

**CS537**

**Numerical Analysis  
and Computing**

**Lecture 2**

**Locating Roots of Equations**

Professor Jun Zhang

Department of Computer Science  
University of Kentucky  
Lexington, KY 40206-0046

September 9, 2008

## **What is the Root**

Many physical system can be written in the form of an equation

This equation represents the relationship between the dependent variables and independent variables

The root of a nonlinear equation is the solution of that equation

It can also be said to the solution of a nonlinear system

Most nonlinear equations are too complicated to have an analytical solution

In practice, we are more interested in finding some numerical solutions

Explicit numbers, approximate solutions,

## Roots of a Functions

Let  $f(x)$  be a function that has values of opposite signs at the two ends of a given interval  $[a,b]$  with  $a < b$ , i.e.,  $f(a) \cdot f(b) < 0$ . If  $f(x)$  is continuous on  $[a,b]$ , then there exists a number  $c$  in  $[a,b]$  such that  $f(c) = 0$

$c$  is called a root of function  $f(x) = 0$

**Example.** The function

$$f(x) = x^2 + 2x - 3 = 0$$

has a root in the interval  $[0,2]$ . It has two roots in the interval  $[-5,2]$

Remark: roots are not necessarily unique in a given interval

Need some root finding algorithms for general functions

## Bisection Method

Given an interval  $[a, b]$  and a continuous function  $f(x)$ , if  $f(a) \cdot f(b) < 0$ , then  $f(x)$  must have a root in  $[a, b]$ . How to find it?

We suppose  $f(a) > 0$  and  $f(b) < 0$

Step 1. Compute the midpoint  $c = \frac{b+a}{2}$ , stop if  $\frac{|b-a|}{2}$  is small, and take  $c$  as the root

Step 2. Evaluate  $f(c)$ , if  $f(c) = 0$ , a root is found

Step 3. If  $f(c) \neq 0$ , then either  $f(c) > 0$  or  $f(c) < 0$

Step 4. If  $f(c) < 0$ , a root must be in  $[a, c]$

Step 5. Let  $b \leftarrow c, f(b) \leftarrow f(c)$ , go to Step 1.

## Convergence Analysis

Let  $r$  be a root of  $f(x)$  in the interval  $[a_0, b_0]$ . Let  $c_0 = \frac{a_0 + b_0}{2}$  be the midpoint, then

$$|r - c_0| \leq \frac{b_0 - a_0}{2}$$

If we use the bisection algorithm, we compute and have  $a_0, b_0, c_0, a_1, b_1, c_1, \dots$ , then

$$|r - c_n| \leq \frac{b_n - a_n}{2} \quad (n \geq 2)$$

Since the interval is halved at each step, we have

$$b_n - a_n = \frac{b_{n-1} - a_{n-1}}{2} = \dots = \frac{b_0 - a_0}{2^n}$$

Hence

$$|r - c_n| \leq \frac{b_0 - a_0}{2^{n+1}}$$

which is the maximum error if we take  $c_n$  as an approximate to the root  $r$

## Linear Convergence

A sequence  $\{x_n\}$  has linear convergence to a limit  $x$  if there exists a constant  $C$  in the interval  $[0,1)$  such that

$$|x_{n+1} - x| \leq C|x_n - x| \quad (n \geq 1)$$

By recursion, we have

$$\begin{aligned} |x_{n+1} - x| &\leq C|x_n - x| \leq C^2|x_{n-1} - x| \\ &\leq \dots \leq C^n|x_1 - x| \end{aligned}$$

Or equivalently, a linear convergence satisfies

$$|x_{n+1} - x| \leq AC^n \quad (0 \leq C < 1)$$

For some positive number  $A$

The bisection algorithm has a linear convergence rate with  $C = \frac{1}{2}$  and  $A = b_0 - a_0$

## Stopping Criterion

What is our goal? When to stop? How many iterations?

Our goal is to find  $r \in [a,b]$  such that  $f(r) = 0$

With bisection algorithm, we generate a sequence such that

$$|r - c_n| < \varepsilon$$

For some prescribed number  $\varepsilon > 0$

i.e., we find a point  $c_n$  inside the interval  $[a,b]$  that is very close to the root  $r$ . We then use  $c_n$  as an approximate to  $r$

It is not guaranteed, however, that  $f(c_n)$  is *very close* to 0

## How Many Iterations

If we want the approximate root  $c_n$  is close to the true root  $r$ , i.e., we want

$$|r - c_n| < \varepsilon,$$

Then the number of bisection steps  $n$  satisfies

$$\frac{b - a}{2^{n+1}} < \varepsilon$$

Or

$$n > \frac{\log(b - a) - \log(2\varepsilon)}{\log 2}$$

**Example.** Find a root in  $[16,17]$  up to machine single precision

$a = (10\ 000.0)_2$ ,  $b = (10\ 001.0)_2$  so  $r$  must have a binary form

$r = (10\ 100.***...)_2$ . We have a total of **24** bits, **5** is already fixed. The accuracy will be up to another **19** bits, which is between  $2^{-19}$  and  $2^{-20}$ . We choose  $\varepsilon = 10^{-20}$ . Since  $b - a = 1$ , we need  $2^{n+1} > 2^{20}$ , yielding  $n \geq 20$

## Newton's Method

Given a function  $f(x)$  and a point  $x_0$ , if we know the derivative of  $f(x)$  at  $x_0$ , we can construct a linear function that passes through  $(x_0, f(x_0))$  with a slope  $f'(x_0) \neq 0$  as

$$l(x) = f'(x_0)(x - x_0) + f(x_0)$$

Since  $l(x)$  is close to  $f(x)$  at  $x_0$ , if  $x_0$  is close to  $r$ , we can use the root of  $l(x)$  as an approximate to  $r$ , the root of  $f(x)$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$x_1$  may not be close to  $r$  enough, we repeat the procedure to find

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}, \dots$$

Under certain conditions,  $\{x_n\}$  converges to  $r$

## From Taylor Series

If  $f(x_0) \neq 0$ , but  $x_0$  is close to  $r$ , we may assume that they differ by  $h$ , i.e.,  $x_0 + h = r$ , or

$$f(x_0 + h) = f(r) = 0$$

Using Taylor series expansions

$$f(x_0) + hf'(x_0) + \frac{h^2}{2} f''(x_0) + \dots = 0$$

Ignoring the higher order terms, we have

Or 
$$f(x_0) + hf'(x_0) = 0$$

$$h = -\frac{f(x_0)}{f'(x_0)}$$

Since  $h$  does not satisfy  $f(x_0 + h) = 0$ , we use

$$x_1 = x_0 + h = x_0 - \frac{f(x_0)}{f'(x_0)}$$

as an approximate to  $r$ , and repeat the process

## Fast Convergence

Find a root for the following function  $f(x)$ , starting at  $x_0 = 4$

$$f(x) = x^3 - 2x^2 + x - 3$$

$$f'(x) = 3x^2 - 4x + 1$$

$n$	$x_n$			$f(x_n)$
0	4.0			33.0
1	3.0			9.0
2	2.4375			2.04
3	2.21303	27224	731445	0.256
4	2.17555	49386	143684	$6.46 \times 10^{-3}$
5	2.17456	01006	550714	$4.48 \times 10^{-6}$
6	2.17455	94102	932841	$1.97 \times 10^{-12}$

Each iteration gains double digits of accuracy and  $f(x_n)$  decreases quadratically to **0**

## Convergence Analysis

Let the function  $f$  have continuous first and second derivatives  $f'$  and  $f''$ , and  $r$  be a simple root of  $f$  with  $f'(r) \neq 0$ . If  $x_0$  is *sufficiently* close to  $r$ , then Newton's method converges to  $r$  quadratically.

$$|x_{n+1} - r| \leq C|x_n - r|^2$$

If  $x_n$  differs from  $r$  by at most one unit in the  $k$ th decimal place, i.e.,

$$|x_n - r| \leq 10^{-k}$$

Then, for  $c = 1$ , we have

$$|x_{n+1} - r| \leq 10^{-2k}$$

The number of correct decimal digits doubles after another iteration

## Convergence Proof

Let  $e_n = r - x_n$ . Newton's method gives a sequence  $\{x_n\}$  such that

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

Using Taylor's expansion, there exists a point  $\xi_n$  between  $x_n$  and  $r$  for which

$$\begin{aligned} 0 &= f(r) = f(x_n + e_n) \\ &= f(x_n) + e_n f'(x_n) + \frac{1}{2} e_n^2 f''(\xi_n) \end{aligned}$$

It follows that

$$e_n f'(x_n) + f(x_n) = -\frac{1}{2} e_n^2 f''(\xi_n)$$

## Convergence Proof Cont.

We thus have

$$e_{n+1} = -\frac{1}{2} \left( \frac{f''(\xi_n)}{f'(x_n)} \right) e_n^2$$

Define an upper bound

$$c(\delta) = \frac{1}{2} \frac{\max_{|x-r| \leq \delta} |f''(x)|}{\min_{|x-r| \leq \delta} |f'(x)|}, \quad (\delta > 0)$$

We can choose  $\delta$  small so that

$$|e_n| = |x_n - r| \leq \delta \quad \text{and} \quad |\xi_n - r| \leq \delta$$

This is to guarantee that  $x_n$  is close to  $r$  within a distance of  $\delta$

## Convergence Proof Cont.

For very small  $\delta > 0$ , we have

$$\begin{aligned} |e_{n+1}| &= \frac{1}{2} \left| \frac{f''(\xi_n)}{f'(x_n)} \right| e_n^2 \leq c(\delta) e_n^2 \\ &\leq \delta c(\delta) |e_n| = \rho |e_n| \end{aligned}$$

With  $\rho = \delta c(\delta) < 1$  if  $\delta$  is small enough, therefore

$$|x_{n+1} - r| = |e_{n+1}| \leq \rho |e_n| \leq |e_n| \leq \delta$$

$x_{n+1}$  is also close to  $r$  within a distance of  $\delta$ . By recursion, if  $x_0$  is close to  $r$ , then

$$|e_n| \leq \rho |e_{n-1}| \leq \rho^2 |e_{n-1}| \leq \dots \leq \rho^n |e_0|$$

Since  $\rho < 1$ , this is to say

$$\lim_{n \rightarrow \infty} |e_n| = 0 \quad \text{as} \quad n \rightarrow \infty$$

## **Weakness of Newton's Method**

Newton's method converges fast, only when  $x_0$  is chosen close to  $r$ . In practice, there might be a number of problems also

- 1.) needs derivative value and availability
- 2.) starting point must be close to  $r$
- 3.) lose quadratic convergence if multiple root
- 4.) iterates may runaway (not in convergence domain)
- 5.) flat spot with  $f'(x_n) = 0$
- 6.) cycling iterates around  $r$

## Systems of Nonlinear Equations

Newton's method is really useful for finding zero of a system of nonlinear equations

$$f_1(x_1, x_2, \dots, x_n) = 0$$

$$f_2(x_1, x_2, \dots, x_n) = 0$$

$$\vdots \quad \vdots$$

$$f_n(x_1, x_2, \dots, x_n) = 0$$

Written in vector form as

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

Where

$$\mathbf{f} = (f_1, f_2, \dots, f_n)^T$$

$$\mathbf{x} = (x_1, x_2, \dots, x_n)^T$$

We have

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - [\mathbf{f}'(\mathbf{x}^{(k)})]^{-1} \mathbf{f}(\mathbf{x}^{(k)})$$

$\mathbf{f}'(\mathbf{x}^{(k)}) = \mathbf{J}(\mathbf{x}^{(k)})$  is the Jacobian matrix

## A 3 Equation Example

$$f_1(x_1, x_2, x_3) = 0$$

$$f_2(x_1, x_2, x_3) = 0$$

$$f_3(x_1, x_2, x_3) = 0$$

Using Taylor expansion

$$\begin{aligned} & f_i(x_1 + h_1, x_2 + h_2, x_3 + h_3) \\ &= f_i(x_1, x_2, x_3) + \\ & \quad h_1 \frac{\partial f_i}{\partial x_1} + h_2 \frac{\partial f_i}{\partial x_2} + h_3 \frac{\partial f_i}{\partial x_3} + \dots \end{aligned}$$

Let  $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)}, x_3^{(0)})^T$  be an approximate solution and the computed correction be  $\mathbf{h} = (h_1, h_2, h_3)^T$ . Hence

$$\mathbf{0} \approx \mathbf{f}(\mathbf{x}^{(0)} + \mathbf{h}) = \mathbf{f}(\mathbf{x}^{(0)}) + \mathbf{f}'(\mathbf{x}^{(0)})\mathbf{h}$$

## Example Cont.

The Jacobian matrix is

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix}$$

It follows that

$$\mathbf{h} \approx -[\mathbf{f}'(\mathbf{x}^{(0)})]^{-1} \mathbf{f}(\mathbf{x}^{(0)})$$

Hence, the new iterate is

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - [\mathbf{f}'(\mathbf{x}^{(0)})]^{-1} \mathbf{f}(\mathbf{x}^{(0)})$$

In practice, we solve the Jacobian matrix in

$$[\mathbf{J}(\mathbf{x}^{(k)})] \mathbf{h}^{(k)} = -\mathbf{f}(\mathbf{x}^{(k)})$$

So that

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{h}^{(k)}$$

## Secant Method

In Newton's method

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

We need to evaluate  $f(x_n)$  and  $f'(x_n)$  at each iteration

We can approximate the derivative at  $x = x_n$  by

$$f'(x_n) \approx \frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n}$$

Thus, the secant method generates iterates

$$x_{n+1} = x_n - \left( \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \right) f(x_n)$$

Only one functional evaluation at each iteration

## Comments

Secant method needs two iterates to start with, can use bisection method to generate the second iterate

Secant method does not need to know the derivative of  $f(x)$

If  $|f(x_n) - f(x_{n-1})|$  is small, the computation may lose significant digits and becomes unstable

The convergence rate of secant method is superlinear

$$|e_{n+1}| \leq C|e_n|^\alpha$$

With  $\alpha = \frac{1}{2}(1 + \sqrt{5}) \approx 1.62$ . Its convergence rate is between that of bisection method and the Newton's method

# Hybrid Approaches

In practice, hybrid methods are usually used

For example, we can use Bisection Method to generate initial iterates that are close to the root, so that Newton's method can be applied

When the evaluation of derivative is expensive, the Secant Method should be used to replace the Newton's method

The trade-offs between Bisection Method and Newton's Method are robustness and fast convergence

The Secant Method is supposed to come between these two methods, to take the advantages of both, and avoid the disadvantages of either