

This assignment is due in your tutorial on 26 October 2007.

1. [10 marks: 5 marks for each part]

Consider the function $f(x) = x^{1/4}$ for real positive x .

- (a) Is this function well-conditioned or ill-conditioned in a relative sense with respect to small relative changes in the value of the input argument x ?

Justify your answer.

- (b) Write a little MatLab program to verify your predictions from part (a).

For several real positive values of x , make a small relative perturbation of x to \hat{x} and compute $y = f(x)$ and $\hat{y} = f(\hat{x})$. Print, x , \hat{x} , y , \hat{y} and the associated relative errors in $(x - \hat{x})/x$ and $(y - \hat{y})/y$.

Hand in your MatLab program and its output.

Also include a brief explanation of why you believe your computational results from part (b) support your theoretical predictions from part (a).

2. [10 marks: 5 marks for each part]

Throughout this question, assume that you are working with a 12-decimal-digit floating-point number system (i.e., $\beta = 10$ and $p = 12$) with $L = -100$ and $U = +100$ that uses the *round-to-nearest* rounding rule for all arithmetic operations.

Also, assume that x is a normalized floating-point number in this 12-decimal-digit floating-point number system satisfying $-1 < x < 1$.

Consider the two mathematically equal expressions

$$(a) \quad \frac{1}{1-x^2} - \frac{1}{1+x^2} \qquad (b) \quad \frac{2x^2}{(1-x)(1+x)(1+x^2)}.$$

- (a) Show that one of the expressions always gives a very accurate answer in a relative error sense. That is, the relative error associated with the computed value is at most a few multiples of machine epsilon.

- (b) Give an example which illustrates that the other expression can give a computed value that is much less accurate in a relative error sense. That is, the relative error associated with the computed value is much larger than a few multiples of machine epsilon.

3. [10 marks: 5 marks for each part]

For a series of n real values x_1, x_2, \dots, x_n , you can compute the mean by

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

and the standard deviation by

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2} \quad (1)$$

- (a) Give an example that shows that underflow in the formula (1) may result in a computed value that is very far from the true value in a relative error sense.
- (b) How can you reformulate (1) so that it never overflows and, even if it does underflow, underflow is not a serious problem.
4. [6 marks]

List at least three distinct ways in which the evaluation of the quadratic formula

$$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

for the roots of the quadratic equation $ax^2 + bx + c = 0$ may suffer significant numerical difficulties in floating-point arithmetic.

Assume a, b, c are normalized floating-point numbers and $b^2 - 4ac \geq 0$ in both exact and floating-point arithmetic, so that you don't have to consider complex numbers.

5. [15 marks: 5 marks for each part]

Consider the integral

$$I_n = \int_0^1 x^n e^{x-1} dx \quad (2)$$

where n is a non-negative integer. Note that $x^n e^{x-1} > 0$ for all non-negative integers n and for all $x \in (0, 1)$. Therefore, $I_n > 0$ for all non-negative integers n . Moreover, $x^n e^{x-1} - x^{n+1} e^{x-1} = x^n(1-x)e^{x-1} > 0$ for all non-negative integers n and for all $x \in (0, 1)$. Therefore, $I_n > I_{n+1}$ for all non-negative integers n . That is, the sequence I_0, I_1, I_2, \dots is positive and monotonically decreasing.

One way to compute I_n for any non-negative integer n is as follows. Note that

$$I_0 = \int_0^1 e^{x-1} dx = [e^{x-1}]_0^1 = 1 - e^{-1} = 1 - 1/e \quad (3)$$

Moreover, for $n \geq 1$, integrating (2) by parts leads to

$$I_n = \int_0^1 x^n e^{x-1} dx = [x^n e^{x-1}]_0^1 - \int_0^1 n x^{n-1} e^{x-1} dx = 1 - n I_{n-1} \quad (4)$$

Therefore, we can use the recurrence

$$\begin{aligned} I_0 &= 1 - 1/e \\ I_n &= 1 - n I_{n-1} \quad \text{for } n \geq 1 \end{aligned} \quad (5)$$

to evaluate I_n for any non-negative integer n .

- (a) Write a MatLab program that uses the recurrence (5) to compute and print I_n for $n = 0, 1, 2, \dots, 25$.

Hand in your MatLab program and its output.

[Before you print the values I_n for $n = 0, 1, 2, \dots, 25$, you might want to execute the MatLab statement “format short e”. The default format is not suitable for printing I_n for $n = 0, 1, 2, \dots, 25$.]

- (b) Your program should print reasonable values of I_n for $n = 0, 1, 2, \dots, 15$ or so, but the values are clearly wrong for $n = 20, 21, \dots, 25$, since the computed I_n values are not positive and monotonically decreasing, as they should be.

Explain why your program computes such inaccurate values for I_n for $n = 20, 21, \dots, 25$.

In this explanation, it is not sufficient to say that there is round-off error in the computation. Although this is true and should play a part in your explanation, there is round-off error in almost all floating-point computations and most of them produce accurate results. You need to explain why the round-off error produces such bad results in this case.

- (c) Can you find a way to re-arrange the recurrence

$$I_n = 1 - nI_{n-1}$$

and start it from a different initial value so that your new recurrence computes accurate values for I_n for $n = 0, 1, 2, \dots, 25$?

Explain why you believe your new method produces accurate results.

Write a little MatLab program that uses your new method to compute and print I_n for $n = 0, 1, 2, \dots, 25$.

Hand in your program and its output.