CS1133, Spr 2018, HW 2 (Lect 4, Pts=5+20+20)

Due February 6, 2018 (before 11:00 pm)

Problem 1 [5 pts]

Here we recompute first derivative of the same function using the same parameters p and q and at the same location x. However, instead of performing a single calculation using a single step size with each run, we use a vector of step sizes so that all the calculation that we need are performed all at once without having to run the program multiple number of times. The vector of step sizes must have values that are decreasing in values.

Perform the calculations for both methods (central finite differencing and the complex step methods) all in one program. Produce two separate vectors of computed first derivatives of the function corresponding to the step sizes used. Collect the corresponding relative errors in the form of a 2D array named `RELERRORS`. Column 1 of `RELERRORS` must contain the values of step sizes used. Columns 2 and 3 store the relative errors obtained for the two methods.

Display, using format `shorte`, the values of `RELERRORS`. Before displaying that array, display an appropriate heading in the form of a string that labels the three separate columns. These are the only display your program should produce. Make sure that the Matlab editor detects no errors or warnings in your program (no red marks).

Do you see a trend? Describe your findings semi-qualitatively. Explain them. Write your comments in the form of Matlab comments at the end of your program.

Problem 2 [20 pts]

Use the following block of statements to generate randomly the original weights and their new weights in kilograms for a certain number of people:

```
nPeople = 10;
avgWt = 75;     % average weight in Kg
sdWt = 12;      % standard deviation of weight
Wts = avgWt + sdWt .*  randn(nPeople,1);
rangeWt = 3;
NewWts = Wts + rangeWt * (2 * rand(nPeople,1) - 1);
```

Their original weights and new weights are stored in variables named `Wts` and `NewWts`, respectively.

The problem here is to find the person with the greatest weight change. For a given person, the weight change can be a weight gain (given by a positive value) or a weight loss (given by a negative value). Find that person and that person's weight change (with the correct sign to indicate a weight gain or loss).

Also find the person with the smallest relative weight change. Find that person and that person's relative weight change.

Note that if x is the original weight and y the new weight, then the weight change is given by $y - x$, and the relative weight change by $(y - x)/x$. Thus the change (relative or not) is positive for a weight gain and negative for a weight loss.

Of course you cannot use the data as given in the above table. You must use the data that your own program generates.

Here is an example of the output results that the program produces:

```
     Wts      NewWts  ChangeWts RelChangeWts
   89.9094   90.0362    0.1268    0.0014
   62.1996   60.5891   -1.6104   -0.0259
   86.2047   86.1381   -0.0666   -0.0008
   79.2039   79.9482    0.7444    0.0094
   74.6519   75.7267    1.0748    0.0144
   77.1894   76.5625   -0.6269   -0.0081
   56.2193   55.4239   -0.7954   -0.0141
   73.9855   76.9134    2.9279    0.0396
   94.2474   91.4738   -2.7736   -0.0294
   76.1802   78.4912    2.3110    0.0303

The person with the largest weight change is person number
     8

whose weight change is
    2.9279

The person with the smallest relative weight change is person number
     3

whose relative weight change is
  -7.7274e-04
```

Your output display must follow the format as shown in this example.

Problem 3 [20 pts]

In this problem, we want to compute the value of a given function, $f(x)$, at various value of x. There are n such values varying from `xStart` to `xEnd`, in equal step. The function is defined, for magnitudes of x greater than one, in the form of an infinite series:

$$f(x) = \sum_{k=1}^{\infty} \frac{2^{k-1}}{x^{2^{k-1}} + 1}.$$

Since numerically we can only sum a finite series, we need to truncate it to m (another variable). Thus, the results can only be approximate:

$$f(x) \approx \sum_{k=1}^{m} \frac{2^{k-1}}{x^{2^{k-1}} + 1}.$$

We expect the results to be more and more accurate with increasing m.

It turns out that the exact function is given by $1/(x-1)$, for magnitude of x greater than one.

Here using $n = 11$, `nStart` $= 1.03$, `xEnd` $= 1.33$, and $m = 7$, compute the values of the above function. Concatenate all your results to form a 2D array RESULTS, where column 1, 2, 3, and 4 contain the value of x, the approximate function, the exact function and the relative error. Finally, display the values in `RESULTS` in the long fixed-point format. Write the program so that the Matlab editor does not indicate any warning or error. Be mindful of efficiency issues.