University of Colorado
Department of Computer Science

Numerical Computation CSCI 3656

Spring 2016

Problem Set 7 Solutions

1. *[10 pts]* Use Newton's divided differences to fit a parabola to the last three points in your data set from PS6. See pp. 142 for the formula.
$f[1990 \quad 2000] = \frac{369.52-354.35}{2000-1990}$;
$f[2000 \quad 2010] = \frac{389.85-369.52}{2010-2000}$;
$f[1990 \quad 2000 \quad 2010] = \frac{f[2000 \quad 2010]-f[1990 \quad 2000]}{2010-1990}$.
$P_2(x) = 354.35 + f[1990 \quad 2000](x-1990) + f[1990 \quad 2000 \quad 2010](x-1990)(x-2000)$

   Do you get the same polynomial as you did in problem 3 of that problem set? Yes.

   Should you? Yes; there is only one parabola $P_3(x)$ that fits a given set of (linearly independent) points. Lagrange and Newton's divided differences are two different recipes for finding the coefficients of that polynomial.

2. *[16 pts]* Problem 2 on page 156 of the textbook. (Note: there is no Theorem 3.3 in the book. He meant Theorem 3.4.)

   (a) The 2^{nd} degree Lagrangian interpolation polynomial:
   $P_2(x) = -\frac{ln2}{2}(x^2-5x+4) + \frac{ln4}{6}(x^2-3x+2)$

   (b) $P_2(3) = 1.1552$

   (c) $f(x) - P_2(x) = \frac{(x-1)(x-2)(x-4)}{3!} * f^{(2)}(c)$, where $1 < c < 4$
   $= \frac{(x-1)(x-2)(x-4)}{3!} * \frac{1}{c}$
   $= \frac{((3)-1)((3)-2)((3)-4)}{3!} * \frac{1}{c}$
   $= -\frac{1}{3c}$
   Taking the absolute value, and using $c = 1$, we have: $|f(x) - P_2(x)| \le \frac{1}{3}$.

   (d) $|f(x) - P_2(x)| = |1.0986 - 1.1552| = 0.0566$.
   We can verify that our error adheres to the bound: $0.0566 \le \frac{1}{3}$.

3. *[24 pts]* In your favorite programming language, implement a program that takes a bunch of $(x, f(x))$ pairs, computes the corresponding cubic natural splines, and draws the curve defined by those equations. (It should also plot the points themselves, so that you can see whether the spline curve goes through them.) Test your code out on the following data set. Please turn in a plot of the results as well as a copy of your code.

x	$f(x)$
1	1
2	3
3	2
4	1
5	2
6	4
7	5

Here is some Matlab code that solves for the coefficients of a natural cubic spline and plots the spline curve:

```
function [coeffs] = cubicSpline(xys)
    if(size(xys,1) < 3)
       coeffs = -1;
       return
    end
    n = size(xys,1);          % n-1 equations
    coeffs = zeros(n-1,4);    % a,b,c,d coefficients for spline equations
    del = zeros(n-1,1);       % temporary variable
    Delta = zeros(n-1,1);     % temporary variable
    M = eye(n);               % Tri-diagonal matrix
    b_vec = zeros(n,1);       % used to solve for c's
    % a, temporary variables
    for i = 1:1:(n-1)
        coeffs(i,1) = xys(i,2);
        del(i) = xys(i+1,1) - xys(i,1);
        Delta(i) = xys(i+1,2) - xys(i,2);
    end
    for i = 2:1:n-1
        M(i,i-1) = del(i-1);
        M(i,i) = 2*(del(i-1) + del(i));
        M(i,i+1) = del(i);
        b_vec(i,1) = 3 * ( Delta(i)/del(i) - Delta(i-1)/del(i-1) );
    end
    % c
    c = linsolve(M,b_vec);
    coeffs(:,3) = c(1:n-1,1);
    % b, d
    for i = 1:1:(n-1)
        coeffs(i,2) = (Delta(i)/del(i)) - (del(i)/3) * (2*c(i) + c(i+1));
```

```
        coeffs(i,4) = (c(i+1) - c(i)) / (3*del(i));
    end
end
```

Here is a plot generated using the splines' coefficients: