CS/MA 321–001 Project #4

P1.Natural Cubic Spline (4pt)

1. Complete the following program to determine the natural cubic spline interpolant $S(x)$ corresponding to the data points `tdata` and `ydata` (both are length-$(n+1)$ vectors containing respectively the knots and function values for interpolation). Write this program such that if the input `x` is a length-k vector, then the output `sx` is a vector with `sx(i)` $= S($`x(i)`$)$.

```
function sx = SplineCubic(tdata, ydata, x)
% Function sx = SplineCubic(tdata, ydata, x) evaluates the natrual cubic
% spline interpolant S(x), corresponding to interpolation points {tdata, ydata},
% at x. If the input x is a vector, then the output sx(i) = S( x(i) ).
```

2. Go through the MATLAB program `demo_spline.m` (see attached) line-by-line and try to understand what it is doing. (I may ask questions about this function during the evaluation.)

   Properly modify `demo_spline.m` so it can be used to test the `SplineCubic` from above.

3. **Runge phenomenon**. Consider to interpolate the Runge function $f(x) = \frac{1}{1+25x^2}$ with $n+1$ equally spaced nodes over the interval $[-1, 1]$. Generate the interpolation functions with

   (a) The Lagrange polynomial $P_n(x)$; (use your code from Project II)

   (b) The natural cubic spline function $S(x)$.

   Graph $f(x)$, $P_n(x)$ and $S(x)$ over the interval $[-1, 1]$. Try with $n = 5, 10, 20$ and explain what you observe (put some comments in your MATLAB code).

P2.Initial Value Problem (4pt) Consider the following initial value problems

$$\text{IVP1:} \quad x' = 2 - 2x - e^{-4t}, \quad x(0) = 1 \quad \text{with exact solution} \quad x(t) = 1 + \frac{1}{2}e^{-4t} - \frac{1}{2}e^{-2t};$$

$$\text{IVP2:} \quad x' = x + 5e^{t/2}\cos(5t) - \frac{1}{2}e^{t/2}\sin(5t), \quad x(0) = 0 \quad \text{with exact solution} \quad x(t) = e^{t/2}\sin(5t).$$

1. For each problem, apply Euler's method to find approximate solution. Try different step sizes $h = 0.1, 0.05, 0.001$ and reproduce Figure 1. (Plot the exact solution $x(t)$ over $t \in [-5, 5]$.)

2. Repeat the experiment from above with the Runge-Kutta method (of order 4). Do you obtain better approximation?

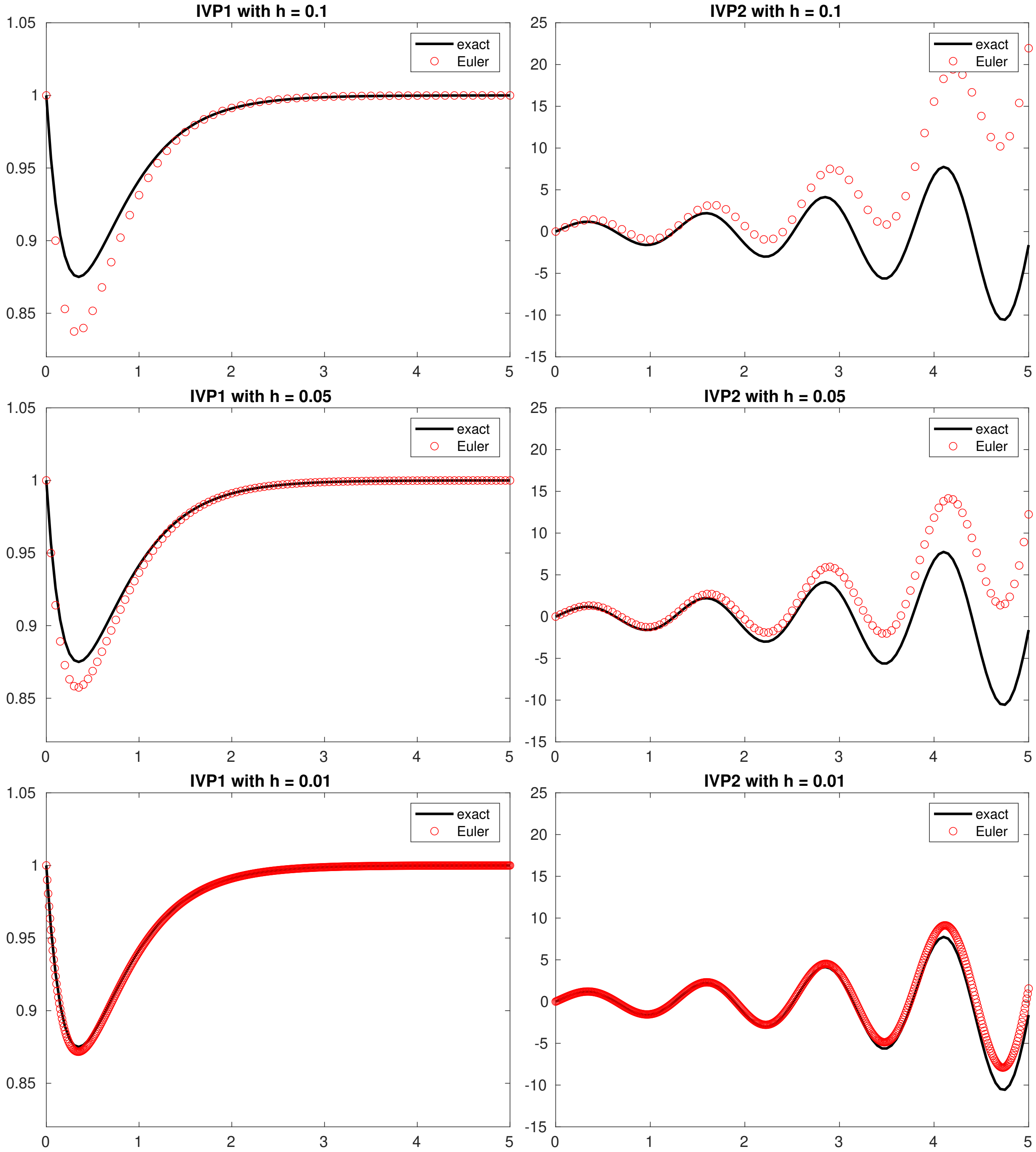

Figure 1: Initial value problem

demo_spline.m

```matlab
close all; clc;

figure;
axis([-1,1,-1,1]); hold on;
title('left click to select, right click to draw, hit 0 to return')

% Each while loop generates one curve.
while(1)
    % Part I: Select interpolation points on the x-y plane,
    % collected in the vector X and Y.
    X = []; Y = [];
    while(1)
        [x, y, flag] = ginput(1); % help ginput
        if flag == 48        % key "0" hit detected
            return;
        elseif flag ~= 1     % right click detected
            break
        end
        X = [X,x]; Y = [Y,y];    % save the clicked point in to X and Y
        plot(x,y, '.k');         % display the clicked point
        axis([-1,1,-1,1]);
    end

    % Part II: generate plane curve that goes through the selected points

    N = length(X);               % number of points
    t = linspace(0,1,N);         % parameter t
    tp = linspace(0,1, 1000);    % parameter for plotting

    xp = spline(t,X,tp);     % find spline for x
    yp = spline(t,Y,tp);     % find spline for y

    plot(xp, yp, '-b');      % plot the spline
end
```