

**MS554 Lecture Notes: Solving Non-linear Equations (II)**  
Lecture 6; Sept. 20, 2001

## Objectives:

- Fixed-point iteration, and convergence of algorithms (see lecture 6 notes).
- Example: calculation of wave-length in intermediate water depths.
- **Materials:** Hornberger and Wiberg, Chapter 2; Gerald and Wheatley, Chapter 1; Forsythe et al., Chapter 7; Golub and Ortega, Chapter 5.

## Loose ends from previous lecture

1. We talked before about solving the general non-linear equation: find  $x$  such that  $g(x) = h(x)$ ; where  $g(x)$  and  $h(x)$  are any functions. This can always be rewritten to be “find  $x$  such that  $f(x) = 0$ , so that the problem becomes one of finding the root(s) of  $f(x)$ .”
2. The problem can be further generalized by letting  $x$  be a vector of dimension  $N \times 1 = [x_1, x_2, x_3, \dots, x_N]$ . The function  $f(x)$  is then a series of functions  $f_i(x)$ , and to be well-posed, you’d need  $N$  functions, which could be represented as a matrix  $F(x)$ . The problem would then be to find vectors  $x$  such that  $F(x)=0$ , and you’d be solving a *system of non-linear equations*.
3. Another generalization of the problem is to allow that the roots of  $f(x)$  may be complex. For example,  $x^2 + 1$  has roots of  $\pm i$ .
4. Some of the methods that we covered last week can be used to find complex roots (Newton’s and secant methods), and some can’t (bisection method). Some prove useful for solving systems of non-linear equations (Newton’s method).
5. *Global convergence* means that the method converges no matter how far away the initial guess is from the root.
6. The choice of criteria for testing convergence can either test that the function evaluation ( $f(x_n)$ ) is sufficiently close to zero, or that successive iterations are close ( $|x_{n+1} - x_n| < \epsilon$ ). Neither are perfect.

## Convergence of bisection method

The bisection method is linearly convergent:  $e_{n+1} \leq e_n/2$ ; the error term is halved with every iteration (recall that  $e_n = x_n - x_r$ ). This means that, for a binary-based floating point number, you’d get an extra digit of precision with every iteration.

## Convergence of Newton's method

*Note: see notes from lecture 6*

Newton's method is guaranteed to converge "close enough" to the root. Golub and Ortega (page 157) show that the region of convergence can be guaranteed in the neighborhood in which  $f(x)$  is *convex* around the root  $x_r$ . A convex function is one that satisfies

$$\begin{aligned} f''(x) &\geq 0; \text{ for all } x, \\ f'(y) &\geq f'(x), \text{ if } y > x, \\ f(\alpha x + (1 - \alpha)y) &\leq \alpha f(x) + (1 - \alpha)f(y); \end{aligned} \quad (1)$$

for any  $\alpha$  in  $(0, 1)$ , and any  $x$  and  $y$ . Geometrically, convex functions "bend upwards". If  $-f(x)$  is convex, then  $f(x)$  is concave, and global convergence still holds. If the function is convex within a region around the root, then Newton's method will converge if your initial guess is within that region.

## Example: estimate wave length

*From Hornberger and Wiberg*

One non-linear problem that you may face someday is to estimate wavelength ( $L$ ) given wave period ( $T$ ), and water depth ( $h$ ). The wavenumber,  $k$ , is defined  $k = 2\pi/L$ . The wave speed,  $c$ , equals  $gT/(2\pi)$  in *deep* water, and satisfies  $c = L/T$ . If  $k_d$  equals the wave number in deepwater, then the wave number in shallower water should satisfy;

$$\frac{k_d}{k} \approx \tanh(kh) \quad (2)$$

where  $\tanh$  is the hyperbolic tangent. If we know that in deep water ( $h/L > 1/2$ ), the wave period is 15 s, then  $L_d = gT^2/2\pi$ , and  $k_d = 2\pi/L = XXX$ . We can find the wavelength,  $k$ , in a water depth of say, 20 m, by using some of the techniques that we've learned.

The problem lends itself to fixed-point iteration;  $k = g(k)$  where  $g(k) = \frac{k_d}{\tanh(kh)}$ . For wave period of 15 s, I get a wavenumber of 0.0318 for a water depth of 20m, equal to a wavelength of 197.4 m. Using this relationship, I got convergence within 42 iterations.

The problem can also be solved using the secant method, and I got convergence to the same degree of tolerance within 6 iterations.