Chapter 3. Nonlinear Equations in one variable

In this chapter we will design algorithms to approximate solutions to nonlinear scalar equations

$$f(x) = 0$$

where the variable x runs in an interval $[a, b]$. The topic provides an opportunity to discuss various issues and concepts that arise in more general circumstances.

3.1 Solving Nonlinear Equations

Methods to solve $f(x) = 0$ for x.

- Not realistic to expect closed form formula like quadratic formula.
- Must resort to iterative methods.
- start with x_0 then generate $x_1, x_2, \ldots$

3.1 Solving Nonlinear Equations

Methods to solve $f(x) = 0$ for x.

- Why is it necessary to to know more than one method?
- Answer 1: Cost of evaluating f
- Answer 2: Cost of evaluating f'.
- Try MATLAB: "help" *fzero*

3.1 Solving Nonlinear Equations

Desirable qualities in methods to solve $f(x) = 0$ for x.

- Efficient–requires small number of function evaluations.
- Robust–fails rarely.
- Requires minimal smoothness of f.
- Does not need much additional data, like f'.
- Generalizes to multivariable solutions $f(x, y) = 0$.

3.2 Bisection Method

Let f be continuous on $[a, b]$, satisfying $f(a)f(b) < 0$. Then f has a root between a and b, that is, there exists a number x^* in $[a, b]$ so that $f(x^*) = 0$ by the IVT.

3.2 Bisection Method

- Use Bisection Method to approximate root of $f(x) = x^3 + x - 1$.
- Start with $a = -1, b = 3$.
- Use two iterations.
- Use python code.

3.2 Bisection Method

- When to stop?
- $|x_n - x_{n-1}| <$ atol.
- $|x_n - x_{n-1}| <$ rtol $|x_n|$.
- $|f(x_n)| <$ ftol.

3.2 Bisection Method

- How accurate and how fast is Bisection?
- Solution error $= |x^* - x_n| < \frac{b-a}{2^{n+1}}$.
- A solution is correct to within p decimal places if the error is less than 0.5×10^{-p}.
- Use bisection method to find a root of $f(x) = \cos x - x$ in the interval $[0, 4]$ to within six correct decimal places.

3.3 Fixed-Point Iteration

- Use python to apply cos to an arbitrary starting number.
- The real number x^* is a fixed point of g if $g(x^*) = x^*$.
- FPI: initial guess x_0 and $x_{k+1} = g(x_k)$.
- $r = 0.7390851332$ is a (approx.) fixed-point of $g(x) = \cos x$.
- Notice similarity with finding root x^* of $f(x) = \cos x - x$.

3.3 FPI

Fixed points are roughly divided into three classes

- Convergent: Ex $g(x) = \pm\frac{1}{2}x$.
- Divergent: Ex: $g(x) = \pm 3x$.
- Stable: Ex: $g(x) = -x$.
- Geometry using cobweb (staircase) diagrams (figure 3.2 on page 47).

3.3: FPI Error Estimates

- Example: $g(x) = \frac{1}{4}x + 2$.
- $x_0 = 0$.
- $|e_k| = |x_k - x^*| = (\frac{1}{4})^k |x_0 - x^*| = (\frac{1}{4})^k \frac{8}{3}$
- Linear convergence with rate $\frac{1}{4}$.

3.3 Nonlinear FPI

- Convergent: Ex $g(x) = \cos x$.
- Convergent and Divergent: Ex: $g(x) = x^3$.
- Lookup Wikipedia: Fixed point (mathematics)
- Geometry using cobweb (staircase) diagrams.

3.3 Theorem Fixed Point Convergence

Let $g(x)$ be continuously differentiable and $x^* = g(x^*)$ is a fixed point. If $|g'(x)| \leq \rho < 1, \forall x \in (a, b)$ then FPI converges linearly to x^* with rate $\log_{10} \rho$, On the other hand if $|g'(x^*)| > 1$, then x^* is an unstable fixed point.

3.3 FPI

Proof of convergence of FPI uses INEQUALITIES to find a BOUND on error! This is the best we can do. We cannot determine the error exactly; we can only hope to find a bound on it.

- $e_i = |x_i - x^*|$
- Contraction by factor ρ by MVT: $e_{i+1} \leq \rho e_i$
- Iterate: $e_i \leq \rho^i e_0$

3.3 FPI

Example:

- $g(x) = 2.8x - x^2$.
- $x_1^* = 0, x_2^* = 1.8$.
- $g'(x_1^*) = 2.8, g'(x_2^*) = -.8$.
- cobweb (staircase) diagram.

3.3 FPI: Example

Consider the FPI $x \to g(x) = x^2 - 0.24$. (a) Do you expect FPI to calculate the root -0.2, say, to 10 or correct decimal places, faster of slower than the Bisection Method? (b) Find the other fixed point. Will FPI converge to it?

3.3 FPI Example

Apply FPI to find the solution of each equation to 8 correct decimal places.

- $e^x + x = 7$
- $x^5 + x = 1.$

3.3 FPI (Efficieny)

- Linear Convergence: $|e^{(k+1)}| \leq \rho |e^{(k)}|$.
- Huge advantage to arrange that ρ from the Fixed Point Theorem be small.
- $g'(x) \leq .9$ then it takes about 22 iterates to stabilize a digit.
- $g'(x) \leq .1$ then it takes about 1 iterate to stabilize a digit.
- Fastest Convergence is Quadratic Convergence when $|g'(r)| = 0$.

3.4 Newton's Method

Suppose that f is twice continuously differentiable and $f(r) = 0$. If $f'(r) \neq 0$, then Newton's method

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

is *locally quadratically convergent* to x^*, i.e. $e_{k+1} \leq Me_i^2$.

3.4 Newton's Method

Start with your e_0. Say your friend has a lucky initial guess and her initial error is $\frac{e_0}{2}$.

- Linear Convergence with ρ : all of your friend's subsequent errors will be approximately $\frac{1}{2}$ of your errors.
- Quadratic Convergence (say $M = 1$) : your friend's first error will be your $\frac{e_1}{4}$ and her second error will be your $\frac{e_2}{16}$ ....

3.4 Newton's Method (Convergence Speeds)

Start with x_0 having one decimal place accuracy.

- Linear Convergence (and $\rho = .1$) : It requires 49 iterations to obtain x_{49} with 50 decimal place accuracy.
- Quadratic Convergence (and $M = 1$) : It requires 6 iterations to obtain x_6 with 64 decimal place accuracy.
- In practice one does not compute ρ or M precisely; one approximates.

3.4 Newton's Method (Examples)

- Approximate $\sqrt{2}$.
- Solve $0 = x^3 + x - 1, x_0 = -0.7$.
- Geometry of Newton's method.

3.4 Newton's Method (Convergence and Roundoff Error)

One of the goals of numerical analysis is to compute answers within a specified level of accuracy. Working in double precision means that we store and operate on numbers that are kept to about 16 digits of accuracy. Can answers always be computed to 16 correct significant digits?

3.4 Newton's Method (Convergence and Roundoff Error)

- Use Bisection to approximate root of $f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$ to within 6 correct significant digits.
- Exact Answer: $x = \frac{2}{3} = 0.666666\ldots$.
- The problem is that matlab thinks it has found a root $x = 0.666664123535156$.

3.4 Newton's Method (Convergence and Roundoff Error)

Assume that f is a function and that x^* is a root. Assume that x_k is an approximate for x^*. The *backward error* of the approximation x_k is $|f(x_k)|$ and the *forward error* is $e_k = |x_k - x^*|$. The forward error is the only error we have studied before today. We used to call the forward error simply *the error*.

3.4 Convergence and Roundoff Error

Now that we have two types of error to discuss, we can explain what happened in the previous problem. Bisection search was unable to approximate 6 significant digits of the solution because the backward error became much smaller than the forward error and the Bisection method stopped; this is called *the stopping criteria*.

3.4 Stopping an Iterative Procedure

- Use one or both of the following
- $|x_n - x_{n-1}| < \text{atol}$
- $|f(x_n)| < \text{ftol}$.

3.4 Convergence and Roundoff Error

The usage of "backward" and "forward" can be viewed by thinking of the "domain" of the problem (the function $f(x) = x^3 - 2x^2 + \frac{4}{3}x - \frac{8}{27}$ in the case of $f(x) = 0$) as the input of our method and the output as the solution ($x_k = 0.666664123535156$ in this case).

3.4 Convergence and Roundoff Error

The subject of backward and forward error is relevant to stopping criteria for equation solvers. The goal is to find the root x^* satisfying $f(x^*) = 0$. How do we determine if x_k is a good approximation? Two possibilities: (1) "forward" make $|x_k - x^*|$ small or (2) "backward" make $|f(x_k)|$ small.

3.4 Convergence and Roundoff Error

Whether forward or backward error is appropriate depends on the problem. For the Bisection method both errors are easy to bound.

3.4 Root Finding Without Derivatives

- Secant Method
- x_0, x_1 two initial guesses.
- $x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$.
- Use secant method to approximate $\sqrt{2}$ with initial guesses $x_0 = 0, x_1 = 1$.

3.4 Globalizing Newton's Method

Both Newton's method and the secant method are guaranteed to converge only locally: x_0 must be "close enough". Contrast this with the Bisection method which guarantees convergence with $\rho = 0.5$. A natural idea is to combine the methods into a hybrid one (see MATLAB's *fzero* commmand and numpy's *fsolve* command).

3.4 Minimizing a function in one variable

We will not cover this section this semester.