Lecture 3: Solving Equations Using Fixed Point Iterations

Instructor: Professor Amos Ron Scribes: Yunpeng Li, Mark Cowlishaw, Nathanael Fillmore

Our problem, to recall, is solving equations in one variable. We are given a function f, and would like to find at least one solution to the equation $f(x) = 0$. Note that, *a priori*, we do not put any restrictions on the function f; we do need to be able to evaluate the function: otherwise, we cannot even check that a given solution $x = r$ is true, i.e., that $f(r) = 0$. In reality, the mere ability to be able to evaluate the function does not suffice. We need to assume some kind of "good behavior". The more we assume, the more potential we have, on the one hand, to develop fast algorithms for finding the root. At the same time, the more we assume, the fewer functions are going to satisfy our assumptions! This is a fundamental paradigm in Numerical Analysis.

Recall from last week that we wanted to solve the equation:

$$\begin{array}{ll} x^3 = \sin x & \text{or} \\ x^3 - \sin x = 0 & \end{array} \tag{1}$$

We know that 0 is a trivial solution to the equation, but we would like to find a non-trivial numeric solution r. In a previous lecture, we introduced an iterative process for finding roots of quadratic equations. We will now generalize this process into an algorithm for solving equations that is based on the so-called *fixed point iterations*, and therefore is referred to as *fixed point algorithm*. In order to use fixed point iterations, we need the following information:

1. We need to know that there is a solution to the equation.
2. We need to know approximately where the solution is (i.e. an approximation to the solution).

1 Fixed Point Iterations

Given an equation of one variable, $f(x) = 0$, we use fixed point iterations as follows:

1. Convert the equation to the form $x = g(x)$.
2. Start with an initial guess $x_0 \approx r$, where r is the actual solution (root) of the equation.
3. Iterate, using $x_{n+1} := g(x_n)$ for $n = 0, 1, 2, \ldots$.

How well does this process work? We claim that:

CLAIM 1.1. *Define $(x_n)_0^\infty$ using $x_{n+1} := g(x_n)$ as described in the process above. If $(x_n)_0^\infty$ converges to a limit r, and the function g is continuous at $x = r$, then the limit r is a root of $f(x)$: $f(r) = 0$.*

Why is this true? Assume that $(x_n)_0^\infty$ converges to some value r. Since g is continuous, the definition of continuity implies that

$$\lim_{n\to\infty} x_n = r \Rightarrow \lim_{n\to\infty} g(x_n) = g(r)$$

Using this fact, we can prove our claim:

$$g(r) = \lim_{n\to\infty} g(x_n) = \lim_{n\to\infty} x_{n+1} = r.$$

Thus, $g(r) = r$, and since the equation $g(x) = x$ is equivalent to the original one $f(x) = 0$, we conclude that $f(r) = 0$.

Note that, for proving this claim, we had to make some assumption on g (viz, g is a continuous function). This follows the general pattern: the more restrictions we put on a function, the better we can analyze the function numerically.

The real trick of fixed point iterations is in Step 1, finding a transformation of the original equation $f(x) = 0$ to the form $x = g(x)$ so that $(x_n)_0^\infty$ converges. Using our original example, $x^3 = \sin x$, here are some possibilities:

1. $x = \frac{\sin x}{x^2}$
2. $x = \sqrt[3]{\sin x}$
3. $x = \sin^{-1}(x^3)$
4. $x = \frac{\sin x - 1}{x^2 + x + 1} + 1$
5. $x = x - \frac{x^3 - \sin x}{3x^2 - \cos x}$

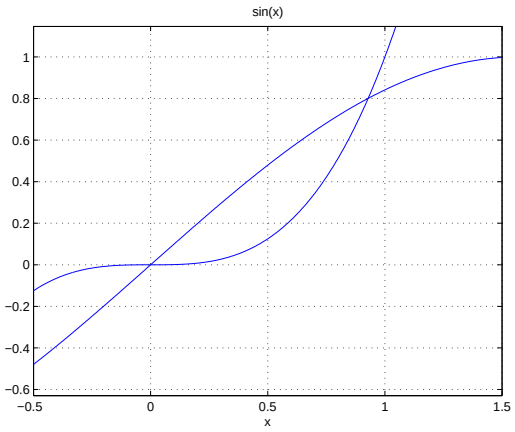

Figure 1: Graphical Solution for $x^3 = \sin x$

We can start with $x_0 = 1$, since this is a pretty good approximation to the root, as shown in Figure 1. To choose the best function $g(x)$, we need to determine how fast (and if) x_n converges to a solution. This is key, because different transformations of a single $f(x) = 0$ to $x = g(x)$ can result in a sequence of x_n that diverges, converges to the root slowly, or converges to the root quickly.

One good way to measure the speed of convergence is to use the ratio of the errors between successive iterations. The error at iteration n can be defined as:

$$e_n := x_n - r \tag{2}$$

where r is the actual solution (an alternative definition, is $e_n := |x_n - r|$.) To measure the rate of convergence, we can take the ratio μ_{n+1} between the error at iteration $n+1$ and the error at the previous iteration:

$$\mu_{n+1} := \frac{e_{n+1}}{e_n} = \frac{x_{n+1} - r}{x_n - r} \tag{3}$$

However, as you may have noticed, we are using the actual solution r, which we do not know, to calculate the error e_n and the ratio μ_n. For as long as we do not know r , we can approximate the error $e_n = x_n - r \approx x_n - x_{n-1}$, and thus we can calculate the error ratio as:

$$\mu_{n+1} = \frac{e_{n+1}}{e_n} \approx \frac{x_{n+1} - x_n}{x_n - x_{n-1}} \tag{4}$$

Note that the magnitude of the error ratio is what is important, so we can safely ignore the sign.

1.1 Order of Convergence

Clearly, we would like the magnitude of the error ratio to be less than 1. We introduce now two notions concerning orders of convergence.

1.1.1 Linear Convergence

Linear convergence requires that the error is reduced by at least a constant factor at each iteration:

$$|e_{n+1}| \leq c \cdot |e_n| \tag{5}$$

for some fixed constant $c < 1$. We will study algorithms that converge much more quickly than this, in fact, we have already seen an algorithm (the square root algorithm) that has *quadratic convergence*.

1.1.2 Quadratic Convergence

Quadratic convergence requires that the error at each iteration is proportional to the square of the error on the previous iteration:

$$|e_{n+1}| \leq c \cdot |e_n|^2 \tag{6}$$

for some constant c. Note that, in this case, c does not have to be less than 1 for the sequence to converge. For example, if $|e_n| \approx 10^{-4}$, then $|e_{n+1}| < c \cdot 10^{-8}$, so that a relatively large constant can be offset by the squaring.

1.2 Superlinear Convergence

In general, we can have

$$|e_{n+1}| \leq c \cdot |e_n|^\alpha \tag{7}$$

for constants c and α. If $\alpha = 1$, we have linear convergence, while if $\alpha = 2$ we have quadratic convergence. If $\alpha > 1$ (but is not necessarily 2), we say we have superlinear convergence.

It is important to note that equations 5, 6, and 7 provide *lower bounds* on the convergence rate. It is possible that an algorithm with quadratic convergence will converge more quickly than the bound indicates in certain circumstances, but it will never converge more slowly than the bound indicates. Also note that these bounds are a better indicator of the performance of an algorithm when the errors are small ($e_n \ll 1$).

1.3 Experimental Comparison of Functions for Fixed Point Iterations

We Now return to our test problem:

EXAMPLE 1.1. *Solve the equation*

$$x^3 = \sin x. \tag{8}$$

How do the functions we considered for $g(x)$ compare? Table 1 shows the results of several iterations using initial value $x_0 = 1$ and four different functions for $g(x)$. Here x_n is the value of x on the nth iteration and μ_n is the error ratio of the nth iteration, as defined in Equation 4.

	$g(x) = \sqrt[3]{\sin x}$	$g(x) = \frac{\sin x}{x^2}$	$g(x) = x + \sin x - x^3$	$g(x) = x - \frac{\sin x - x^3}{\cos x - 3x^2}$
x_1:	0.94408924124306	0.84147098480790	0.84147098480790	0.93554939065467
μ_1:	-0.05591075875694	-0.15852901519210	-0.15852901519210	-0.06445060934533
x_2:	0.93215560685805	1.05303224555943	0.99127188988250	0.92989141894368
μ_2:	0.21344075183985	-1.33452706115132	-0.94494313796800	0.08778771478600
x_3:	0.92944074461587	0.78361086350974	0.85395152069647	0.92886679103170
μ_3:	0.22749668328899	-1.27349109705917	-0.91668584457246	0.18109456255982
x_4:	0.92881472066057	1.14949345383611	0.98510419085185	0.92867234089417
μ_4:	0.23059142581182	-1.35803100534498	-0.95508533025939	0.18977634246913
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
x_{26}:	0.92862630873173	-0.00000000000000	0.89462921748990	0.92862630873173
μ_{26}:	0	-1.00000000000000	-0.97525571602895	NaN
x_{27}:	0.92862630873173	0	0.89614104323697	0.92862630873173
μ_{27}:	NaN	-1.00000000000000	-0.97635939022401	NaN

Table 1: Comparison of Functions for Fixed Point Iterations

We can see from the table that when we choose $g(x) = \sqrt[3]{\sin x}$ or $g(x) = x - \frac{\sin x - x^3}{\cos(x) - 3x^2}$ (columns 1 and 4, respectively), the algorithm does converge to 0.92862630873173 with the error ratio μ_n far less than 1. However, when we choose $g(x) = \frac{\sin x}{x^2}$, the error ratio is greater than 1 and the iteration do not converge. For $g(x) = x + \sin x - x^3$, the error ratio is very close to 1. It appears that the algorithm does converge to the correct value, but very slowly.

Why is there such a disparity in the rate of convergence?

2 Error Analysis of Fixed Point Iterations

In order to carry our analysis of fixed point iterations, we assume that g is differentiable. Note the continued theme: the more we restrict a function, the better our tools are for analyzing the function

numerically. We previously assumed g to be continuous; now we raised the bar and assume it to be differentiable.

We would like to see how g is behaving in the area around the solution, r. For this, we will use the Taylor expansion with remainder. Remember that, for any differentiable function g, and for any two values x, a, there exists c such that

$$g(x) = g(a) + g'(c) \cdot (x - a). \tag{9}$$

We don't know what the precise value of c is, but we know that it exists and is between a and x.

2.1 Testing for Divergence

Substituting x_n for x and r (the analytic solution) for a, we can use Equation 9 to provide a test for divergence:

$$\begin{aligned} e_{n+1} &= x_{n+1} - r \\ &= g(x_n) - g(r) \\ &= (x_n - r) \cdot g'(c_n) \quad c_n \text{ between } x_n \text{ and } r \\ &= e_n \cdot g'(c_n) \end{aligned} \tag{10}$$

As x_n approaches r, c_n (which is between x_n and r) is getting closer to r, and, therefore $g'(c_n) \approx g'(r)$. This means that if $|g'(r)| > 1$, once x_n gets close enough to r, the error will get larger, and the sequence $x_0, x_1, x_2, \ldots$ will *never* converge.

This suggests a straightforward test to see if a particular function g is a poor choice for fixed point iterations. Simply check the numerical values of g' in some interval that is known the contain the solution r. If $|g'(x)| > 1$ on this interval, then the sequence of $x_0, x_1, x_2, \ldots$ will not converge and g is a poor choice.

For example, if we look at $g(x) = \frac{\sin x}{x^2}$ for $x \in [.9, 1]$, we see that $|g'(x)|$ is greater than 1 over the interval. If we look at $g(x) = \sqrt[3]{\sin x}$ on the same interval, its derivative is around .23 there. This explains why the choice $g(x) = \frac{\sin x}{x^2}$ failed in our experiment and $g(x) = \sqrt[3]{\sin x}$ produced nice results.

2.2 Testing for Convergence

We now have a way to test for divergence, but what about testing for convergence? We can use local analysis around the actual solution r to determine the relevant behavior of g. Define the interval $I = [x_0 - \delta, x_0 + \delta]$ for some small constant δ with the analytic solution $r \in I$. If we can guarantee that the magnitude of the derivative is less than 1 over I:

$$\forall_{c \in I} \left| g'(c) \right| \leq \lambda < 1$$

for some constant λ, then we will have convergence, as long as our values x_i stay in the interval I. We can quantify this with the following two claims:

Claim 2.1. *If $\forall_{c \in I}\, 1 > \lambda \geq g'(c) \geq 0$ then the sequence $x_0, x_1, x_2, \ldots$ stays in interval I, and will converge to the root r.*

Proof. Assume that x is our present approximation of r, and assume by induction that is lies in the interval I. Our next approximation is $g(x)$ and once we show that it also lies in the interval, our inductive proof will go through (our initial x_0 lies in I by assumption!). Now, our error analysis shows that $g(x) = g(r) + (x - r)g'(c)$, for some c between x and r. Now, $x \in I$, by induction, and $r \in I$ by assumption, hence $c \in I$, hence $0 < g'(c) < 1$. Also, $g(r) = r$. Altogether,

$$g(x) = r + (x - r)g'(c) = r(1 - g'(c)) + xg'(c),$$

so, $g(x)$ is an average of r and x, with positive weights, hence lies between x and r hence lies in I.

The above induction proves that all the iterative values $x_0, x_1, \ldots$ lie in I. Our error analysis then can be applied to show that we have convergence, since everywhere in I $g' < \lambda < 1$. □

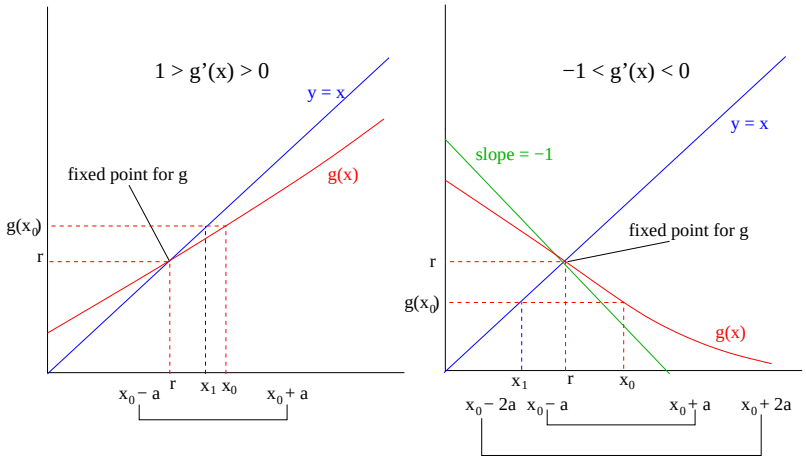

Figure 2: Examples of Convergence for Functions g with Positive and Negative Slope

If the derivative of g is not positive on I, it is possible that the iterations do not stay inside I, and hence the fact that g' is small enough on I may not suffice. However, once we assume that $r \in I$ (as before), while g' is small enough on a slightly larger interval, convergence can be guaranteed:

CLAIM 2.2. *If $r \in I$ as before, and, with $I' := [x_0 - 2\delta, x_0 + 2\delta]$, $\forall_{c \in I'} |g'(c)| \leq \lambda < 1$, then $x_0, x_1, x_2, \ldots$ converges to r.*

Proof. We need to prove that the iterations $x_0, x_1, \ldots$ stay in I': since g' is small enough on I', our previous error analysis will show that the error decreases by at least a factor of λ per iteration.

Now, x_0 surely lies in I'. Assume by induction that $x_0, \ldots, x_{n-1}$ all lie in I'. Then, the errors $e_0, e_1, \ldots e_n$ decrease (in magnitude), hence $|e_n| < |e_0| < \delta$. This means that x_n lies in $(r - \delta, r + \delta)$. The latter interval may not be a subset of I, but is surely a subset of I' since $r \in I$. So, $x_n \in I'$, hence $|e_{n+1}| < \delta$, and our induction continues successfully. □

Examples of these two situations are depicted in Figure 2.

The discussion made clear that we would like the magnitude $|g'(r)|$ to be small (and we need that to be true not only at r but also around r). So, what is the "best" value for $g'(r)$? It makes sense to think that the best is $g'(r) = 0$. In the next section we will show that this property leads to quadratic convergence.

2.3 Testing for Convergence

First, a small aside. One might question the value of all this convergence analysis in practice. For example, how can we test the derivative near the root when, by assumption - since this is what we're trying to find - we don't know where the root is? The answer is that often even when we don't know the precise location of the root (say, to 10 decimal places), we can easily get a rough estimate of where the root is. For example, we may not know an exact value for $\sqrt{5}$, but we know that $\sqrt{5} \in [2.2, 2.3]$ (well, those of us that do not know this, should know that $\sqrt{5} \in [2, 2.5]$...) We can use this estimate to check the derivative. For example, let our fixed point equation be

$$x = \frac{x^2 + 5}{2x} = x/2 + 2.5/x,$$

(whose solution is $\sqrt{5}$). Then

$$g'(x) = 1/2 - 2.5/x^2.$$

Now, choose $x_0 := 2.25$, and take $\delta = 1/8$. Then $I' = [2, 2.5]$, and if $x \in I'$, then $g'(x) \in [-.125, .1]$. Also, $I = [2.125, 2.375]$. So, it is enough to know that $\sqrt{5} \in I$ (indeed), to invoke our test theorem above and to conclude that the iterations will converge (fast, since g' is really small on I').

So the theory does have a practical value.

3 Quadratic Convergence of Fixed Point Iterations

Let $x = g(x)$ have solution r with $g'(r) = 0$. Further, assume that g is doubly differentiable on the interval $I = [x_0 - \delta, x_0 + \delta]$, with $r \in I$. Then, from the Taylor expansion of g we know for any $x, a \in I$, there is some c between x and a such that:

$$g(x) = g(a) + g'(a) \cdot (x - a) + g''(c)\frac{(x-a)^2}{2} \tag{11}$$

Using our equation for error and substituting Equation 11 yields:

$$\begin{aligned} e_{n+1} &= x_{n+1} - r \\ &= g(x_n) - g(r) \\ &= g'(r)(x_n - r) + g''(c_n)\frac{\overbrace{(x_n - r)}^{e_n}{}^2}{2} && (\text{some } c_n \in I) \\ &= g''(c_n)\frac{e_n^2}{2} && (\text{since } g'(r) = 0) \end{aligned}$$

which is quadratic convergence!

Note that, in order to obtain quadratic convergence, we need two things:

1. An improved algorithm (fixed point iterations with a special function g with $g'(r) = 0$).
2. Improved regularity of g (g must be twice differentiable).

These are two independent issues. Constructing a transformation of the original equation $f(x) = 0$ to $x = g(x)$ with $g'(r) = 0$ is dependent only on our sophistication in transforming f, and is under our control. However, the regularity of g is something that we have little control over. If g is twice differentiable, it will, almost surely, imply that f has the same property. If f is a "bad"

function (i.e. not differentiable enough), only a miracle will produce a doubly differentiable g. Without such a miracle, g will not be regular enough, and the error analysis which led to quadratic convergence will no longer be valid. Moral: do not expect that good algorithms perform well on bad functions. In particular, if f is not twice differentiable, do not waste your time to find g with $g'(r) = 0$, since you are not likely to get quadratic convergence.