Math 56 Compu & Expt Math, Spring 2013: Homework 1

due 10am Thursday April 4th

Meta-tasks this week: i) Get a website if you don't already have one (see Resources); until then you can email me an archive of your codes. ii) You should by the 2nd homework be writing up answers using the LaTeX package, which is the ubiquitous, free, professional typesetting package for mathematicians. See X-hour and Resources page for how to install and use.

I recommend MATLAB/octave or python/SAGE for this part of the course. (C, fortran, or java would take much more time to code and debug.) Certain languages I don't recommend, merely since I don't know them (lisp, java, ...)

Please read carefully and try to answer all questions asked!

1. Asymptotics.
   (a) Is $\frac{e^n}{10+ne^n} = O(n^{-1})$ as $n \to \infty$? Prove your answer, i.e. if true, exhibit a C and n_0 in the definition of big-O.
   (b) Prove whether $x! = o(10^x)$ as $x \to \infty$.
   (c) For what range of N is an algorithm taking $10^6 N \log N$ effort faster than one taking N^2 effort?
2. Explore numerically then describe the apparent convergence rate of the fraction of heads in n random fair coin tosses (go up to 10^9). [You don't need to use big-O notation, since any rigorous statement here would be probabilistic anyway.] Your finding turns out to be universal to all such Monte Carlo methods!
3. Consider the series $y = \sum_{k=1}^{\infty} k^{-4}$.
   (a) Measure the convergence rate of the error $\varepsilon_n = |\hat{y}_n - y|$ for the n-term truncated approximation, by plotting ε_n vs n. Choose axis types so that the graph appears linear—what is the slope? State the type/order of convergence. [Hint: For the exact y either look it up or use the converged $\hat{y}$ after you've done d) below!]
   (b) How useful is a graph with linear axes here? Why?
   (c) Prove a big-O bound on effort (i.e. n) in terms of desired error ε. [Hint: as in lecture, but then flip the result.]
   (d) Does it matter in which order you do the sum? Give "converged" answers for both orderings, and explain which one is more accurate.
4. Write *your own, documented* function that finds one approximate root of $f(x)$, a given function of one variable, by bisection: given two starting arguments $a < c$ with $f(a)$ and $f(c)$ of opposite sign, set $b = (a+c)/2$ then replace the list a, b, c by either $a, (a+b)/2, b$, or by $b, (b+c)/2, c$, depending on what the sign of $f(b)$ tells you on which side the root lies, then iterate until you decide when to stop. Your inputs should be a *handle* to a function, the pair a, c, and an error tolerance; the output the root. It should stop and report if ever the signs don't make sense.
   (a) Add a *test script* for this function which shows it finding the root of sine in $[3, 4]$, also failing gracefully here given the input pair $a = 0$, $c = \pi$. This could be in the same text file.
   (b) State the type of convergence with n, the number of iterations, and give the tightest error bound you can in big-O notation. What n is needed to find a root to 15-digit accuracy? What happens in practice if you demand 20 digits? (These should be answered by thinking, showing working; then you can check with your code.)

BONUS State one advantage of this method over Newton's method.

5. Visualizing the complex plane.

   (a) Compute on paper: i) $\sqrt{2i}$, ii) Im $1/(3+4i)$, iii) $e^{13\pi i/4}$, iv) $|1-2i|$.

   (b) Make a short Matlab code which makes a 3D height plot of the absolute value of a given function $f(z)$ on the complex plane $z = x + iy$ for $x, y \in [-2, 2]$. [Hint: at some point you'll want to use `[X,Y] = meshgrid(...);` then apply your function to all complex grid values `X+1i*Y` at once.]

   (c) Use your code to make a plot showing the poles (non-smooth points) in the complex plane for $f(x) = 1/(1+x^2)$. What happens to the phase in the neighborhood of each pole?

   (d) Do the same for $f(x) = \sinh^{-1} x$. Where are any singularities? [Hint: they are of weaker type than poles, but visible especially if you plot the real part]

   (e) Use this to predict the convergence rate of a Taylor series for $\sinh^{-1} x$ about $x = 1$ (careful), when evaluated at $x = 0.3$. [Don't try to generate the series!]

6. Give an exact formula, in terms of β and t, for the smallest positive integer n that does not belong to the floating-point system $\mathbf{F}$, and compute n for IEEE double-precision. Give one line of code, and its output, which demonstrates this is indeed the case.

7. Let `x` be any positive number (try a really large one!). What should the following Matlab code do mathematically, and what does it do in practice? Explain why.

```
for i=1:60, x = sqrt(x); end, for i=1:60, x = x^2; end
```