MMF 2021F ASSIGNMENT #1 Due: 1 October 2019.

This assignment is due at the **start** of the lecture on Tuesday, 1 October 2019.

For the questions that require you to write a MatLab program, hand-in the program and its output as well as any written answers requested in the question. Your program and its output, as well as your written answers, will be marked. Your program should be "well-commented" and you should use good programming style, etc. When first learning to program in MatLab, students often produce long, messy output. Try to format the output from your program so that it is easy for your TA to read and to understand your results. For example, if you are asked to print a table of values, as in Question 2, print them as a table that fits neatly on one page. To this end, you might find it helpful to read "A short description of fprintf" on the course webpage

`http://www.cs.toronto.edu/~krj/courses/2021/`

Marks will be awarded for well-formatted, easy-to-read output.

Also, your TAs will appreciate your using a word processor to write the answers to questions (or parts of questions) that do not require a program. If you do write those answers by hand, make sure that they are easy to read.

In the next tutorial on Thursday, September 26, you can ask your TA about the paragraphs above if you have any questions about what is meant there.

1. [10 marks]

   Assume that you are using a computer that conforms to the IEEE floating-point standard. In particular, assume that, when you add any two normalized floating-point numbers, x and y, the computed result, $\mathrm{fl}(x+y)$, satisfies

   $$\mathrm{fl}(x+y) = (x+y)(1+\delta)$$

   for some δ satisfying $|\delta| \leq \frac{1}{2}\,\epsilon_{\mathrm{mach}}$, assuming that there is no overflow or underflow when you perform the addition, where ϵ_{mach} is the *machine epsilon* for the floating-point number system.

   Show that, if you add three non-negative floating-point numbers, a, b, c, (i.e., $a \geq 0$, $b \geq 0$ and $c \geq 0$), then the result satisfies

   $$\mathrm{fl}((a+b)+c) = (a+b+c)(1+\tilde{\delta})$$

   for some $\tilde{\delta}$ for which $|\tilde{\delta}|$ is small, assuming that there is no overflow or underflow in the computation. Try to find as tight a bound as you can on $|\tilde{\delta}|$ in terms of ϵ_{mach}.

   (A similar result holds if you add three non-positive floating-point numbers, a, b, c, (i.e., $a \leq 0$, $b \leq 0$ and $c \leq 0$), but you don't have to prove this result for this question.)

2. [25 marks: the marks for each part are indicated below]

(a) [5 marks]

Write a MatLab function exp1(x) that computes an approximation to e^x by summing the terms in the Taylor series

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

from left to right until the sum does not change.

Hand in your MatLab function exp1(x).

(b) [5 marks]

Test your function exp1(x) by computing exp1(x) for $x = -25, -24, -23, \ldots, 25$ (i.e., for x = $-25 : 25$ in MatLab notation).

For each value of x, compute the relative error

$$\frac{\text{exp1}(x) - \exp(x)}{\exp(x)}$$

where exp(x) is the MatLab function that approximates e^x, and print both x and the relative error.

Hand in a table showing your computed results as well as the MatLab program that computes the table.

The exp(x) in the formula above is the MatLab approximation to e^x. It is very accurate for all $x \in [-25, 25]$. So, you can consider it to be the "true value".

(c) [10 marks]

Explain your numerical results. In particular, state for which values of x your function exp1(x) is accurate and explain why it is accurate for those values of x. Similarly, state for which values of x your function exp1(x) is inaccurate and explain why it is inaccurate for those values of x. You can consider exp1(x) to be of intermediate accuracy (i.e., neither very accurate nor very inaccurate) for some values of x if you want. State what you mean by accurate, inaccurate and intermediate.

You need to say more than there are rounding and truncation errors in the computation, since there are rounding and truncation errors in both the accurate and inaccurate approximations to e^x. You need to explain why your function exp1(x) is not seriously affected by the rounding and truncation errors in the cases for which it works well and why the rounding and truncation errors cause a severe loss of accuracy in the cases for which it does not work well.

(d) [5 marks]

Make a small change to your function exp1(x) so that it computes an accurate approximation to e^x for all $x = -25 : 25$. Call your new function exp2(x). (Alternatively, you can call exp1(x) from within exp2(x).)

Hand in your MatLab function exp2(x) as well as a table (similar to the one produced in part (b)) showing the relative errors for exp2(x) for $x = -25 : 25$.

3. [10 marks]

Suppose an option depends on the value of an underlying, S, and time, t. Most of the rest of this course will focus on numerical methods for computing the price, $V(S,t)$, of such an option. However, these numerical methods nearly always have an associated error, ϵ. So, assume that what we really compute is an approximation, $\hat{V}(S,t)$, to $V(S,t)$ and that $\hat{V}(S,t) - V(S,t) = \epsilon$, where ϵ is small, but depends on both S and t. Moreover, we don't know ϵ exactly, but we often have a rough idea about its magnitude. Furthermore, the more computational work that we are willing to do in the numerical method, the smaller in magnitude we can make ϵ.

Suppose that, in addition to the price, $V(S,t)$, we also want to compute an estimate of $\partial V(S,t)/\partial S$. (We may need this for hedging and risk management.) Since we don't have an explicit formula for $V(S,t)$, we can't compute this partial derivative directly. However, we can estimate it by either

$$\frac{\partial V(S,t)}{\partial S} \approx \frac{\hat{V}(S+h,t) - \hat{V}(S,t)}{h}$$

or

$$\frac{\partial V(S,t)}{\partial S} \approx \frac{\hat{V}(S+h,t) - \hat{V}(S-h,t)}{2h}$$

where $|h|$ is a small nonzero number.

Note that, the error, $\hat{V}(S+h,t) - V(S+h,t) = \epsilon_{1,h}$, that you get when you compute $\hat{V}(S+h,t)$ normally depends on h and is usually different from the error, $\hat{V}(S,t) - V(S,t) = \epsilon_2$, that you get when you compute $\hat{V}(S,t)$ in the first approximation to $\partial V(S,t)/\partial S$ above. Similarly for the second approximation. However, assume that $|\epsilon_{1,h} - \epsilon_2| \leq \epsilon$ and that you can control the size of ϵ (at least roughly). However, the smaller you make ϵ, the more costly the computation becomes.

For each approximation listed above for $\partial V(S,t)/\partial S$, discuss how you would choose ϵ and h to get an approximation to $\partial V(S,t)/\partial S$ with an absolute error that is less than about 0.01 in magnitude. State if you have to make any other assumptions about any other quantities.

Try to choose ϵ to be as large as possible, while still meeting the accuracy requirement, since, as noted above, the smaller you make ϵ, the more costly the computation becomes.