Scientific Computing CS660 Fall '11

Angjoo Kanazawa

September 6, 2011

1 September 1st Class one

1.1 Logistics

Prof. Howard Elman CSI 2120 TR 2pm-3:15pm class url:http://www.cs.umd.edu/ elman/660.11/syl.html

- Scientific Computing puts heavier emphasis on computing, Numerical Analysis is more about proofs/theories.
- 4-6 hw asg: **35%** Penalty on late assignments (-15% after 24 hrs, -30% after 48 hrs.
- in-class midterm: **25%**
- final project: **40%**

1.2 Content

Newton's Method: Root finding. Objective: Fine x s.t. $f(x) = 0$, where f is a scalar, $f : \mathbf{R} \Rightarrow \mathbf{R}$ function. Where does the function cross 0 (x-axis)?

Given x_n, some guess, find where the line through $(x_n, f(x_n))$ tangent to the solution curve intersects the x-axis. Call that pt of intersection x_{n+1}

The equation of the tangent line:

$$\frac{y - f(x_n)}{x - x_n} = f'(x_n)$$

Set $y = 0$: then

$$\begin{aligned}\frac{0 - f(x_n)}{x_{n+1} - x_n} &= f'(x_n)\\ \frac{x_{n+1} - x_n}{-f(x_n)} &= 1/f'(x_n)\\ x_{n+1} &= x_n - f(x_n)/f'(x_n)\end{aligned}$$

Another Derivation Consider the taylor series $f(x_n + (x - x_n)) = f(x_n) + f'(x_n)(x - x_n) + 1/2f''(x_n)(x - x_n)^2 +$ etc. This is a function of some variable x. Approximate it just by using the first two terms (linear approximation). So above becomes a new function $f(x_n+(x-x_n)) \ = f(x_n)+f'(x_n)(x-x_n) = l(x)$. Find where $l(x) = 0$. That is: $x_{n+1} = x_n - f(x_n)/f'(x_n)$

This is not guaranteed to work, i.e. when the tangent doesn't cross the x-axis.

Problem: given $\alpha \in \mathbf{R} > 0$ Find $1/\alpha$ without doing any division. First thing you need to do if identify (concoct) a $f(x)$ whose root is $1/\alpha$. Naive: try $f(x) = x - 1/\alpha$. But this won't work, because this requires division. Howabout: $f(x) = \alpha x - 1$. $f'(x) = \alpha$, so in the newton iteration..

$$\begin{aligned} x_{n+1} &= x_n - \frac{\alpha x - 1}{\alpha} \\ &= x_n - x_n + 1/\alpha \\ &= 1/\alpha \end{aligned}$$

No! because you need to divide.

Answer: $f(x) = \alpha - 1/x$. Not transparent, because intuitively it looks like there's a divide into it. $f'(x) = - - 1/x^2 = 1/x^2$. Then, $f/f' = \alpha x^2 - x$. So given x_n, the iteration goes $x_{n+1} = x_n - (\alpha x_n^2 - x_n) = 2x_n - \alpha x_n^2 = x_n(2 - \alpha x_n)$.

Numerical example in matlab: let $\alpha = 2$, solve with this method. Use $x_0 = 0.1$. Notice that $err(i)/err(i-1)^2$ is constant and *is* α.

$$\begin{aligned} \frac{err(i)}{err(i-1)^2} &= \frac{|x - x_{n+1}}{x - x_n}^2 \\ &\approx 1/2|\frac{f''(x_n)}{f'(x_n)}| \\ &= 1/2\frac{\frac{2x}{x^4}}{\frac{1}{x^2}} \\ &= 1/2\frac{2x^3}{x^4} = \frac{1}{x} \end{aligned}$$

So at $x = 1/\alpha$, this is α.
Notice the $err(i)$ decreases faster as iteration moves on, this is the super linear convergence property of Newton's method. With $\alpha = 0.25$, same thing.

Analysis of the Newton's Method: $|x_{x_{n+1}}|/|x - x_n|^2 \approx 1/2|f''(x_n)/f'(x_n)| \approx 1/x$ So @ $x = 1/\alpha$, this turns out to be α, just for this example.

The ratio was derived from the idea to find a patter in the errors s.t. $e_{n+1}\ ce_n^2$, for some $c \in \mathbf{R}$

The question is to find trends in data, and this relationship $e_{n+1}\ ce_n^p$ is useful to tell us the rate of convergence as the solution approches the optimal one. (I think p goes down to the golden ratio). Our goal is to find what p is for a specific problem.

2 September 6th class 2

2.1 Process of Scientific Computing

- Start with a mathematical model. (In general we don't have an analytic solution, so we get insight from numerical computation)

- Example with heat conduction in a bar:
  - A 1-D object $\in [0,1]$, $u(x)$ =temperature in the bar, with $u(0) = 0, u(1) = 0$. q=heat flow induced by a heater of intensity f.
  - We want what the temperature will be given q.
  - get some models: *Fourrier's Law*: $q = -ku'$, k=conductivity coefficient, transfer of heat in direction of decreasing temperature (hence the -). *Conservation of energy*: $q' = f$.
  - **Goal**: find u. **Equation of interest**: $-(ku')' = f$, the 1D diffusion equation.
  - Typical strategy: lay down a grid $x_o = 0, x_1, \ldots, x_n, x_n + 1 = 1$. Compute a discrete solution, $\bar{u}$, vector of size n. $\bar{u} = [u_1, \cdot, u_n]^T$, where $u_i \approx u(x_i)$
  - **Claim**: we can find the discrete sol, $\bar{u}$ by solving an algebraic system of equations. For this example, this system is a matrix equation

$$A\bar{u} = \bar{b}$$

- No matter how hard I try, we're never going to get the exact solution. This process $A\bar{u} = \bar{b}$ leads to errors
- **Sources of error**:
  1. Modeling error: we may not know k exactly.
  2. Discretization error/Truncation error: difference between the discrete and the continuous values (from the approximation on a discrete set of points)
  3. Representation error: we don't have the entire $\mathbf{R}$, we only have a finite set, in floating point format. (A and b may have error)
  4. Additional error: from the computation of $\tilde{u}$, will get something else $\hat{\tilde{u}} \neq \tilde{u}$
     we can show:
     $$\frac{||\hat{\tilde{u}} - \tilde{u}||}{\hat{\tilde{u}}||} \leq K(A)\mu$$
     That is if we solve the problem appropriately (like being careful about pivoting etc).
- We're really trying to solve $\tilde{A}\tilde{u} = \tilde{b}.$, where $\tilde{A} \approx A$, $\tilde{u} = \bar{u}$. Typically $\approx$ is machine precision, 10^{-16}
- In the end, we want $u(x)$, and would be happy with $u(x_j)$, $j = 1, \ldots, n$. That will get $\tilde{u_j}$
- The moral is that when we do this stuff, we're not just doing mathematics.

2.2 Floating Point Arithmetic

decimal numbers (base 10). Consider the example 6522 and 10.31

- $6522 = 6(10^3) + 5(10^2) + 2(10^1) + 2(10^0)$
- $10.31 = 1(10^1) + 0 + 3(10^-1) + 1(10^-2)$
- Normalize the numbers! then,,
- $6522 = 6.522 \times 10^3$ so we can express numbers with one digit to the left o fthe decimal point.
- $10.31 = 1.031 \times 10^1$
- For any number but 0, it has a fomr $z \times 10^p$, where $z \in [1, 10)$.

Computers use binary representation. Example: $3_{10} = 1(2^1) + 1(2^0) = 11$ or 1.1000×2^1.

$$\begin{aligned} 23_{10} &= 16 + 4 + 2 + 1 && (1)\\ &= 1(2^4) + 0(2^3) + 1(2^2) + 1(2^1) && (2)\\ &= 10111_2 && (3)\\ &= 1.0111 \times 2^4 (normalized) && (4) \end{aligned}$$

Here, normalized means $z \times 2^p$, where $z \in [1, 2)$.
Definition: z is the *mantissa/significand*, p is the *exponent.*
Convention: use d digits(bits) for the mantissa, $m \leq p \leq M$.
IEEE standard (2 words-64bits or 1 word-32bits)

- In a single precision (32 bits to use), $d = 24$ and $m = -126 \leq p \leq M = 127$. the first bit is the sign bit, then we have 23 bits for the mantissa, then the 8 bits for exponent.
- In a double precision (64 bits), $d = 53$ and (1 for sign, 52 for mantissa, 11 bits for the exponent) $m = -1022$, $M = 2^{11-1} - 1 = 1023$
- The decision to normalize helps us here because every number has a 1 in the highest bit. So we only need 23 bit to represent a number with 24 binary digits.
- To understand the exponential, suppose we have 3 bits at our disposal (not 8). What numbers can be stored with 3 bits? $000, 001, 011, \ldots, 111$, or $0, \ldots, 7 = 2^3 - 1$. (last number is always $2^b - 1$)
- Let's subtract 3, the *bias*, from this. then, this gives us $-3, \ldots, 4$. This suggests that with three digits, these are the exponents I can represent. These are our candidate for m and M.
- Exponent is represented using an offset bias
- Another example. Suppose we had 4 bits to use for exponents. What numbers can we store with 4 bits? $0, \ldots, 15 = 2^{bits} - 1$, Now subtract with bias 7. Then we can represent $-7, \ldots, 8 = 2^{bits}/2 = 2^{bits-1}$

- So for single precision, with 8 exponents, $M = 2^{8-1} = 128, m = M - 1$. But why $M = 127$? because we don't have as many numbers as we want because we need to take care of 0, NaN, etc so we leave aside 2 bits from the exponent to take care of such busines.