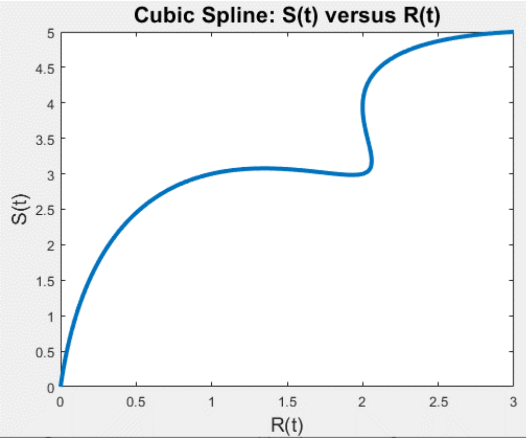
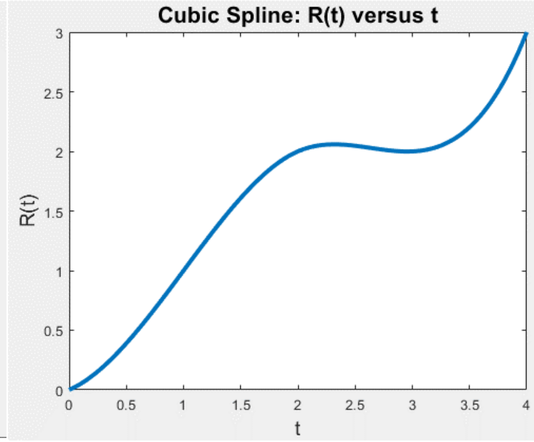
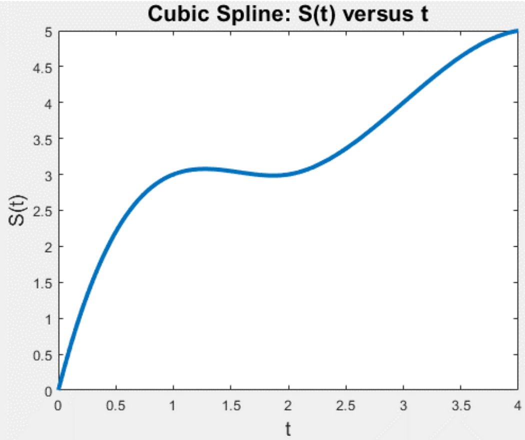
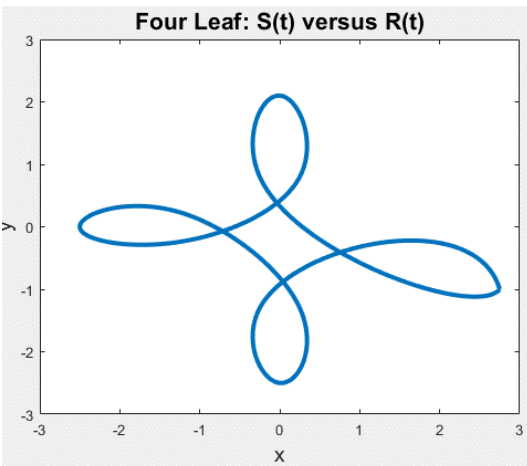
MACM 316: CA5 **AUTHOR: Andy Liu** **STUDENT ID: 301472847**

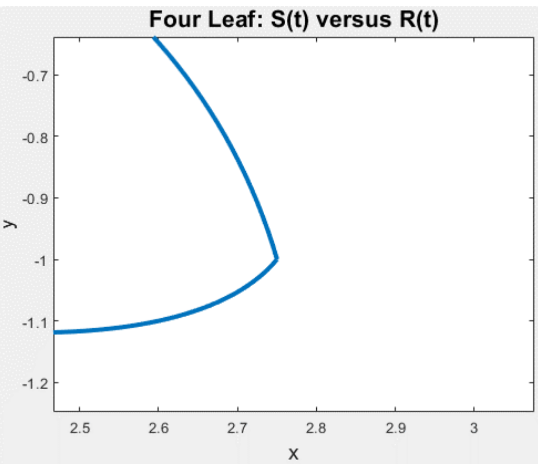


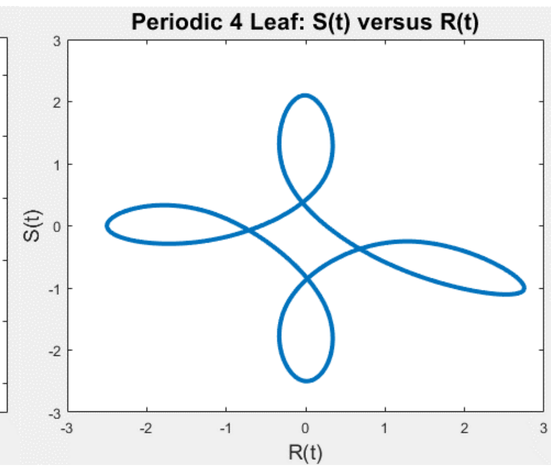


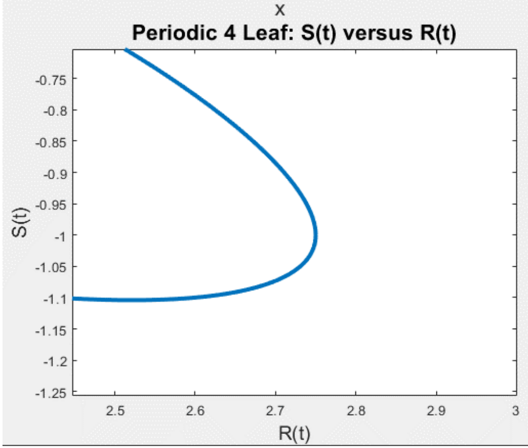


a) plots of S versus R, R versus t, S versus t are above.



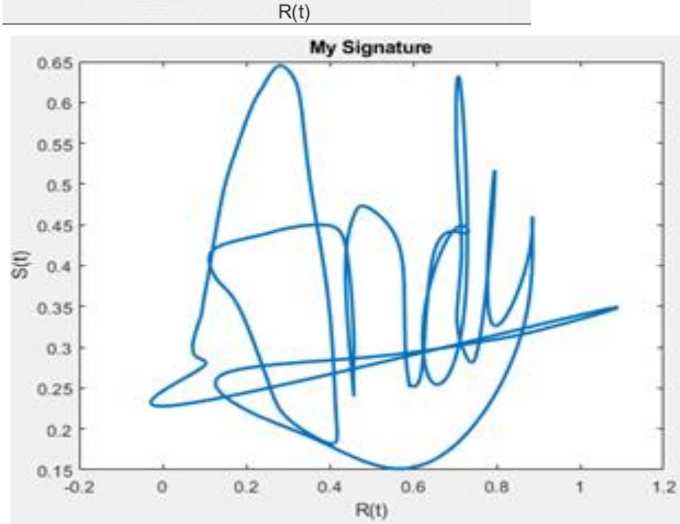






b) The above 2 plots from the top left show the four-leaf plot using a not-a-knot endpoint which results in a cusp.

c) The 3rd and 4th plots from the left use periodic endpoint conditions using a modified perspline.m function. The cusp is removed as seen in the zoomed plot and the curve on the right leaf becomes smooth.



d) The plot on the left is my drawn image. It is my signature, and it was drawn using ginput.

```
x = [0.0, 1.0, 2.0, 2.0, 3.0]; y = [0.0, 3.0, 3.0, 4.0, 5.0]; t = [0.0, 1.0, 2.0, 3.0, 4.0];
t1 = [0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 11.0, 12.0];
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];

tvalues = linspace(t(1), t(end), 1000); Leaf_tvalues = linspace(t1(1), t1(end), 1000); % T-values
Rx = spline(t, x, tvalues); Sy = spline(t, y, tvalues); % Interpolating Rx and Sy
Rx1 = spline(t1, x1, Leaf_tvalues); Sy1 = spline(t1, y1, Leaf_tvalues); % Interpolating the leaf
per_Rx = perspline(t1, x1); per_Sy = perspline(t1, y1); % Periodic interpolation
```

```
load('mysignature.mat')
n = numel(x);
drawt = 0:(n-1);
drawx = drawx.';
drawy = drawy.';
drawrx = perspline(drawt, drawx);
drawsy = perspline(drawt, drawy);
figure;
plot(drawrx, drawsy, 'LineWidth', 2);
xlabel('R(t)');
ylabel('S(t)');
title('My Signature');
```

- I removed the code to plot all these values because the code is self-explanatory and repetitive.
- It essentially repeats this block of code with different values for x and y:

```
figure; % 4 Leaf periodic
plot(per_Rx, per_Sy, 'LineWidth', 2);
xlabel('R(t)');
ylabel('S(t)');
title('Periodic 4 Leaf: S(t) versus R(t)');
```

```
% PERSPLINE: Perform cubic spline interpolation on a given set
%            of data points, using periodic end-point conditions.

% NOTE: Must have y(1)=y(end)!!  So this is a modified version
% of the data used for the other spline examples.

function s_values = perspline(x, y)
    x = x.';
    y = y.';
    % Set up the matrix
    y(1) = y(end);
    n = length(x) - 1;
    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;
    s_values = [];

    % 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;
      s_values = [s_values, yy];
    end
end
```