CS4210 Assignment 2 Due: 9/19/14 (Fri) at 6pm

You must work either on your own or with one partner. You may discuss background issues and general solution strategies with others, but the solutions you submit must be the work of just you (and your partner). If you work with a partner, you and your partner must first register as a group in CMS and then submit your work as a group. Each problem is worth 5 points. One point may be deducted for poor style.

Topics: Polynomial interpolation

1 A Quintic Polynomial Interpolation Problem

Ordinarily, we could interpolate the six points $(d_1, y_1), \ldots, (d_6, y_6)$ with the quintic polynomial

$$p(d) = a_1 + a_2 d + a_3 d^2 + a_4 d^3 + a_5 d^4 + a_6 d^5.$$

(See lecture code `InterpV`.) In this problem you solve a variant of this problem. In particular, you are to implement the following function:

```
  function a = Quintic(d,y)
% d is a column 4-vector with d(1) < d(2) < d(3) < d(4)
% y is a column 4-vector
% a is a column 6-vector with the property that if
%
%    p(d) = a(1) + a(2)d + a(3)d^2 + a(4)d^3 + a(5)d^4 + a(6)d^5
%
% then  p(d(i)) = y(i) (i=1:4), p'(d(1)) = p'(d(4)), and p"(d(1)) = p"(d(4)).
```

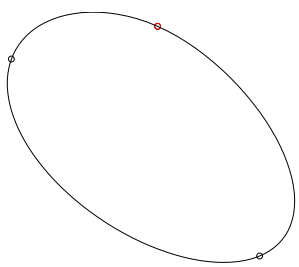
In other words, the quintic interpolates at four points and the two extra degrees of freedom are used to ensure that the slopes at the endpoints match and that the second derivatives at the endpoints match. Submit `Quintic` to CMS. A test script `P1` is available.

2 Odometer-Based Interpolation

Suppose we are given three distinct points in the plane (u_1, v_1), (u_2, v_2), (u_3, v_3) and define the following "odometer" values:

$$\begin{aligned} d_1 &= 0 \\ d_2 &= d_1 + \sqrt{(u_2 - u_1)^2 + (v_2 - v_1)^2} \\ d_3 &= d_2 + \sqrt{(u_3 - u_2)^2 + (v_3 - v_2)^2} \\ d_4 &= d_3 + \sqrt{(u_1 - u_3)^2 + (v_1 - v_3)^2} \end{aligned}$$

(Picture yourself driving around the triangle defined by the three points, looking at the odometer every time you pass through a vertex.) We wish to construct a closed curve $(p_x(d), p_y(d))$ that passes through the three points and is smooth, e.g.,



Here are the rules:

- $p_x(d)$ and $p_y(d)$ are quintic polynomials in d.
- $(p_x(d_1), p_y(d_1)) = (u_1, v_1)$
- $(p_x(d_2), p_y(d_2)) = (u_2, v_2)$
- $(p_x(d_3), p_y(d_3)) = (u_3, v_3)$
- $(p_x(d_4), p_y(d_4)) = (u_1, v_1)$
- $p'_x(d_1) = p'_x(d_4)$ and $p_x"(d_1) = p_x"(d_4)$
- $p'_y(d_1) = p'_y(d_4)$ and $p_y"(d_1) = p_y"(d_4)$

The last four conditions ensure that everything is smooth through the second derivative when we "drive through" the first vertex (u_1, v_1). Implement a function

```
function [ax,ay] = QuinticCurve(u,v)
```

The inputs `u` and `v` should be column 3-vectors that specify the three points. The coefficients of the fitting quintics should be returned in `ax` and `ay`:

$$\begin{aligned} p_x(d) &= ax_1 + ax_2 d + ax_3 d^2 + ax_4 d^3 + ax_5 d^4 + ax_6 d^5 \\ p_y(d) &= ay_1 + ay_2 d + ay_3 d^2 + ay_4 d^3 + ay_5 d^4 + ay_6 d^5 \end{aligned}$$

A six-by-six linear system is involved and you may use the \ operator. Your implementation should also display the three points (highlighting (u_1, v_1) in red) and the interpolating curve. Use `axis equal off`. Submit `QuinticCurve` to CMS. A test script `P2` is provided.

3 Evaluating Cubics via Differences

Refer to page 17 in CVL Chapter 2 where "differences" are discussed. We change the notation a bit for simplicity. We are given a function $f(x)$ and points $x_i = x_1 + (i-1)h$, let $f_i = f(x_i)$. The k-th order difference of f at $x_1, \ldots, x_k$ is defined by

$$f\{x_i, \ldots, x_{i+k}\} = \begin{cases} f_i & \text{if } k = 0 \\ f\{x_{i+1}, \ldots, x_{i+k}\} - f\{x_i, \ldots, x_{i+k-1}\} & \text{if } k > 1 \end{cases}.$$

For example,

0th Order	1st Order	2nd Order	3rd Order	4th Order
f_1				
f_2	$f_2 - f_1$			
f_3	$f_3 - f_2$	$f_3 - 2f_2 + f_1$		
f_4	$f_4 - f_3$	$f_4 - 2f_3 + f_2$	$f_4 - 3f_3 + 3f_2 - f_1$	
f_5	$f_5 - f_4$	$f_5 - 2f_4 + f_3$	$f_5 - 3f_4 + 3f_3 - f_2$	$f_5 - 4f_4 + 6f_3 - 4f_2 + f_1$
f_6	$f_6 - f_5$	$f_6 - 2f_5 + f_4$	$f_6 - 3f_5 + 3f_4 - f_3$	$f_6 - 4f_5 + 6f_4 - 4f_3 + f_2$

Suppose f is a cubic. Since a k-th order difference is a multiple of a k-th order divided difference, and since a kth order divided difference is "secretly" a k-th order derivative, it follows that all the 4th order differences will be zero. Let's take a look at the above difference array with the example $f(x) = 2x^3 - x^2 + x - 3$ with $x_i = i$:

```
f(1) =   -1
f(2) =   11     12
f(3) =   45     34   22
f(4) =  113     68   34   12
f(5) =  227    114   46   12   0
f(6) =  399    172   58   12   0
```

Notice that the 3rd order differences are constant. This gives us a very fast way to figure out $f(7)$. At the start we have

```
f(1) =   -1
f(2) =   11      12
f(3) =   45      34   22
f(4) =  113      68   34   12
f(5) =  227     114   46   12   0
f(6) =  399     172   58   12   0
f(7) =   ?       ??  ???   12   0
```

Since `???-58 = 12` we have

```
f(1) =   -1
f(2) =   11      12
f(3) =   45      34   22
f(4) =  113      68   34   12
f(5) =  227     114   46   12   0
f(6) =  399     172   58   12   0
f(7) =   ?       ??   70   12   0
```

Since `??  - 172 = 70` we have

```
f(1) =   -1
f(2) =   11      12
f(3) =   45      34   22
f(4) =  113      68   34   12
f(5) =  227     114   46   12   0
f(6) =  399     172   58   12   0
f(7) =   ?      242   60   12   0
```

Since `?  - 399 = 242` we have

```
f(1) =   -1
f(2) =   11      12
f(3) =   45      34   22
f(4) =  113      68   34   12
f(5) =  227     114   46   12   0
f(6) =  399     172   58   12   0
f(7) =  641     232   60   12   0
```

It checks out: $f(7) = 2 * 7^3 - 7^2 + 7 - 3 = 686 - 49 + 7 - 3 = 641$. Using this very fast evaluation idea, implement the following function

```
  function q = CubicValues(p,n)
% p is a column 4-vector with the property that
%            p(1) = c(x0)
%            p(2) = c(x0+h)
%            p(3) = c(x0+2h)
%            p(4) = c(x0+3h)
%
% where c(x) is some cubic polynomial, x0 is real and h>0 is real.
%
% q is a column n-vector (n>=4) with the property that
%            q(i) = c(x0+(i-1)h), i=1:n
```

Submit `CubicValues` to CMS. A test script `P3` is available.

4 Cubic Hermite Interpolation on a Line Segment

Read about cubic hermite interpolation in CVL §3.2.1 and consider the following implementation:

```
  function [a,b,c,d] = HCubic(xL,yL,sL,xR,yR,sR)
% Cubic Hermite interpolation
% (xL,yL,sL) and (xR,yR,sR) are x-y-slope triplets with xL and xR distinct.
% a,b,c,d are real numbers with the property that if
%           p(z) = a + b(z-xL) + c(z-xL)^2 + d(z-xL)^2(z-xR)
% then p(xL)=yL, p'(xL)=sL, p(xR)=yR, p'(xR)=sR.
a = yL; b = sL; delx = xR - xL;
yp = (yR - yL)/delx;
c  = (yp - sL)/delx;
d  = (sL - 2*yp + sR)/(delx*delx);
```

In this problem you are to use the cubic Hermite interpolation idea to estimate the value of a function $f(x, y)$ at the point

$$(x_*, y_*) \;=\; (x_1 + t_*(x_2 - x_1), y_1 + t_*(y_2 - y_1))$$

given that you know the value of f and its partial derivatives f_x and f_y at both $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$. In particular, implement the following function:

```
  function [zStar,fxStar,fyStar] = HConLineSeg(P,F,tStar)
% P is a length-2 cell array of structures that identify 2 distinct points in
%    the plane: (P{1}.x,P{1}.y) and (P{2}.x,P{2}.y).
% F is a length-2 cell array of structures that encode the value of a
%    function f(x,y) at these points together with the value of the x and y
%    partial derivatives, e.g., F{k}.fvalue, F{k}.fx, and F{k}.fy , k=1:2
% tStar is a scalar that satisfies 0<=tStar<=1.
% zStar = c(tStar) where c(t) is the piecewise cubic hermite interpolant of
%    the function f(P{1}.x+t*(P{2}.x-P{1}.x),P{1}.y+t*(P{2}.y-P{1}.y)) at
%    t=0 and t=1.
% fxStar and fyStar are estimates of f's partial derivatives in the x and y directions
%    at tStar
```

Submit `HConLineSeg` to CMS. A test script `P4` is provided.

5 Hermite Cubic Interpolation on a Triangle

Refer to problem P2.4.6 in CVL which outlines a framework for estimating the value of a function $f(x, y)$ given that its value is known on the vertices. This problem is the same thing except that you also know the value of f's partial derivatives at each of the vertices. Implement the following function and submit it to CMS.

```
  function z = HConTriangle(P,F,alpha)
% P is a length-3 cell array of structures that identify 3 distinct points in
%    the plane: (P{1}.x,P{1}.y), (P{2}.x,P{2}.y), and (P{3}.x,P{3}.y).
% F is a length-3 cell array of structures that encode the value of a
%    function f(x,y) at these points together with the value of the x and y
%    partial derivatives, e.g., F{k}.fvalue, F{k}.fx, and F{k}.fy , k=1:3
% alpha is a length-3 vector of nonnegative numbers that sum to one.
% z is an estimate of f(xtilde,ytilde) where
%           xtilde = alpha(1)*P{1}.x + alpha(2)*P{2}.x + alpha(3)*P{3}.x
%           ytilde = alpha(1)*P{1}.y + alpha(2)*P{2}.y + alpha(3)*P{3}.y
%
```

The score on this problem will depend upon the quality of the approximation. A test script `P5` is provided.