MA 780
Homework #2
Due Feb. 17

1. Given $(x_i, f(x_i), f'(x_i))$, $i = 0, 1, \cdots, n$, $n > 0$. Let $p_{2n+1}(x)$ be the Hermite interpolation polynomial satisfying

$$p_{2n+1}(x_i) = f(x_i), \quad p'_{2n+1}(x_i) = f'(x_i), \qquad i = 0, 1, \cdots, n.$$

Show that if $f(x) \in C^{2n+2}$, then we have the following error estimate

$$R(x) = f(x) - p_{2n+1}(x) = \frac{f^{(2n+2)}(\xi(x))}{(2n+2)!}\omega^2_{n+1}(x).$$

Give a *least upper bound* for the case $n = 1$.

2. (a) Show that there is no polynomial $p_3(x)$ that interpolates

$$f(-1) = 1, \quad f'(-1) = 1, \quad f'(1) = 2, \quad f(2) = 1.$$

Hint: Use the un-determined coefficient method and show that the system of equations for the coefficients does not have a solution. Explain as much as you can.

(b) Now we add two conditions

$$f(1) = 0, \quad f''(1) = 2.$$

Use the divided difference table to find the polynomial interpolation and verify your result.

3. Let $\{x_i\}_0^n$, $x_0 = a$, $x_n = b$ be a equally spaced nodal points for the interval $[a, b]$, that is, $x_{i+1} - x_i = h = (b - a)/n$ is a constant. Assume also that $u(x) \in C^5[a, b]$ (can this be relaxed?).

(a) In the interval $[x_{i-1}, x_{i+1}]$, we can approximate $u(x)$ by a quadratic interpolation using nodal points x_{i-1}, x_i, x_{i+1}. From your interpolation function, find an approximation to $u''(x_i)$ and an error estimate. **Hint:** Use the second order divided difference.

(b) Using the formula we can get a linear system of equations to approximate the boundary value problem

$$u''(x) = f(x), \qquad a < x < b, \qquad u(a) = 0, \quad u(b) = 0,$$

as

$$\frac{U_{i-1} - 2U_i + U_{i-1}}{h^2} = f(x_i), \qquad U_0 = 0, \quad U_n = 0, \quad i = 1, 2, \cdots n - 1,$$

where $U_i \approx u(x_i)$. Write down the matrix-vector form ($AU = F$) of the linear system of equations above.

(c) Write down the component form of the SOR(ω) method for the system of equations above. Do the Jacobi, Gauss-Seidel, SOR(ω) converge?

Hint: The SOR(ω) method is $\mathbf{x}^{k+1} = (1-\omega)\mathbf{x}^k + \omega\mathbf{x}^{k+1}_{Gauss-Seidel}$, which converges for $0 < \omega < 2$ for this problem.

4. Write down the linear system of equations for the moments $S_3''(x_j) = M_j$ for the periodic cubic spline and simplify it for the equally spaced nodal points by setting $h = x_j - x_{j-1}$. Do some research about how to solve the linear system of equations.

5. (a) Derive a quadratic spline $S_2(x) \in C^1$ that satisfies

$$S_2(x_i) = f(x_i), \quad i = 0, 1, \cdots, n, \quad \text{and} \quad S_2'(x_0) = f'(x_0). \tag{1}$$

Hint: Write the quadratic spline $S_2(x)$ as

$$S_2(x) = a_i + b_i(x - x_i) + c_i(x - x_i)(x - x_{i+1}), \quad x \in [x_i,\, x_{i+1}]\,,$$

and derive a system of equations for a_i, b_i and c_i.

(b) **Extra credit:** Let $R(x) = f(x) - S_2(x)$ and $h = \max\{|x_{i+1} - x_i|\}$. Show the following error estimates if $f(x) \in C^3$.

$$\begin{aligned}
&\|R''\|_\infty \le h\|f'''\|_\infty, \quad \text{excluding the nodes,}\\
&\|R'\|_\infty \le h^2\|f'''\|_\infty/8,\\
&\|R\|_\infty \le (x_n - x_0)h^3\|f'''\|_\infty/8.
\end{aligned}$$

6. **Programming Part:**

Use *a cubic spline* (either the cardinal or B-splines, see text and pseudo-code there) assuming $S_3'(x_0) = f'(x_0)$ and $S_3'(x_n) = f'(x_n)$ to approximate functions $f(x) = \cos(10\pi x)$ and $f(x) = \frac{1}{1+25x^2}$ in HW#1.

Plot the function and your approximation on the same plot. Plot also of the errors and compare with the results from HW 1.

The project part can be done as group up to 4 people. You need to state your group in you HW. Analysis can be done individually or in group. If you can not get your code working, you can use Hermite cubic interpolation, or even piecewise linear interpolations. A couple of points may be deducted though.

7. **An extra credit project for the semester.** Group work is OK.

This is example of how to simulate mean-curvature flows using a cubic spline. Assume that we have a two-dimensional smooth and closed curve $\mathbf{X}(s,t) = (X(s,t), Y(s,t))$ whose motion satisfies $\frac{d\mathbf{X}}{dt} = \mathbf{n}f(\kappa)$, where κ is the curvature of the curve, $\mathbf{n}$ is the normal direction, for example $f(\kappa) = 1$, or $f(\kappa) = -1$, or $f(\kappa) = C\kappa$.

(a) Initially, we can use a set of discrete points (X_k, Y_k) to represent the curve at $t = 0$, $k = 1, 2, \cdots N$, $X_1 = X_{N+1}$, $Y_1 = Y_{N+1}$. We can treat the points (X_k, Y_k) as the function of the arc-length s, such that $s_0 = 0$, $s_{k+1} = s_k + \sqrt{(X_{k+1} - X_k)^2 + (Y_{k+1} - Y_k)^2}$. Thus, we have $X_k^0 \approx X(s_k, t^0)$, $Y_k \approx Y(s_k, t^0)$, where the superscript is the time level, $t^{m+1} = t^m + \Delta t$ with $t^0 = 0$.

(b) Use a cubic spline with the period boundary condition to approximate the curve to obtain $(X(s,t^m), Y(s,t^m))$ so that we can get the tangential ($\boldsymbol{\tau}$) and normal directions $\mathbf{n}$, and the curvature information, for example, $\boldsymbol{\tau} = (\frac{dX(s_k,t^k)}{ds}, \frac{dY(s_k,t^m)}{ds})/|\boldsymbol{\tau}|$, where $|\boldsymbol{\tau}| = \sqrt{(\frac{dX(s_k,t^m)}{ds})^2 + (\frac{dY(s_k,t^m)}{ds})^2}$.

(c) Evolve the boundary using the Euler's method $\mathbf{X}^{m+1} = \mathbf{X}^m + \Delta t\, \mathbf{n}^m f(\kappa^m)$.

Hint: In continuous case, this is an initial value problem

$$\frac{d\mathbf{X}(\mathbf{s},\mathbf{t})}{dt} = \mathbf{n}f(\kappa), \qquad t > 0, \qquad \mathbf{X}(\mathbf{s},\mathbf{0}) = \mathbf{X}_0(s). \tag{2}$$

We can use forward Euler method to solve the problem

$$\frac{\mathbf{X}^{k+1} - \mathbf{X}^k}{\Delta t} = \mathbf{n}^{\mathbf{k}} f(\kappa^k), \quad k = 0, 1, \cdots, \tag{3}$$

where $\mathbf{X}^k$ is the initial position of the curve. A pseudo code is given below:

```
% Set-up
    clear all; close all;
    n=80; h=2/n; x=-1:h:1; y=x;     % Domain: [-1, 1] x [-1, 1]
    n1 = n; tfinal=0.5; ds = 2*pi/n1; theta=0: ds:2*pi-ds;
    for m=1:n1;   % An initial ellipse
       X(m)=0.6*cos(theta(m)); Y(m) = 0.4*sin(theta(m));
     end
     plot(X,Y)
    axis([-1 1 -1 1]); axis('square'); hold % Visualize the initial curve

% Time loop

    dt = h*h/2; t=0;
    while t < tfinal
       % Spline interpolation to get
       for m=1: n1-1.   % Need to do it for m=n1 separately.
          dx = X(m+1)-X(m); dy = Y(m+1)-X(m);  ds = sqrt( dx*dx + dy*dy + 1e-15);
          dx = dx/ds dy = dy/ds;  % The tangent direction
          nx = -dy;  ny=dx;  % The normal direction
          X1(m) = X(m) - dt*nx; Y1(m) = Y(m) - dt*ny;   % f=-1;
       end
       m = n1;
          %%%%% Fill in

       t = t + dt;
       plot(X1,Y1)
       X=X1; Y= Y1;   % Overwrite the old boundary data
    end
```