CS412 Spring Semester 2011

Homework Assignment #3

Due Thursday March 10th 2011, in class

Sum of all problems : 120%, Maximum possible score : 100%.

1. [40%] In this problem you will write a `MATLAB` function that computes the coefficients of a cubic Lagrange polynomial. You will also be introduced to the `MATLAB` functions `poly`, `polyval` and `error`.

   Consider $N+1$ values along the x-axis, denoted by $x_0, x_1, x_2, \ldots, x_{N-1}, x_N$. Each of the *Lagrange polynomials* $l_0(x), l_1(x), \ldots, l_N(x)$ is a polynomial of degree N, defined in such a way that it satisfies the following property:

   $$l_i(x_j) = \begin{cases} 1, & j = i \\ 0, & j \neq i \end{cases} \tag{1}$$

   In class, we derived a formula for each $l_i(x)$, as follows:

   $$l_i(x) = \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_N)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_N)} = \frac{\prod_{j\neq i}(x-x_j)}{\prod_{j\neq i}(x_i-x_j)}$$

   For this problem, you must generate each l_i in the more standard form:

   $$l_i(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

   Write a `MATLAB` function `lagrange_cubic(x0,x1,x2,x3,k)` which returns a row vector `p=[`a_3`,`a_2`,`a_1`,`a_0`]` with the coefficients of the cubic Lagrange polynomial $l_{\mathtt{k}}(x)$ defined over the points `x0`,`x1`,`x2`,`x3`. If the parameter `k` is not an integer between 0 and 3, the function should abort with the error message `'Index out of bounds'`, using the `MATLAB` command `error`.

   Turn in your code, and also test your implementation by using `x0=1,x1=2, x2=3,x3=5`, and report the coefficients you generated for `k=0,1,2,3`.

   Hint: Each l_i can be expressed compactly as $l_i(x) = q_i(x)/q_i(x_i)$, where $q_i(x) = \prod_{j\neq i}(x - x_j)$. The `MATLAB` function `poly(u)` takes as argument a vector `u=[`u_1`,`u_2`,`$\ldots$`,`u_m`]` and returns another vector `c=[`c_m`,`$\ldots$`,`c_1`,`c_0`]` with the coefficients of the polynomial:

   $$c_mx^m + \cdots + c_1x + c_0 = (x - u_1)(x - u_2)\cdots(x - u_m)$$

   You can use function `poly` to generate the coefficients of $q_i(x)$ defined above. Once these coefficients have been computed (and stored in a vector, say `c`) the function `polyval(c,`w`)` can be used to evaluate this polynomial at an arbitrary point w. Thus, by calling `polyval(c,`x_i`)` you can also compute the quantity $q_i(x_i)$.

2. [40%] Using the function `lagrange_cubic` from Problem 1, write a function `lagrange_interpolation(x,y)` which takes the following arguments:

   - `x` is a row vector containing the 4 values `x=[`x_0`,`x_1`,`x_2`,`x_3`]`.
   - `y` is a row vector containing the 4 values `y=[`y_0`,`y_1`,`y_2`,`y_3`]`.

   This function should implement the Lagrange interpolation method to construct a cubic polynomial $P(x) = a_3x^3+a_2x^2+a_1x+a_0$ which interpolates the four data points (x_0, y_0),(x_1, y_1),(x_2, y_2) and (x_3, y_3). Function `lagrange_interpolation` should return a row vector `p=[`a_3`,`a_2`,`a_1`,`a_0`]` containing the coefficients of the interpolant $P(x)$.

   Turn in your code and test your implementation by computing the coefficients of the cubic polynomial that interpolates the four data points:

   $$(-1, -10) \quad (0, -4) \quad (2, 2) \quad (3, 14)$$

   Additionally, the result of your function `lagrange_interpolation(x,y)` should be identical to the `MATLAB` built-in function `polyfit(x,y,3)` which performs the same task, but using the Vandermonde matrix approach. Check that the two methods produce the same result for the points given.

3. [40%] In this last problem, you will construct a piecewise cubic polynomial function that interpolates the N data points $(x_1, y_1), (x_1, y_1), \dots, (x_N, y_N)$.

   In each subinterval $I_k = [x_k, x_{k+1}]$ we define our interpolant as a cubic polynomial $s_k(x) = a_3^{(k)}x^3 + a_2^{(k)}x^2 + a_1^{(k)}x + a_0^{(k)}$. The cubic polynomials s_k are constructed such that:

   - For $k = 2, 3, \dots, n\text{-}2$, $s_k(x)$ should interpolate points $(x_{k\text{-}1}, y_{k\text{-}1})$, (x_k, y_k), (x_{k+1}, y_{k+1}) and (x_{k+2}, y_{k+2}).
   - $s_1(x)$ should interpolate (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4).
   - $s_{N\text{-}1}(x)$ should interpolate $(x_{N\text{-}3}, y_{N\text{-}3})$, $(x_{N\text{-}2}, y_{N\text{-}2})$, $(x_{N\text{-}1}, y_{N\text{-}1})$ and (x_N, y_N).

   Implement a function `piecewise_cubic(x,y)` with arguments:

   - `x` is a row vector containing the N values `x=[`x_1`,`x_2`,...,`x_N`]`.
   - `y` is a row vector containing the N values `y=[`y_1`,`y_2`,...,`y_N`]`.

   The function should return a $(N-1) \times 4$ matrix `M` with the coefficient of each s_k on the respective row:

   $$\mathtt{M} = \begin{bmatrix} a_3^{(1)} & a_2^{(1)} & a_1^{(1)} & a_0^{(1)} \\ a_3^{(2)} & a_2^{(2)} & a_1^{(2)} & a_0^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ a_3^{(N-1)} & a_2^{(N-1)} & a_1^{(N-1)} & a_0^{(N-1)} \end{bmatrix}$$

Turn in your code, and also test your implementation by running the commands on the `MATLAB` script file provided to you in

`http://pages.cs.wisc.edu/~cs412-1/hw/hw3.m`

Note: You are free to use either function `lagrange_interpolation(x,y)` from Problem 2, or the built-in function `polyfit(x,y,3)` in your implementation.