Assignment 1: Fun with Roundoff Error

ACM 106a: Introductory Methods of Computational Mathematics (Fall 2013)

Due date: Tuesday, October 8, 2013

Instructions:

- Assignments are due at the **start** of class (1pm) on the due date.
- Write your name and ID# clearly on all pages, and underline your last name.
- **Matlab files:** please submit a single zip/rar/etc file with file name in the format **Lastname_Firstname_ID#_A1** to **homework.acm106a@gmail.com** that decompresses to a single folder of the same name containing the following:
  - A single thoroughly commented Matlab script file containing all commands used for each assigned programming/simulation problem. File names should have format **Lastname_Firstname_ID#_A1_P#** :

    e.g. **Ames_Brendan_12345678_A1_P1** for problem 1.
  - All Matlab functions used to perform any assigned programming/simulation problems with appropriate file names. Please add a comment to the beginning of each file with the format **Lastname_Firstname_ID#_A1_P#**
  - A diary file of your session for each programming/simulation problem with file name in the format **Lastname_Firstname_ID#_A1_P#_DIARY.txt**. Please also submit a hard copy of the diary and any relevant derivations, pseudocode, etc. you may want considered for partial credit with your submitted solution sets at the beginning of class.

1 Evaluation of negative exponentials

In this exercise, we consider evaluation of the function $x \mapsto e^{-x}$.

(a) Recall that e^x has Taylor series expansion

$$e^x = \sum_{n=0}^{d} \frac{x^n}{n!} + O(x^{d+1}).$$

Write a Matlab function to compute the approximation of e^{-x} given by the truncated Taylor series

$$e^{-x} = \sum_{n=0}^{d} \frac{(-x)^n}{n!}$$

by naively taking the sum of each term in this expansion (i.e., your code should compute x, $x^2/2$, etc. and take their sum). Use your function with $d = 10, 100, 200, 1000$, to evaluate e^{-x} at $x = 1, 5, 10, 15, 20$. Compare your obtained approximate values to those returned by the command `exp(-x)` (use `format long e` to ensure values are reported as floating point decimal numbers with 15 digit mantissas). How many correct digits of e^{-x} do you obtain? Estimate as a function of d, i.e., in the form $O(f(d))$, how many floating point operations (flops) are needed to compute e^{-x} in this manner and report the actual time (using `tic/toc`) needed to compute your approximations. Explain carefully the observed phenomena.

(b) Hopefully, Part (a) has convinced you that this naive approach to computing e^{-x} is misguided. Next, design an algorithm to compute the truncated Taylor series of e^{-x} without explicitly computing each term in the series or the coefficient $1/n!$. As before, evaluate your function with $d = 10, 100, 200, 1000$, and $x = 1, 5, 10, 15, 20$, and compare your obtained values with the "true" value given by Matlab. How many digits of accuracy do you obtain? Explain why this is the case.

Give an estimate on the number of flops needed by your algorithm as a function of d and the actual time needed by your algorithm. Is your method faster/slower than that in Part (a)?

Hint: the algorithm should vaguely resemble Horner's method for polynomial evaluation.

(c) Finally, we consider evaluating e^{-x} using the identity $e^{-x} = 1/e^x$. Compute e^{-x} using this formula and each of your functions for evaluating the truncated Taylor series of e^x from the previous parts with each $d = 10, 100, 200, 1000$ and $x = 1, 5, 10, 15, 20$.

Again, compare the obtained values (in terms of digits of accuracy) to the value given by `exp(-x)` and explain the observed phenomena. Compare the actual time needed for each method to that in Parts (a) and (b).

2 The Quadratic Formula

Recall that the roots of the quadratic polynomial $ax^2 + 2bx + c = 0$, $a \neq 0$ are given by

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - ac}}{a}.$$

(a) Write a Matlab function to calculate the root

$$x_{+} = \frac{-b + \sqrt{b^2 - ac}}{a}.$$

Use your function to estimate the roots of the polynomials with coefficients

$$(a, b, c) = (1, 10^k, 1), \quad k = 1, 3, 5, 7, 9.$$

Compute the value $p(x_+)$ of the polynomial $p(x) = ax^2 + 2bx + c$ at the obtained root x_+. What happens to the approximate root and the polynomial value as $b \to \infty$ (with $a = c = 1$)? Why?

(b) Modify the quadratic formula to compute x_+ more accurately for large values of b and implement as a function in Matlab. Explain carefully why your modifications should lead to a better approximation of the root when $b^2 >>> ac$. Test your algorithm using the coefficients $(a, b, c) = (1, 10^k, 1)$, $k = 1, 3, 5, 7, 9$, and compare to the results of part (a).

Hint: your modification should compute the roots with as little round off error as possible.