

Notes for 2015-01-23

Logistics

- Please make sure you can access CMS and Piazza
- PS1 is posted and is due next Wednesday

I lost on Jeopardy!

These are questions you should know something about from the reading.

1. Suppose you want to find $f(x_*) = 0$, and you find an approximation \hat{x} . What's the relation between:
 - (a) Absolute error ($|\hat{x} - x_*|$)
 - (b) Relative error ($|\hat{x} - x_*|/|x_*|$)
 - (c) Residual error ($f(\hat{x})$)
2. Why might you sometimes need to use termination criteria that aren't based on error measures?
3. Suppose x_* is a fixed point of $g(x)$. Under what condition does fixed point iteration on g converge to x_* starting from nearby points?
4. MATLAB has a fairly robust function `fzero` that finds the zeros of a nonlinear function of one variable (using a combination of bisection and secant iteration). Why might you want to sometimes roll your own?

Choice of functions and variables

Root-finding problems are hard or easy depending on how they are posed. Often, the initial problem formulation is not the most convenient. For example, consider the problem of finding the positive root of

$$f(x) = (x + 1)(x - 1)^8 - 10^{-8}.$$

This function is terrifyingly uninformative for values close to 1. Newton's iteration is based on the assumption that a local, linear approximation provides a good estimate of the behavior of a function. In this problem, a linear approximation is terrible. Fortunately, the function

$$g(x) = (x + 1)^{1/8}(x - 1) - 10^{-1}$$

has the same root, which is very nicely behaved.

There are a few standard tricks to make root-finding problems easier:

- Scale the function. If $f(x)$ has a zero at x_* , so does $f(x)g(x)$; and sometimes we can analytically choose a scaling function to make the root finding problem easier.
- Otherwise transform the function. For example, in computational statistics, one frequently would like to maximize a likelihood function

$$L(\theta) = \prod_{j=1}^n f(x_j; \theta)$$

where $f(x; \theta)$ is a probability density that depends on some parameter θ . One way to do this would be find zeros of $L'(\theta)$, but this often leads to scaling problems (potential underflow) and other numerical discomforts. The standard trick is to instead maximize the log-likelihood function

$$\ell(\theta) = \sum_{j=1}^n \log f(x_j; \theta),$$

often using a root finder for $\ell'(\theta)$. This tends to be a much more convenient form, both for analysis and for computation.

- Change variables. A good rule of thumb is to pick variables that are naturally *dimensionless*¹. For difficult problems, these dimensionless variables are often very small or very large, and that fact can be used to simplify the process of coming up with good initial guesses for Newton iteration.

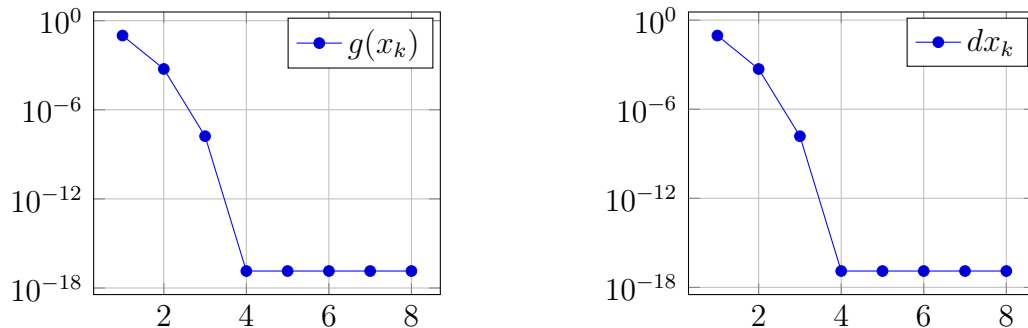


Figure 1: Convergence of Newton's iteration for $g(x) = (x+1)^{1/8}(x-1) - 0.1$

Monitoring convergence

A very natural way to monitor convergence is via a semi-logarithmic plot. We plot both the residual error $g(x_k)$ and the change $x_{k+1} - x_k$. Linearly convergent iterations such as bisection and simple fixed-point methods should produce linear convergence plots. Newton is quadratically convergent, which means that the convergence plot should look like a downward-facing parabola once convergence kicks in (Figure 1). In practice, convergence is so fast that there are often only two or three points between when the iteration begins to converge and when floating point errors begin to dominate.

If you compute a derivative wrong when you code Newton iteration, the resulting iteration will often converge – but linearly. So looking at a plot like this is a good way to diagnose coding errors that affect performance without ultimately affecting the ability to compute a correct result.

Note that I have used the `pgfplots` package for drawing my plots; this tends to look nice in \LaTeX , but it's not always the simplest thing. You are welcome to mimic this file if you want to use `pgfplots` to draw convergence plots for your homework; alternately, you can use MATLAB graphics. Or, if you prefer, you can use Python and `matplotlib`. We only really need MATLAB code for problems where you will submit the code.

¹Those of you who are interested in applied mathematics more generally should look up the Buckingham Pi Theorem — it's a tremendously useful thing to know about.

Starting points

All root-finding software requires either an initial guess at the solution or an initial interval that contains the solution. This sometimes calls for a little cleverness, but there are a few standard tricks:

- If you know where the problem comes from, you may be able to get a good estimate (or bounds) by “application reasoning.” This is often the case in physical problems, for example: you can guess the order of magnitude of an answer because it corresponds to some physical quantity that you know about.
- Crude estimates are often fine for getting upper and lower bounds. For example, we know that for all $x > 0$,

$$\log(x) \leq x - 1$$

and for all $x \geq 1$, $\log(x) > 0$. So if I wanted to $x + \log(x) = c$ for $c > 1$, I know that c should fall between x and $2x - 1$, and that gives me an initial interval. Alternatively, if I know that $g(z) = 0$ has a solution close to 0, I might try Taylor expanding g about zero – including higher order terms if needed – in order to get an initial guess for z .

- Sometimes, it’s easier to find local minima and maxima than to find zeros. Between any pair of local minima and maxima, functions are either monotonically increasing or monotonically decreasing, so there is either exactly one root in between (in which case there is a sign change between the local min and max) or there are zero roots between (in which case there is no sign change). This can be a terrific way to start bisection.

Problems to ponder

1. Analyze the convergence of the fixed point iteration

$$x_{k+1} = c - \log(x_k).$$

What is the equation for the fixed point? Under what conditions will the iteration converge with a good initial guess, and at what rate will the convergence occur?

2. Repeat the previous exercise for the iteration $x_{k+1} = 10 - \exp(x_k)$.
3. Analyze the convergence of Newton's iteration on the equation $x^2 = 0$, where $x_0 = 0.1$. How many iterations will it take to get to a number less than 10^{-16} ?
4. Analyze the convergence of the fixed point iteration $x_{k+1} = x_k - \sin(x_k)$ for x_k near zero. Starting from $x = 0.1$, how many iterations will it take to get to a number less than 10^{-16} ?
5. Consider the cubic equation

$$x^3 - 2x + c = 0.$$

Describe a general purpose strategy for finding *all* the real roots of this equation for a given c .

6. Suppose we have some small number of samples X_1, \dots, X_m drawn from a Cauchy distribution with parameter θ (for which the pdf is)

$$f(x, \theta) = \frac{1}{\pi} \frac{1}{1 + (x - \theta)^2}.$$

The *maximum likelihood estimate* for θ is the function that maximizes

$$L(\theta) = \prod_{j=1}^m f(X_j, \theta).$$

Usually, one instead maximizes $l(\theta) = \log L(\theta)$ — why would this make sense numerically? Derive a MATLAB function to find the maximum likelihood estimate for θ by finding an appropriate solution to the equation $l'(\theta) = 0$.

7. The Darcy friction coefficient f for turbulent flow in a pipe is defined in terms of the Colebrook-White equation for large Reynolds number Re (greater than 4000 or so):

$$\frac{1}{\sqrt{f}} = -2 \log_{10} \left(\frac{\epsilon/D_h}{3.7} + \frac{2.51}{\text{Re}\sqrt{f}} \right)$$

Here ϵ is the height of the surface roughness and D_h is the diameter of the pipe. For a 10 cm pipe with 0.1 mm surface roughness, find f for Reynolds numbers of 10^4 , 10^5 , and 10^6 . Ideally, you should use a Newton iteration with a good initial guess.

8. A cable with density of 0.52 lb/ft is suspended between towers of equal height that are 500 ft apart. If the wire sags by 50 ft in between, find the maximum tension T in the wire. The relevant equations are

$$c + 50 = c \cosh \left(\frac{500}{2c} \right)$$
$$T = 0.52(c + 50)$$

Ideally, you should use a Newton iteration with a good initial guess.