Numerical Mathematics:
Homework 3

Problem 1

Write a function called:

`[ y_int ] = lagrangeinterp( x_int, X, Y )`

that evaluates a Lagrange interpolation formula for a given set of points. Here, `X` and `Y` are column vectors whose entries contain the coordinates x_1, x_2,...x_n and values $f(x_1)$, $f(x_2)$,...$f(x_n)$, respectively. The output `y_int` are the values of the interpolating polynomial at the interpolation points `x_int`, which are again both column vectors.

To code things efficiently, we are going to re-write the Lagrange interpolation formula. Let $l(x) := (x - x_1)(x - x_2)\cdots(x - x_n)$, so that:

$$L_{n-1,k(x)} = l(x)\frac{w_k}{x - x_k}, \quad w_k := \frac{1}{\prod_{j=1, j\neq k}^{n} (x_k - x_j)}.$$

By doing so, the interpolating polynomial can be re-expressed as:

$$P_{n-1}(x) = \sum_{k=1}^{n} f(x_k) L_{n-1,k}(x) = l(x) \sum_{k=1}^{n} f(x_k) \frac{w_k}{x - x_k}$$

Explain in your own words why the formula above is computationally more efficient than classical one derived in lecture.

Note: Your matlab routine must take special care of the case when one of the interpolating points coincide with x_1, x_2,...x_n to avoid division by zero!

Solution. Shown below is the code.

```
function [ y_int ] = lagrangeinterp( x_int, X, Y )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

% number of points:
n = length(X);

% Pre-compute weights:
W = X*ones(1,n) - ones(n,1)*transpose(X)+eye(n);
W = 1 ./ prod(W,2);


% Initialize array of zeros for output
y_int = zeros(size(x_int));

% evaluate function
for k=1:length(x_int)

    % check if interpolation point is not equal to one of the data points
    % to prevent division by zero.
    m = find(X == x_int(k));
    if isempty(m)

        diff = x_int(k)-X;
        y_int(k) = prod(diff) * transpose( 1./diff ) * (W .* Y);

```

```
    else

        % if it is, the value is simply:
        y_int(k) = Y(m);

    end

end


end
```

□

Problem 2

Consider the function:

$$f(x) = \frac{1}{1+25x^2}$$

We are going to approximate this function with an interpolating polynomial. The interpolation nodes are to our discretion. For instance, we may choose equidistant points or the Chebyshev nodes as discussed in class. Write a function called: `[ ] = cheb_vs_equispaced( n , type )` which generates a figure with the follow items all in one plot:

1. A graph of the original function,
2. Markers at the interpolation points
3. A graph of interpolating polynomial.

The argument `n` describes the degree of the interpolating polynomial (hence `n`+1 nodes). The argument `type` specifiies whether one desires an equidistant or Chebhyshev interpolation. Entering `type='equispaced'` should yield an equispaced interpolation, entering `type='Chebyshev'` should yield a Chebyshev interpolation. Describe what you observe from the plots. Which type of interpolation does a better job? Does this make sense?

Solution. Shown below is the code. Interpolation at equispaced points seem to diverge for this function, and for the Chebyshev nodes convergence occurs. That Chebhyshev polynomials perform better certainly make sense, given the discussions in the theory class.

```
function [ ] = cheb_vs_equispaced( n , type )



if strcmp(type,'equispaced')

    % equispaced nodes
    X = transpose(-1:(2/(n)):1);
    Y = 1 ./ (1+25 .* X.^2 );



elseif strcmp(type,'Chebyshev')

    % Chebyshev nodes
    X = cos((2*transpose(0:n) + 1)*pi/(2*n+2));
    Y = 1 ./ (1+25 .* X.^2 );
```

```
    length(X)
end


% interpolating polynomial
x_int = transpose(-1:(1/1000):1);
y_int = lagrangeinterp( x_int, X, Y );

% true function values
y_true = 1 ./ (1+25 .* x_int.^2 );



figure('position',[100 100 850 400])
hold on
plot(X,Y,'ko')
axis([-1 1 -0.25 1])
plot(x_int,y_true,'k-.')
plot(x_int,y_int,'r')
```

□

Problem 3

Let us generalize the concept of polynomial interpolation a bit. Suppose that we are given a continuously differentiable function on an interval $[a, b]$. Assume that, in addition to the function values $f(x_1)$, $f(x_2)$, ... ,$f(x_n)$ at x_1, x_2, ... x_n, the derivative values $f'(x_1)$, $f'(x_2)$, ... ,$f'(x_n)$ are also provided. Let:

$$L_{n-1,k}(x) = \prod_{j=1, j\neq k}^{n} \frac{x - x_j}{x_k - x_j}$$

denote j'th Lagrange interpolation. Show that the $2n-1$ degree polynomial:

$$H_{2n-1}(x) = \sum_{k=1}^{n} f(x_k) H_{n-1,k}(x) + f'(x_k) \hat{H}_{n-1,k}(x)$$

$$H_{n-1,k}(x) = [1 - 2(x - x_k) L'_{n-1,k}(x_k)]\, L^2_{n-1,k}(x), \qquad \hat{H}_{n-1,k}(x) = (x - x_k)\, L^2_{n-1,k}(x)$$

uniquely interpolates the function values and its derivatives at the respective . To show uniqueness, you may want to make use of the fundamental theorem of algebra. Rolle's theorem may also be handy.

Solution. Let us first show that $H_{2n-1}(x)$ interpolates the function values $f(x_1)$, $f(x_2)$, ... ,$f(x_n)$. Observe that

$$H_{n-1,k}(x_i) = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases},$$

and

$$\hat{H}_{n-1,k}(x_i) = 0 \quad \text{for } i = 1, \ldots, n.$$

This gives $H_{2n-1}(x_i) = f(x_i)$

Next, we will show that the derivative $H'_{2n-1}(x)$ interpolates the derivative values $f'(x_1)$, $f'(x_2)$, ... ,$f'(x_n)$. We must look at:

$$H'_{2n-1}(x) = \sum_{k=1}^{n} f(x_k) H'_{n-1,k}(x) + f'(x_k) \hat{H}'_{n-1,k}(x)$$

We must find expressions for $H'_{n-1,k}(x)$ and $\hat{H}'_{n-1,k}(x)$. Elementary calculus shows:

$$H'_{n-1,k}(x) = -2L'_{n-1,k}(x_k)L^2_{n-1,k}(x) + [1 - 2(x - x_k)L'_{n-1,k}(x_k)]\, 2L_{n-1,k}(x)L'_{n-1,k}(x)$$

$$\hat{H}'_{n-1,k}(x) = L^2_{n-1,k}(x) + (x - x_k)2L_{n-1,k}(x)L'_{n-1,k}(x)$$

We have that:

$$H'_{n-1,k}(x_i) = 0 \quad \text{for } i = 1, \ldots, n,$$

and

$$\hat{H}'_{n-1,k}(x_i) = \begin{cases} 1 & i = k \\ 0 & i \neq k \end{cases}.$$

This verifies $H'_{2n-1}(x_i) = f'(x_i)$.

To show uniqueness. Suppose there exists another polynomial $G_{2n-1}(x)$ which does the job. Then consider the difference: $d(x) = H_{2n-1}(x) - G_{2n-1}(x)$. Note that $d(x)$ is at most of degree $2n - 1$. For simplicity assume that the points $x_1 < x_2 < \ldots < x_n$ are ordered. Observe that $d(x)$ has roots in x_1, x_2, ..., x_n (that are n roots). Applying Rolle's theorem $n - 1$ times, we know that $d'(x)$ has (in addition to x_1, x_2, ..., x_n) roots at:

$$x_1 < h_1 < x_2 < h_2 < x_3 < \ldots < h_{n-1} < x_n$$

Hence $d'(x)$ has $n + (n - 1) = 2n - 1$ roots. The degree of $d'(x)$ is however $2n - 2$. This immediately contradicts the fundamental theorem of algebra. □