CS412: Lecture #3

Mridul Aanjaneya

January 27, 2015

Machine ε (epsilon)

A concept that is useful in quantifying the error caused by rounding or truncation is the notion of the machine ε (epsilon). There are a number of (slightly different) definitions in the literature, depending on whether truncation or rounding is used, specific rounding rules, etc. Here, we will define the machine ε as the smallest positive machine number, such that

$$1+\varepsilon \neq 1 \quad \text{(on the computer)}$$

Why isn't the above inequality always true, for any $\varepsilon > 0$? The reason is that when subject to the computer precision limitations, some numbers are "too small" to affect the result of an operation, e.g.

$$1 = 1.\underbrace{000\ldots000}_{23 \text{ digits}} \times 2^0$$
$$2^{-25} = 0.\underbrace{000\ldots000}_{23 \text{ digits}}01 \times 2^0$$
$$1+2^{-25} = 1.\underbrace{000\ldots000}_{23 \text{ digits}}01 \times 2^0$$

When rounding (or truncating) the last number to 23 binary significant digits corresponding to single precision, the result would be exactly the same as the representation of the number $x = 1$! Thus, on the computer we have, in fact, $1 + 2^{-25} = 1$, and consequently 2^{-25} is smaller than the machine epsilon. We can see that the smallest positive number that would actually achieve $1+\varepsilon \neq 1$ with single precision machine numbers is $\varepsilon = 2^{-24}$ (and we are even relying a "round upwards" convention for tie breaking to come up with a value this small), which will be called the machine ε in this case. For double precision, the machine ε is 2^{-53}.

The significance of the machine ε is that it provides an upper bound for the relative error of representing any number to the precision available on the computer; thus, if $q > 0$ is the intended numerical quantity, and $\hat{q}$ is the closest machine-precision approximation, then

$$(1-\varepsilon)q \leq \hat{q} \leq (1+\varepsilon)q$$

where ε is the machine epsilon for the degree of precision used; a similar expression holds for $q < 0$.

Solving nonlinear equations

We turn our attention to the first major focus topic of our class: techniques for solving nonlinear equations. In an earlier lecture, we actually addressed one common nonlinear equation, the *quadratic* equation $ax^2 + bx + c = 0$, and discussed the potential hazards of using the seemingly straightforward quadratic solution formula. We will start our discussion with an even simpler nonlinear equation:

$$x^2 - a = 0, \quad a > 0$$

The solution is obvious, $x = \pm\sqrt{a}$ (presuming, of course, that we have a subroutine at our disposal that computes square roots). Let us, however, consider a different approach:

- Start with $x_0 =$ `<initial guess>`
- Iterate the sequence

$$x_{k+1} = \frac{x_k^2 + a}{2x_k} \tag{1}$$

We can show (and we will via examples) that this method is quite effective at generating remarkably good approximations of $\sqrt{a}$ after just a few iterations. Let us, however, attempt to analyze this process from a theoretical standpoint.

If we assume that the sequence $x_0, x_1, x_2, \ldots$ defined by this method has a limit, how does that limit relate to the problem at hand? Assume $\lim_{k\to\infty} = A$. Then, taking limits on equation (1) gives

$$\lim_{k\to\infty} x_{k+1} = \lim_{k\to\infty} \frac{x_k^2 + a}{2x_k} \Rightarrow A = \frac{A^2 + a}{2A} \Rightarrow 2A^2 = A^2 + a \Rightarrow A^2 = a \Rightarrow A = \pm\sqrt{a}$$

Thus, if the iteration converges, the limit is the solution of the nonlinear equation $x^2 - a = 0$. The second question is whether it may be possible to guarantee that the described iteration *will* converge. For this, we manipulate equation (1) as follows

$$x_{k+1} = \frac{x_k^2 + a}{2x_k} \Rightarrow x_{k+1} - \sqrt{a} = \frac{x_k^2 + a}{2x_k} - \sqrt{a} = \frac{x_k^2 - 2x_k\sqrt{a} + a}{2x_k} = \frac{[x_k - \sqrt{a}]^2}{2x_k}$$

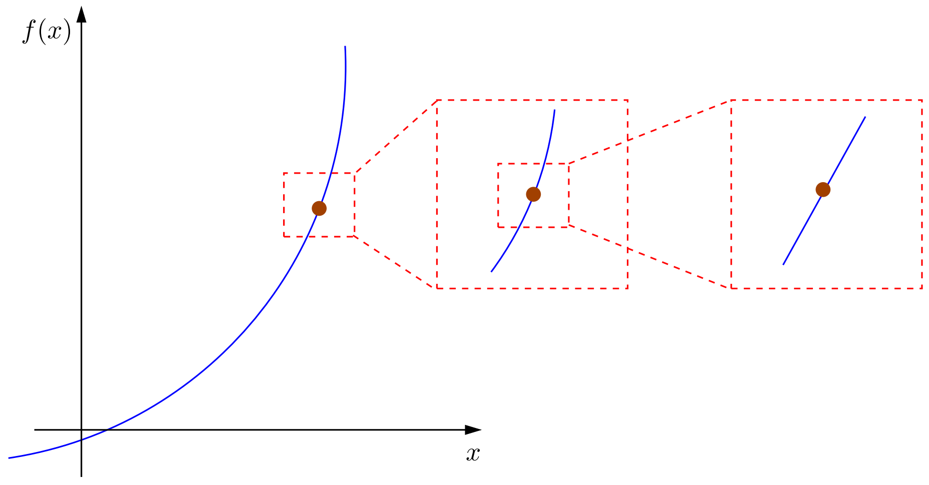
If we denote by $e_k = x_k - \sqrt{a}$ the error (or discrepancy) from the exact solution of the approximate value x_k, the previous equation reads

$$e_{k+1} = \frac{e_k^2}{2x_k} = \frac{e_k^2}{2(e_k + \sqrt{a})} \tag{2}$$

For example, if we were approximating the square root of $a = 2$, and at some point we had $e_k = 10^{-3}$, the previous equation would suggest that $e_{k+1} < 10^{-6}$. One more application of this equation would yield $e_{k+2} < 10^{-12}$. Thus we see that, provided the iteration starts *close enough* to the solution, we not only converge to the desired value, but actually double the number of correct significant digits in each iteration. We defer the detailed proof until after we have introduced the more general method.

Newton's method

This example is a special case of an algorithm for solving nonlinear equations known as Newton's method (also called the *Newton-Raphson* method). The general idea is as follows: if we "zoom" close enough to any smooth function, its graph looks more and more like a straight line (specifically, the *tangent* line to the curve). Newton's method suggests: if after k iterations we have

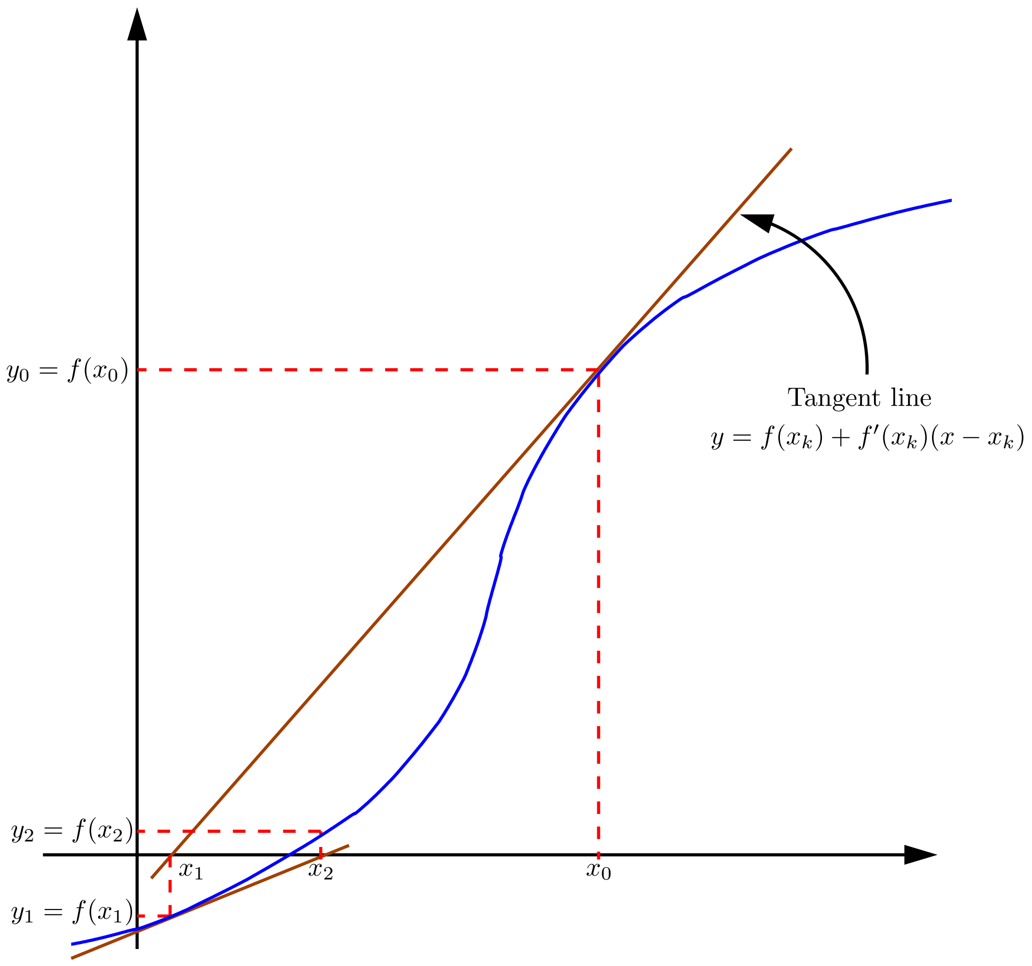


approximated the solution of $f(x) = 0$ (a general nonlinear equation) as x_k, then:

- Form the tangent line at $(x_k, f(x_k))$.
- Select x_{k+1} as the intersection of the tangent line with the horizontal axis $(y = 0)$.

If $(x_n, y_n) = (x_n, f(x_n))$, the tangent line to the plot of $f(x)$ at (x_n, y_n) is:

$$y - y_n = \lambda(x - x_n), \quad \text{where } \lambda = f'(x_n) \text{ is the slope}$$



Thus, the tangent line has equation $y - y_n = f'(x_n)(x - x_n)$. Setting $y = 0$ gives

$$-f(x_n) = f'(x_n)(x - x_n) \Rightarrow x = x_n - \frac{f(x_n)}{f'(x_n)} = x_{n+1}$$

Ultimately, Newton's method reduces to:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3}$$

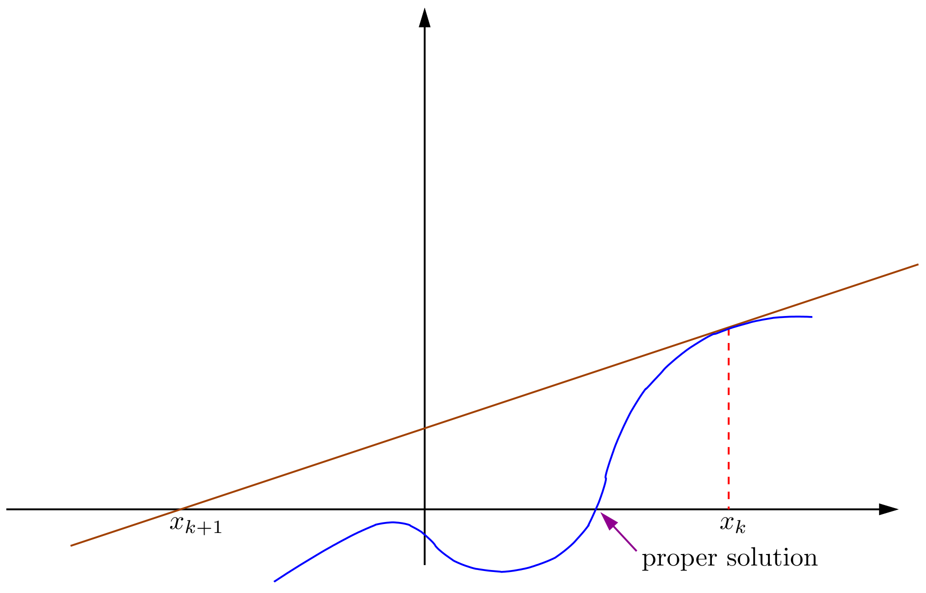
Our previous example (square root of a) is just an application of Newton's method to the nonlinear equation $f(x) = x^2 - a = 0$. Applying equation (3) gives:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - a}{2x_k} = \frac{2x_k^2 - x_k^2 + a}{2x_k} = \frac{x_k^2 + a}{2x_k}$$

which is the same iteration we considered previously.

A few comments about Newton's method:

- It requires the function $f(x)$ to be not only continuous, but differentiable as well. We will later see variants that do not *explicitly* require knowledge of f'. This would be an important consideration if the formula for $f'(x)$ is significantly more complex and expensive to evaluate than $f(x)$, or if we simply do not possess an analytic expression for f' (this could be the case if $f(x)$ is not given to us via an explicit formula, but only defined via a black-box computer function that computes the value).
- If we ever have an approximation x_k with $f'(x_k) \approx 0$, we should expect problems, especially if we are not close to a solution (we would be nearly dividing by zero). In such cases, the tangent line is almost (or exactly) horizontal. Thus, the next iterate can be a very remote value and convergence may be far from guaranteed.



Fixed point iteration

Newton's method is in itself a special case of a broader category of methods for solving nonlinear equations called *fixed point iteration* methods. Generally, if $f(x) = 0$ is the nonlinear equation we seek to solve, a fixed point iteration method proceeds as follows:

- Start with $x_0 =$ `<initial guess>`.
- Iterate the sequence

  $$x_{k+1} = g(x_k)$$

  where $g(x)$ is a properly designed function for this purpose. Note that $g(x)$ is related, but otherwise different than $f(x)$.

Following this method, we construct the sequence $x_0, x_1, x_2, \ldots, x_k, \ldots$ hoping that it will converge to a solution of $f(x) = 0$. The following questions arise at this point:

1. If this sequence converges, does it converge to *a solution of* $f(x) = 0$?
2. Is this iteration guaranteed to converge?
3. How fast does this iteration converge?
4. (Of practical concern) When do we stop iterating and declare that we have obtained an acceptable approximation?

We start by addressing the first question: if the sequence $\{x_k\}$ does converge, can we ensure that it will converge to a solution of $f(x) = 0$? Taking limits on $x_{k+1} = g(x_k)$, and assuming that

1. $\lim_{k\to\infty} x_k = a$, and
2. the function g is continuous,

gives

$$\lim_{k\to\infty} x_{k+1} = \lim_{k\to\infty} g(x_k) \Rightarrow a = g(a)$$

The simplest way to guarantee that a is a solution to $f(x) = 0$ (in other words, $f(a) = 0$) is if we construct $g(x)$ such that

$$x = g(x) \quad \text{is mathematically equivalent to} \quad f(x) = 0$$

There are many ways to make this happen, e.g.,

$$f(x) = 0 \Leftrightarrow x + f(x) = x \Leftrightarrow x = g(x), \quad \text{where} \quad g(x) \equiv x + f(x)$$

or

$$f(x) = 0 \Leftrightarrow e^{-x} f(x) = 0 \Leftrightarrow e^{-x} f(x) + x^2 = x^2 \Leftrightarrow \frac{e^{-x} f(x) + x^2}{x} = x \Leftrightarrow$$

$$\Leftrightarrow g(x) = x \quad \text{where} \quad g(x) \equiv \frac{e^{-x} f(x) + x^2}{x}$$

or

$$f(x) = 0 \Leftrightarrow -\frac{f(x)}{f'(x)} = 0 \Leftrightarrow x - \frac{f(x)}{f'(x)} = x \Leftrightarrow g(x) = x, \quad \text{where} \quad g(x) \equiv x - \frac{f(x)}{f'(x)}$$

The last example is exactly Newton's method; substituting the definition of $g(x)$ above into the iteration $x_{k+1} = g(x_k)$ yields the familiar Newton update equation. Thus, we know that if fixed point iteration converges, it will be to a solution of $f(x) = 0$.