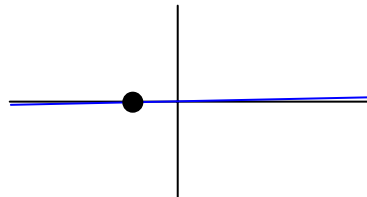


## CS205 - Class 7

*Covered in class:* 1, 4, 5, 7, 8

*Readings:* Heath Chapter 5

1. Nonlinear equations are much more difficult to solve than simple linear equations, thus we will move from systems of equations to scalar equations to get started.
  - a. Before we move from linear to nonlinear equations, let's first move from systems of equations to scalar equations in the linear case to get warmed up.
    - i. consider  $ax=b$  where  $a$  and  $b$  are merely numbers
      1. of course the solution to this is simply  $x=b/a$
      2. the only worry here would be if  $a$  was small or "zero" in which case the  $\det(A)=\det(a)$  is zero and the matrix  $[a]$  is essentially singular or near singular
    - ii. let's take a different, functional look at this
      1. assume that we had a straight line  $y=ax+b$  and that we wanted to find the roots where  $y=0$ , then we need to solve  $0=ax+b$  or  $ax=-b$
      2. This is our linear equation with solution  $x=-b/m$  and now we see that a small slope is equivalent to ill-conditioning
        - a. Consider  $y=(1e-16)x+1e-16$  where the root is  $x=1$  but the division  $x=-(1e-16)/(1e-16)$  looks like  $0/0$  and is indeterminate and full of errors.



- b.
  - b. Now for nonlinear equations we are solving  $A(x)=b$  where  $A$  is a vector valued function of  $x$ .
    - i. Moreover we can write  $A(x)=0$  since the  $b$  term can be incorporated into the vector valued function  $A$  of  $x$ .
    - ii. As a scalar equation, we have  $a(x)=0$  which looks odd, so we switch notation to  $f(x)=0$ .
    - iii. Note how this looks like a root finding problem. In general, any  $g(x)=b$  can be rewritten as  $f(x)=0$  by moving  $b$  to the left hand side.
      1. There may be any number of roots from 0,1,2, etc. to infinity.
      2. Remember "roots"="solutions" from here on out.

iv. Since  $f'(x^*)$  is the slope of the function  $f$  at and “near” a root  $x^*$ ,  $f'(x^*)$  gives us an indication of the condition number. That is, it takes the place of “a” in the linear case.

1. If  $f'(x^*)$  is small or “zero” then we could be in trouble. That is, the condition number is large. Moreover the function is locally *flat*.
2. A simple root has  $f'(x^*) \neq 0$ .
3. A multiple root has  $f'(x^*) = 0$  which could be problematic.

2. We usually solve nonlinear equations with iterative methods:  $x_1, x_2, x_3 \dots x_n$  stopping when the error,  $e_k = x_k - x_{exact}$ , is small, i.e. when  $\|e_k\| < \varepsilon$ .

a. The **convergence rate** is determined by looking at the equation  $\|e_{k+1}\| = C\|e_k\|^r$  with  $C \geq 0$  in the limit as  $k \rightarrow \infty$ .

- i. If  $r=1$ , then we need  $C < 1$  to guarantee convergence. In this case we say that the convergence rate is *linear*.
- ii. If  $r > 1$  we say that the convergence rate is *superlinear*.
- iii. If  $r = 2$  we say that the convergence rate is *quadratic*.
- iv. These terms state how fast we converge once convergence is occurring. However, there is no guarantee of converging to the root we want, or to any root in general.
  1. Moreover, recall that our nonlinear function might be an approximation to what we really want, so how many digits of accuracy we need is important in deciding how fast things need to converge.

3. **fixed point iteration** - iterate  $x_{k+1} = g(x_k)$  to find  $x = g(x)$

a. *locally convergent* if  $|g'(x^*)| < 1$ , i.e. if the initial guess  $x_0$  is *close enough* to  $x^*$  the method will converge

- i.  $e_{k+1} = x_{k+1} - x^* = g(x_k) - g(x^*) = g'(\theta_k)(x_k - x^*) = g'(\theta_k)e_k$  for  $\theta_k$  between  $x_k$  and  $x^*$  as determined by the Mean Value Theorem. *Show that if  $g'(x^*) = 0$  and  $g''(x)$  is bounded, then convergence is in fact quadratic*
- ii. if all the  $|g'(\theta_k)| \leq C < 1$ , then  $|e_k| \leq C|e_{k-1}| \leq C^2|e_{k-2}| \leq \dots \leq C^k|e_0|$  so that as  $k \rightarrow \infty$ ,  $C^k \rightarrow 0$  and  $|e_k| \rightarrow 0$

b. only converges if the initial guess is *close enough* to the solution

c. functions may be *very expensive* to evaluate - try to evaluate them as few times as possible and save the old values for future use

4. **Newton's method**  $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$

a. careful when  $f'(x^*) = 0$ , i.e. poorly conditioned

b. stop when  $|f(x_k)| < \varepsilon$  which is equivalent to  $|x_{k+1} - x_k| < \frac{\varepsilon}{|f'(x_k)|}$

c. uses the tangent line to estimate the zero - *draw a picture of this*

- d. quadratic convergence in general, but only linear convergence for multiple roots.
- e. both  $f(x_k)$  and  $f'(x_k)$  need to be evaluated each iteration and  $f'(x_k)$  may be expensive.

Think of  $f(x)$  possibly being a computer program that is difficult to get derivatives from.

5. **secant method**  $x_{k+1} = x_k - f(x_k) \left( \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \right)$

a. Newton method with the slope estimated by a secant line  $f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$

- b. don't need to evaluate  $f'(x_k)$ 
  - i. moreover only one  $f(x_k)$  evaluation for each iteration
  - ii. other secant methods that do not use two previous iterates to compute the secant line are twice as costly for function evaluations
  - iii. superlinear convergence with  $r = 1.618$ 
    - 1. almost always beats Newton since only a couple more iterations are needed at half the cost

6. **bisection method** – guaranteed to converge to a root in the interval, as opposed to the fixed point iteration methods, e.g. Newton and secant

- a. find an interval  $[a, b]$  with  $f(a)f(b) < 0$  that contains the solution
  - i. note that  $f(a)f(b) = 0$  implies  $f(a) = 0$  or  $f(b) = 0$  and we're done
- b. find the midpoint  $m = (a + b)/2$ , and then if  $f(a)f(m) < 0$  set  $b = m$ , otherwise set  $a = m$  since  $f(b)f(m) < 0$ ,
  - i. note that  $f(m) = 0$  implies we are done
- c. given an error tolerance  $\epsilon$ , continue until  $b - a < \epsilon$
- d. the interval size decreases by  $1/2$  each iteration, so it is linearly convergent with  $C = 1/2$
- e. only  $f(m)$  needs to be evaluated each iteration

f. **mixed methods** – e.g. bisection for safety and secant for speed

- i. start with a bracketing interval  $[a, b]$  and set  $x_0 = a$  and  $x_1 = b$
- ii. apply the secant method to find  $x_{k+1}$  from  $x_k$  and  $x_{k-1}$
- iii. if  $x_{k+1} \in [a, b]$  set  $m = x_{k+1}$  and update the bracketing interval  $[a, b]$  using the bisection method, that is if  $f(a)f(m) < 0$  set  $b = m$ , otherwise  $f(b)f(m) < 0$  where you set  $a = m$
- iv. if  $x_{k+1} \notin [a, b]$  or  $f(b) - f(a)$  is too small to apply the secant method, apply the bisection method with the usual  $m = (a + b)/2$  and then set  $x_k = a$  and  $x_{k+1} = b$
- v. note that  $x_{k-1} = a$  and  $x_k = b$  implies that  $x_{k+1}$  comes from linear interpolation
- vi. needs only one  $f(x_k)$  function evaluation per iteration

7. Let's turn our attention back to systems of nonlinear equations, i.e.  $A(x) = b$  or  $F(x) = 0$ .

- a. Here the **Jacobian** matrix,  $J(x)$ , is rather useful as a linearization of the nonlinear problem.

- b. Here we define  $J_{ij} = \partial F_i / \partial x_j$  where each equation of  $F(x)=0$  is written individually as  $F_i(x)=0$ , and each  $x_j$  is the  $j$ -th component of the  $x$  vector.
- c. For example, consider  $F_1(x) = x_1 + \sin x_2 + 4 = 0$  and  $F_2(x) = (x_1)^2 + x_2 = 0$  which can be written in matrix form as  $F(x) = \begin{pmatrix} x_1 + \sin x_2 + 4 \\ (x_1)^2 + x_2 \end{pmatrix} = 0$ . Then the Jacobian matrix is
- $$J(x) = \begin{pmatrix} 1 & \cos x_2 \\ 2x_1 & 1 \end{pmatrix}.$$
- d. Note that  $J(x)$ , that is  $J$ , generally depends on the  $x$  vector.
- e. In general, we write  $J(x)=F'(x)$  and note that the Jacobian is the multidimensional generalization of  $f'(x)$ .
- f. Thus conditioning in multiple dimensions depends on the Jacobian matrix just as conditioning for scalars depends on  $f'(x)$ .
- g. Moreover, we desire a nonsingular  $J$ , at least “locally” to the solution, in order to proceed.
8. Newton’s method for  $F(x)=0$  is  $x_{k+1} = x_k - J^{-1}(x_k)f(x_k)$ .
- a. Note that  $1/f'(x_k)$  is replaced by  $J^{-1}(x_k)$ .
- b. Instead of computing the inverse, we solve the linear system  $J(x_k)y_k = -f(x_k)$  where  $y_k = x_{k+1} - x_k$ .
- That is, we first solve the linear system to find the increment  $y_k$  and then calculate  $x_{k+1} = x_k + y_k$
  - Now one can see why  $J$  needs to be nonsingular for  $x_k$  “close” to the solution  $x^*$ , i.e. because we need to solve the linear system
- c. This can be very computationally expensive since one has to repeatedly solve a linear system.
- d. Also, one might worry about robustness in case *any* one of the linear system solvers fails
- e. *More times than not, one has to repeatedly solve linearly systems, i.e. it’s not a one time deal. That’s precisely why it is important to look for symmetry, sparsity, and structure in problem formulations.*
- f. There are secant-like methods in multiple dimensions that compute the Jacobian matrix in pieces “on the fly”.
- Broyden’s Method
    - start with  $J_0 = I$
    - solve  $J_k y_k = -f(x_k)$  and set  $x_{k+1} = x_k + y_k$  as usual
    - $J_{k+1} = J_k + \frac{(F(x_{k+1}) - F(x_k) - J_k y_k) y_k^T}{y_k \cdot y_k}$