Numerical Methods I, Fall 2014
Assignment I: Numerical Computing

Aleksandar Donev
Courant Institute, NYU, donev@courant.nyu.edu

September 4th, 2014
Due: September 19th, 2014

For the purposes of grading the maximum number of points is considered to be 100 points.

1 [10 points] Floating-Point Exceptions

Using MATLAB, do some simple calculations that lead to exceptions and report what you get [2pt per item]:

- Generate an overflow other than division by zero, in both single and double precision.
- Divide a normal number by zero and infinity and see if the answer makes sense to you. How about dividing 0 by 0?
- Generate a NaN and then perform some arithmetic operations between a normal number and a NaN. What do you get?
- Perform a comparison (e.g., $x < y$, $x > y$, $x == y$) between infinities, $NaNs$, and normal numbers to determine if and how these are ordered.
- Generate an IEEE floating-point signed positive and negative zero and take their square roots. Can you detect any difference between $+0$ and -0 in MATLAB?

2 [15 points] Numerical Differentiation

Repeat the calculation presented in class for the finite-difference approximation to the first derivative for the centered approximation to the second-order derivative:

$$f''(x = x_0) \approx \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2}$$

2.1 [5pts] Numerical Errors

Consider a simple function such as $f(x) = \sin(x)$ and $x_0 = \pi/4$ and calculate the above finite differences for several h on a logarithmic scale (say $h = 2^{-m}$ for $m = 1, 2, \cdots$) and compare to the known derivative, using both single and double precision. For what h can you get the most accurate answer?

2.2 [10pts] Optimal h

Obtain an estimate of the *truncation error* in the centered difference formula by performing a Taylor series expansion of $f(x_0 + h)$ around x_0. Also estimate what the *roundoff error* is due to cancellation of digits in the differencing. At some h, the combined error should be smallest (optimal, which usually happens when the errors are approximately equal in magnitude). Estimate this h and compare to the numerical observations.

3 [25 points] Computing π

There are many methods to compute many digits of π, and lots of them suffer from numerical accuracy problems. Here is one of them due to Archimedes: Start with $t_0 = 1/\sqrt{3}$ and then iterate

$$t_{i+1} = \frac{\sqrt{1 + t_i^2} - 1}{t_i} \tag{1}$$

and for large i you can get a good approximation $\pi \approx 6 \cdot 2^i \cdot t_i$.

3.1 [15pts] Sources of Error

[5pts] Do this calculation with both single and double precision, and report how many digits of accuracy you get and after how many iterations (Note: $\pi = 3.14159265358979323846264338 3\cdots$ and MATLAB has a built-in constant pi), accompanied with some plots of the convergence.

[10pts] Estimate the relative error due to roundoff and explain when it is large and why. What kind of error dominates the numerical estimate of π for small i and what kind of error dominates for large i?

3.2 [10 pts] The Fix

[5pts] Find a way to rewrite the iteration (1) so that you avoid cancellation errors in the numerator and get much better accuracy and repeat the calculation.

[5pts] How many digits of accuracy can you get with the improved formula? How large does i need to be to achieve the highest possible accuracy, and why?

4 [30 points] Stability and Error Propagation

[From Dahlquist & Bjorck, also discussed in Lecture 1]

Consider error propagation in evaluating

$$y_n = \int_0^1 \frac{x^n}{x+5} dx$$

based on the identity (you may want to derive this yourself)

$$y_n + 5y_{n-1} = n^{-1}.$$

Since $y_n < y_{n-1}$, we have that

$$6y_n < y_n + 5y_{n-1} = n^{-1} < 6y_{n-1},$$

$$0 < y_n < \frac{1}{6n} < y_{n-1},$$

so for large n we have tight bounds

$$\frac{1}{6(n+1)} < y_n < \frac{1}{6n} \tag{2}$$

Hint: Maple tells us that $y_{14} = 0.01122918662647553$, $y_{15} = 0.01052073353428904$ and $y_{16} = 0.00989633232855484$.

4.1 [15pts] Forward iteration

[5pts] Calculate y_n for $n = 1, 2, \cdots$ using the forward iteration $y_n = n^{-1} - 5y_{n-1}$, starting from $y_0 = \ln(1.2)$, using both single and double precision. Plot the results together with the bounds (2) [choose your axes wisely, e.g., plot ny_n instead of y_n].

[10pts] Report the result for the largest $n = n_{max}$ where you actually trust the answer to 4 significant digits, and explain your choices and observations the best you can.

4.2 [15pts] Backward iteration

[5pts] Now start with $n = 2n_{max}$ and repeat the calculation going backward, $y_{n-1} = (5n)^{-1} - y_n/5$, starting with both the lower and upper bound for y_n in (2), at least for double precision. Plot the results on top of the plot from the forward iteration.

[10pts] How many digits do you trust in the answer for $y_{n_{max}}$ now and why?

5 [20 points] Beneficial Cancellation: Computing $\ln(1+x)$ for small x

[Due to William Kahan / David Goldberg]

Introduction: When calculating $\ln(1+x)$ for $0 < x \ll 1$ that is close to machine precision, the argument $1+x$ has a large roundoff error and the relative error in the result is large. The MATLAB function $log1p$ is designed so as to avoid this problem and give an accurate result.

5.1 [5pts] Numerical Troubles

Calculate $log(1+x)$ directly using MATLAB, for logarithmically-spaced (small) values of x, and comment the relative error compared to the built-in function $log1p$ (a picture is worth a thousand words!).
Hint: You may choose to plot $\frac{\ln(1+x)}{x}$ instead to make the behavior near zero easier to study.

5.2 [10pts] Taylor Approximation

Compare the built-in function to the truncated Taylor series of $\ln(1+x)$ for small x. Use only the first couple or first few terms in the Taylor series. Instead of using the built-in function, try the following alternative: Use the naive direct calculation of $log(1+x)$ for $x > x_0$ and the Taylor series for $x \leq x_0$. Try to find an x_0 that is optimal, that is, one for which the worst relative accuracy over the interval $0 < x < 1$ is smallest.

5.3 [5pts] IEEE Magic

A wise person that knows a lot about IEEE arithmetic has shown that using the following alternative calculation

$$\ln(1+x) = \begin{cases} x & \text{if } x < \epsilon \\ \frac{x\ln(1+x)}{(1+x)-1} & \text{otherwise} \end{cases}$$

gives the right answer to within machine precision for all $0 \leq x < 3/4$. Here $\epsilon = eps$ in MATLAB is the unit of least precision. Try this and see how it compares to the built-in $log1p$.
[Extra credit:] Can you explain why this magic works (no proof necessary, just look at an example)?