Math 56 Compu & Expt Math, Spring 2013: HW1 debrief

April 9, 2013

Some of you will notice that I give bonuses for going beyond the usual in any question, not just the stated bonus ones. Effort pays off.

Note to input base-ten exponent notation into Matlab, use e.g. `5e16` *or* `2.2e-16`.

1. 2+1+2 = 5 pts

   (a) Since adding 10 to the denominator decreases the overall quantity, the proof is easy here. See e.g. John. But what if the sign on 10 were changed?

   (b) We discussed being able to use the result that factorial grows faster than even exponential. See Hanh for a proof.

   (c) Firstly, which is bigger as $N \to \infty$? N^2 grows faster than $O(N \log N)$, so the latter is better for large N. The transition point (which is needed only roughly in practice) can be solved by setting the two equal, giving a transcendental equation. You can use your later bisection code, or:

   ```
   fzero( @(x) x^2 - 1e6*x*log(x), 1e7)
   ```

   Note log is natural log by default. A couple of you found another solution at $N \approx 1$, but keep in mind for algorithms, large N is all that matters.

2. 4 pts. If asked to describe convergence, need to give the type (algebraic) and order (most of you agreed it was 1/2). No-one thought of a log-log plot which would make the average slope more visible. Under certain conditions it can be *proven* that with high probability this is the convergence rate (Central Limit Theorem).

3. 3+1+3+2 = 9 pts.

   (a) All of you did good data and plots. But measuring the slope only is meaningful down to when rounding error dominates.

   (b) You all found linear plots to crush data against the axes, being useless to the eye.

   (c) A proof is asked for, not a reiteration of *experimental* observations from (a) !

   (d) You discover (unless you use sage's sum command) that forwards only gives 13 digits, backwards 16 digits. See e.g. Hanh's graph of this.

4. 4 for working code (with documentation) +1+3 = 8 pts. Documentation best if includes the calling command and description of inputs & outputs. Question: should input tolerance on the root be interpreted as absolute or relative? (most of you did absolute)

   (a) Detecting bad signs should be done using the *sign* of f at a, b, c, not merely that f is less than some tolerance. The tolerance is on the x of the root, not on the size of f. We don't know the "typical size" of the user's supplied f. In other words, scaling f by a small (or large) number should have no effect on relative accuracy of f and hence ability to find roots. The following should run in the same way:

   ```
   bisection(@(x) sin(x),3,4,1e-15)
   bisection(@(x) 1e-20*sin(x),3,4,1e-15)
   ```

(b) Demanding tolerance of `1e-20` for the root π crashed or made most of your codes run infinitely long. This is generally bad, and you need to think about how to detect such issues if you are to release software.

BONUS Advantage is it doesn't need derivatives of f.

5. 3+2+2+2+1 = 10 pts

(a) All good.

(b) see many of your codes.

(c) Don't forget elementwise operation, e.g.

```
f = @(x) 1./(1+x.^2)
```

otherwise you'll get garbage (some weird matrix-matrix producting).

(d) The singularities at $\pm i$ are square-root type (locally they look like $\sqrt{}$). Furthur up and down the imaginary axis are *branch cuts*, responsible for the jump in real part. Ben proved there are singularities here by looking at the derivative of $\sinh^{-1}$, which is similar to part (c).

See `http://mathworld.wolfram.com/InverseHyperbolicSine.html`

(e) Dist of nearest sing from expansion pt is $\sqrt{2}$, and dist of x from expansion pt is 0.7. Rate is therefore their ratio $0.7/sqrt2 < 1$, exponential convergence.

6. 4 pts. `(2^53 + 1) - 2^53` gives zero, showing that the +1 is lost in rounding the bracketed expression to a floating-point number. But `(2^53 - 1) - 2^53` gives -1, showing that $2^{53} - 1$ is correctly represented. Notice that a complete answer really should demonstrate that a smaller integer is correctly represented.

7. 3 pts. The power $1/2^{60}$ is so ridiculously small, but, as only a couple of you found, is not enough to reduce all x to exactly 1 in floating point, so something can survive, for x bigger than around 1.52×10^{112}. See Hanh's step-function plot for exponentially large x. If you never found these large x for which the returned answer is not 1, you got 2/3.