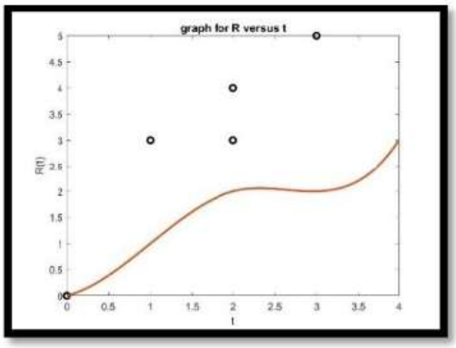
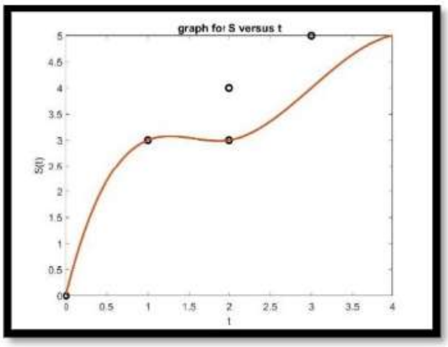
COMPUTING ASSIGNMENT 6

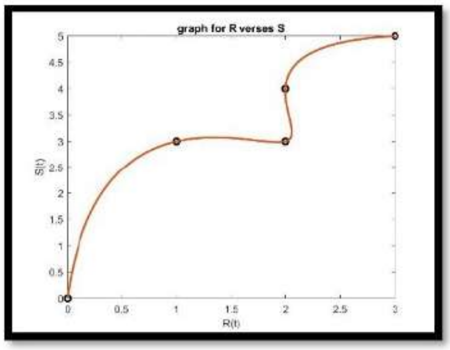
NAVJOT KAUR (301404765)

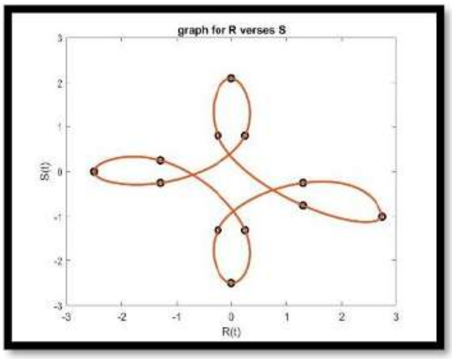
This assignment concerns with cubic spline to interpolate data that cannot be described by a single-valued function. The notion of parametric spline is used to describe such data.

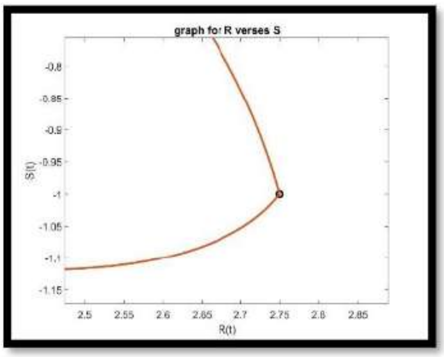
Part a. The data provided contains two same x values for different y and t making it a multiply defined function. So, instead of usual cubic splines the parametric splines are used i.e, $x = R(t)$ and $y = S(t)$ for some parameter t using MATLAB's spline and ppval function. Based on the data given the following plots are plotted where first plot is R vs t, second is S vs t, and the third is R vs S, which is the same plot as provided in the description of the assignment.





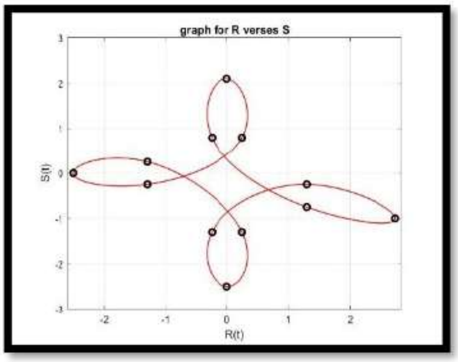


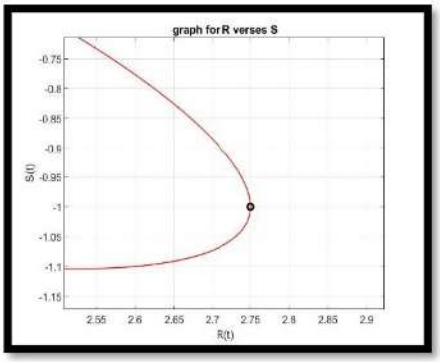


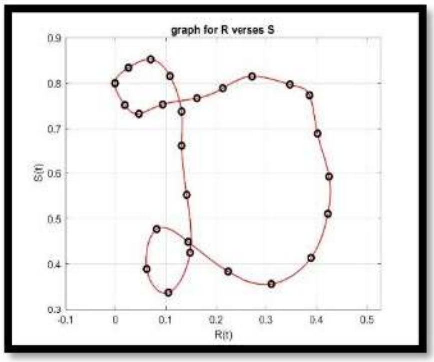


Part b. Here the four-leaf curve is interpolated using parametric splines. Using the data provided the following graph is plotted, where when zoomed into the right most leaf shows the cusp as expected because MATLAB's spline function uses not-a-knot end points to plot graphs.

Part c. Here periodic end-point conditions are used instead of not-a knot conditions to plot a four-leaf curve. The MATLAB code provided below makes use of perspline function from perspline.m, which when plots the above graph makes it smoother as shown below. The curve is smooth because persline uses periodic end point conditions. The second image justifies that the curve is indeed smooth as compared to the cusp plot in part b.







Part d. The third plot shown above is a creative version of my plot which is alphabet D. The curve crosses two time and is smooth as it has same end and starting point. This plot is created using 27 points with the help of MATLAB's ginput function which gives x and y values used to create a smooth curve using periodic end-point conditions (perspline fnction).

MATLAB CODE

```
%part a
x = [0.0, 1.0, 2.0, 2.0, 3.0];
y = [0.0, 3.0, 3.0, 4.0, 5.0];
t = [0, 1, 2, 3, 4];
tx = spline(t, x);
ty = spline(t, y);
dataPoints = linspace(0, 4, 2000);
Rt = ppval(tx, dataPoints);
St = ppval(ty, dataPoints);
plot(x, y, 'ko', dataPoints, Rt,
'LineWidth', 2);
xlabel("t"); ylabel("R(t)"); title("graph
for R versus t");
plot(x, y, 'ko', dataPoints, St,
'LineWidth', 2);
xlabel("t"); ylabel("S(t)"); title("graph
for S versus t");
plot(x, y, 'ko', Rt, St, 'LineWidth', 2);
xlabel("R(t)"); ylabel("S(t)");
title("graph for R verses S");
% part b
x1 = [2.75, 1.3, -0.25, 0.0, 0.25, -1.3,
-2.5, -1.3, 0.25, 0.0, -0.25, 1.3, 2.75];
y1 = [-1.0, -0.75, 0.8, 2.1, 0.8, -0.25,
0.0, 0.25, -1.3, -2.5, -1.3, -0.25, -
1.0];
t1 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11, 12];
tx1 = spline(t1, x1);
ty1 = spline(t1, y1);
dataPoints1 = linspace(0, 12, 2000);
Rt1 = ppval(tx1, dataPoints1);
St1 = ppval(ty1, dataPoints1);
plot(x1, y1, 'ko', Rt1, St1, 'LineWidth',
2);
xlabel("R(t)"); ylabel("S(t)");
title("graph for R verses S");
%part c
Rt2 = perspline(t1, x1);
St2 = perspline(t1, y1);
RvS = perspline(Rt2, St2);
hold on;
plot(x1, y1, 'ko', 'LineWidth', 2);
hold off;
xlabel("R(t)"); ylabel("S(t)");
title("graph for R verses S");
%part d
figure('position', get(0,'screensize'))
%largest window possible
axes('position', [0 0 1 1])
axis square % make x,y-axes equal
[x,y] = ginput; % record mouse clicks
%until 'Enter'
close % get rid of huge window
save mydatafile.mat x y % save x,y data
%points to a file
t2 = linspace(0,1,length(x));
Rt3 = perspline(t2,x');
St3 = perspline(t2,y');
RvS1 = perspline(Rt3, St3);
hold on; plot(x,y,'ko','LineWidth',2);
xlabel("R(t)"); ylabel("S(t)");
title("graph for R verses S"); hold off;
```

perspline.m

```
function [l] = perspline(x,y)
%x = [4.00, 4.35, 4.57, 4.76, 5.26,
5.88]';
%y = [4.77, 5.77, 6.57, 6.23, 4.90,
4.77]';
x = x';
y = y';
n = length(x) - 1;
l =[];

% Set up the matrix
h = diff(x);
diag0 = [1; 2*(h(1:end-1)+h(2:end));
2*h(end)];
A = spdiags([[h;0], diag0, [0;h]], [-1,
0, 1], n+1, n+1);
% Then do a little surgery on the
first/last rows ...
A(1,2)   = 0;
A(1,end) = -1;
A(end,1) = 2*h(1);
A(end,2) = h(1);
dy = diff(y);
% ... and the RHS vector
rhs = 6*[0; diff(dy./h); dy(1)/h(1)-
dy(end)/h(end)];
m = A \ rhs;     % Solve for slopes,
m_i=S''(x_i)

% Compute the cubic polynomial
coefficients
a = y;
b = dy./h - h.*m(1:end-1)/2 -
h.*diff(m)/6;
c = m(1:end-1)/2;
d = diff(m)./h/6;

% Plot each spline along with the data
for i = 1 : n
  xx = linspace(x(i), x(i+1), 100);
  yy = a(i) + b(i)*(xx-x(i)) + c(i)*(xx-
x(i)).^2 ...
       + d(i)*(xx-x(i)).^3;
  l = [l,yy];
  plot(xx, yy, 'r-')
  hold on
end
plot(x,y,'r','LineWidth',1)
hold off
set(gca, 'XLim', [min(x)-0.1, max(
x)+0.1])
xlabel('R(x)'), ylabel('S(x)')
title('R2(t) vs S2(t) (perspline)');
grid on; shg
print -djpeg 'perspline.jpg'
```