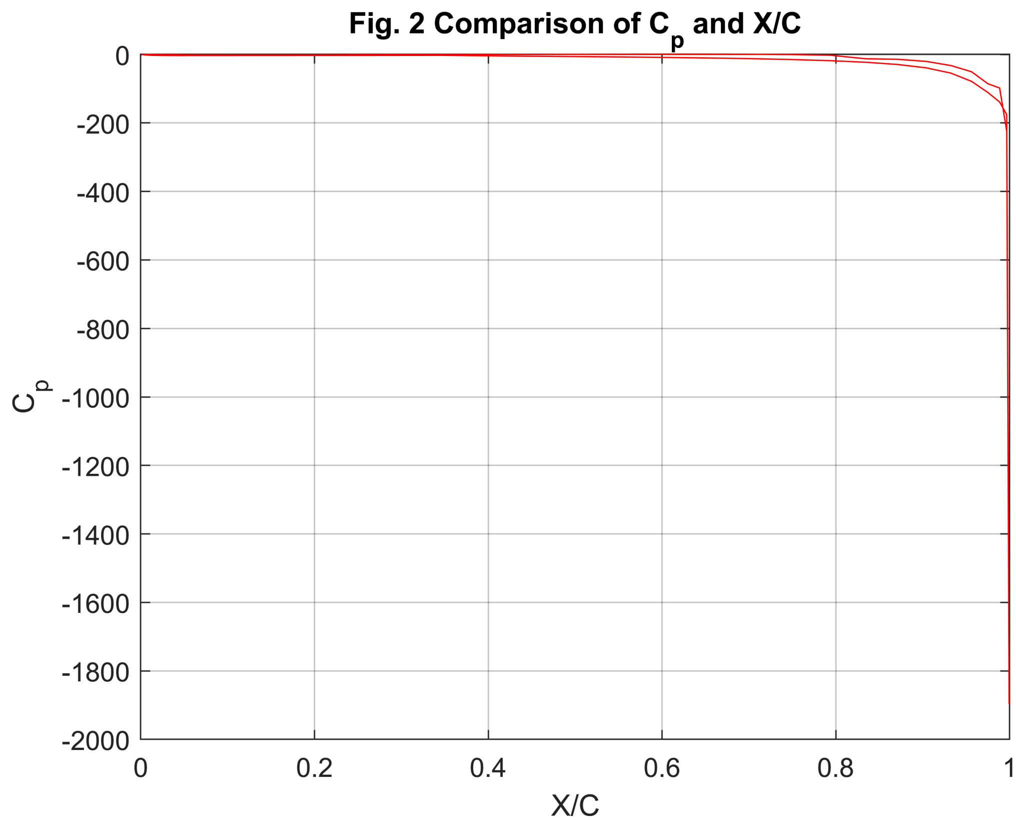
```
        E = sqrt(B-A^2);
        Sj = S(j);
        In(i,j) = C/2*log((Sj^2+2*A*Sj+B)/B) ...
            + (D-A*C)/E*(atan2((Sj+A),E)-atan2(A,E));
        Is(i,j) = (D-A*C)/2/E*log((Sj^2+2*A*Sj+B)/B) ...
            - C*(atan2((Sj+A),E)-atan2(A,E));
    end
    V_n(i,1) = V_inf*cos(beta(i));
end

coeffM = In/2/pi+eye(n)/2;     % coefficient matrix of \lambda_i
lambda = coeffM\(-V_n);        % source strengths

% check whether the sum of source strenghs is 0
fprintf('The sum of all sources is %f',dot(lambda,S));
```

```
The sum of all sources is -0.006808
```

```
Vs = abs(V_inf*sin(beta) + Is*lambda/2/pi);  % tangent velocity
Cp = 1-(Vs/V_inf).^2;                        % pressure coefficient

box on;
```

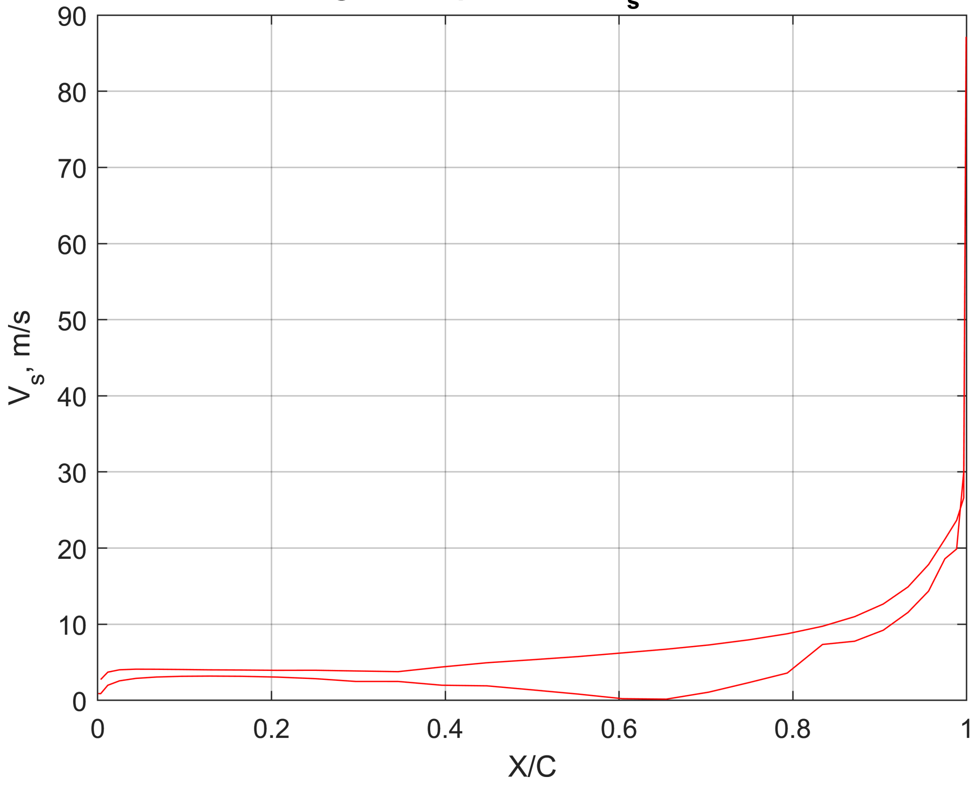



```
figure(2)   %Plotting conX vs Cp
plot(conX,Cp,'r');
grid;
title('Fig. 2 Comparison of C_p and X/C');
xlabel('X/C'); ylabel('C_p');
```




```
figure(3)   %Plotting conX Vs Vs
plot(conX, (Vs),'r');
grid;
title('Fig. 2 Comparison of V_s and X/C');
xlabel('X/C'); ylabel('V_s, m/s')
```

Fig. 2 Comparison of V_s and X/C




```
scale = 0.01;
Q = zeros(101,101);  %We will use this 2D array to save the values of potential function

%Loop to calculate the potential function at points in the space
for i=0:100
    for j=-50:50
        potential = 0;
        if 0 == inpolygon(scale*i,scale*j,X,Y)
            for k = 1:n
                %Calculating the potential function
                potential= potential + (lambda(k)/(2*pi))*log((sqrt((scale*i-conX(k))^2 + (scal
            end
        end
        Q(i+1,j+51) = potential;
    end
end

U = zeros(100, 100);        %To store the x component of the streamlines
V = zeros(100, 100);        %To store the y component of the streamlines

for i = 1:100
    for j = -49:50
        if 0 == inpolygon(scale*i,scale*j,X,Y)  %To avoid making streamlines inside the airfoil
            U(i,j+50) = V_inf + (Q(i+1, j+50) - Q(i, j+50))/scale;  %differentiating the
            V(i,j+50) = (Q(i, j+51) - Q(i, j+50))/scale;            %potential function
        else
            U(i,j+50) = 0;      %put the values inside the airflow to zero
```

```
            V(i,j+50) = 0;      %to optimize the code
        end
    end
end

figure(4)
hold on;
plot(X,Y,'g','LineWidth', 2);   %plotting the airfoil
for i = 1:100
    for j = -49:50
        if 0 == inpolygon(scale*i,scale*j,X,Y)
            v = [U(i,j+50),V(i,j+50)];
            M = 50*norm(v);
            %To make the arrows of the same size
            %Plotting arrows using the quiver function
            quiver(i*scale, j*scale, (U(i,j+50)/M), (V(i,j+50)/M),'color',[0 0 0]);
        end
    end
end
```