ESC 407
Computational Methods in Engineering Science
Homework 4
Interpolation and Numerical Differentiation

Due Date: Friday, October 30 at 11:59 p.m.

Problem 1

Write a MATLAB function `neville` that implements Neville's method to recursively generate all possible interpolating polynomials for adjacent nodes up to order n for $n+1$ distinct points, $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$. The function should take as inputs

(1) row or column vector x containing the $n+1$ abscissas $x_0, x_1, \ldots, x_n$,

(2) row or column vector y containing $y_0, y_1, \ldots, y_n$, which becomes the first column of the Q array,

(3) the point x_e at which each polynomial is evaluated.

The output should be the lower triangular Q array containing all the polynomials evaluated at x_e. Test your function on $x_0 = 2$, $x_1 = 2.2$, and $x_2 = 2.3$ and the y_i's generated using $y_i = \ln x_i$. Use $x_e = 2.1$. The Q array for this data should be

```
Q =
   0.693147180559945                    0                    0
   0.788457360364270    0.740802270462108                    0
   0.832909122935104    0.744005597793436    0.741870046239217
```

Now write a second MATLAB function `neville_f` that calls `neville` repeatedly until a desired approximate relative error is achieved in approximating a function $f(x)$ at the point x_e using successively higher order interpolating polynomials generated by Neville's method. The function will start by applying Neville's method to the points $(a, f(a)), (b, f(b))$, where, generally, $a < x_e < b$. It will generate new x_i's by dividing the interval $[a, b]$ by increasingly higher integers. So the first application of Neville's method is dividing by $n = 1$, the second will be by $n = 2$ and so on. After each Q array is generating, the approximate relative error is checked using

$$\text{approx rel error} = \frac{Q(n+1, n+1) - Q(n, n)}{Q(n+1, n+1)}.$$

The function `neville_f` should take as inputs

(1) the function f as a function handle,

(2) the point x_e where you want to get the approximate value of the function,

(3) the initial interval $[a, b]$ as a row vector,

(4) the desired relative error,

(5) the maximum allowed times the interval $[a, b]$ can be divided.

The function should output

(1) the final approximation of $f(x_e)$,

(2) the approximate relative error,

(3) the number of iterations required to achieve the desired error or an message stating that the maximum number of iterations was exceeded.

Defaults should be set if the desired error or the maximum number of iterations is not set by the user.

Finally, write a MATLAB script called `neville_run` that calls `neville_f` for the following problems with an approximate relative error of 10^{-9}.

(a) $f(x) = x^3 e^x \cos(10x) + 1, \quad [a, b] = [-3, 5], \quad x_e = 2.2,$

(b) $f(x) = \tan\left[\cos\left(\dfrac{\sqrt{5 + \sin x}}{1 + x^3}\right)\right], \quad [a, b] = [-1.5, 30], \quad x_e = 20.$

In each case report

- the final approximation of $f(x_e)$,
- the approximate relative error,
- the true relative error,
- the required number of iterations.

Problem 2

Write a MATLAB function `diff_limit` that uses the centered difference formula of order $O(h^2)$ to approximate the derivative of $f(x)$ at x_0. Start with a step-size of one and repeatedly divide it by 10 until the desired relative error tolerance is obtained or the maximum number of iterations is reached. That is, you will generate the sequence $\{D_k\}$ approximating $f'(x_0)$, where

$$f'(x_0) \approx D_k = \frac{f\left(x_0 + h/10^k\right) - f\left(x_0 - h/10^k\right)}{2\left(h/10^k\right)} \qquad \text{for } k = 0, \ldots, n.$$

Since you do not have an estimate of the relative error after just one estimate of the derivative, you will have to generate two estimates of the derivative before you can start your `while` loop to continue to generate the above sequence.

The function should take as inputs the

(1) function f as a function handle,

(2) the point x_0,

(3) the desired relative error for convergence,

(4) the maximum number of iterations,

(5) the initial step size h.

The function should set reasonable defaults if the relative error, the maximum number of iterations, and/or the initial step size are omitted. The outputs should be

(1) the number of iterations n,

(2) an $n \times 3$ array, the first column containing the values of h, the second containing the sequence of approximations to $f'(x_0)$, and the third containing the approximate relative error for each iteration.

These should all be neatly written to the Command Window.

Write a script called `diff_limit_run` that calls your function to approximate the derivative of the following functions at the indicated points

(a) $f(x) = e^{x^2} \sin\left[\cos\left(\frac{1}{x^3}\right)\right]$ at $x = 1/\sqrt{2}$

(b) $f(x) = \tan\left[\cos\left(\frac{\sqrt{5} + \sin x}{1 + x^3}\right)\right]$ at $x = \frac{1 + \sqrt{5}}{3}$,

(c) $f(x) = x^{x^{x^{x^{x^{x}}}}}$ at $x = 0.0001$ and at $x = 1.5$.[*]

All answers should be accurate to 9 decimal places. You may have to tweak h and the maximum number of iterations to get the desired results.

Problem 3

A projectile is tracked using video equipment that measures its x and y positions as a function of time at 50 times per second. Data for the *perfect* measurements of a projectile can be downloaded from Canvas in a file named `projectile_data`. The first column is the time of each measurement, the second is the corresponding x position, and the third is the corresponding y position.[†]

(a) Plot the trajectory of the projectile.

(b) Write a MATLAB script that uses the $O(h^2)$ forward and backward finite difference approximations, and the $O(h^4)$ centered finite difference approximations for the first and second derivatives, to approximate $\dot{x}$, $\dot{y}$, $\ddot{x}$, and $\ddot{y}$ at every measurement time.

(c) Generate two plots: the first should be $\dot{x}$ and $\dot{y}$ versus time, and the second should be $\ddot{x}$ and $\ddot{y}$ versus time.[‡] Interpret the plots.

[*] $\frac{d}{dx} x^{x^{x^{x^{x^{x}}}}} = x^{x^{x^{x^{x^{x}}}}} \left[x^{-1+x^{x^{x^{x}}}} + x^{x^{x^{x^{x}}}} \left(x^{-1+x^{x^{x}}} + x^{x^{x^{x}}} \ln x \left\{x^{-1+x^{x}} + x^{x^{x}} \ln x \left[x^{-1+x} + x^{x} \ln x \,(1 + \ln x)\right]\right\}\right)\right].$

[†] This data file and be imported into MATLAB using the `dlmread` command.

[‡] Use MATLAB's `figure` command.

(d) Report the average of $\ddot{x}$ and $\ddot{y}$ and comment on the results.[*]

(e) Real measurements will have some error and noise in them. I have simulated that kind of data in the file `projectile_data_noise` that you can download from Canvas. Use that data to generate all the same results you generated for the data in `projectile_data`. Comment on the differences between the different results you obtain from your analysis of the two sets of data.

[*]Use MATLAB's `mean` command.