

*Computational Methods*  
CMSC/AMSC/MAPL 460

Solving nonlinear equations and zero finding

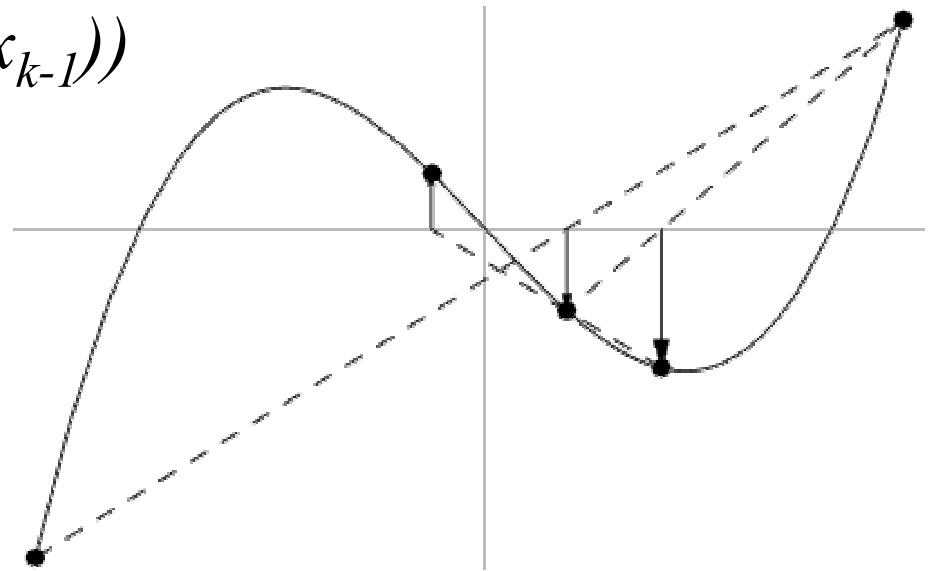
Ramani Duraiswami,  
Dept. of Computer Science

# Secant method

- Instead in the secant method choose two points
- Fit straight line and evaluate its zero

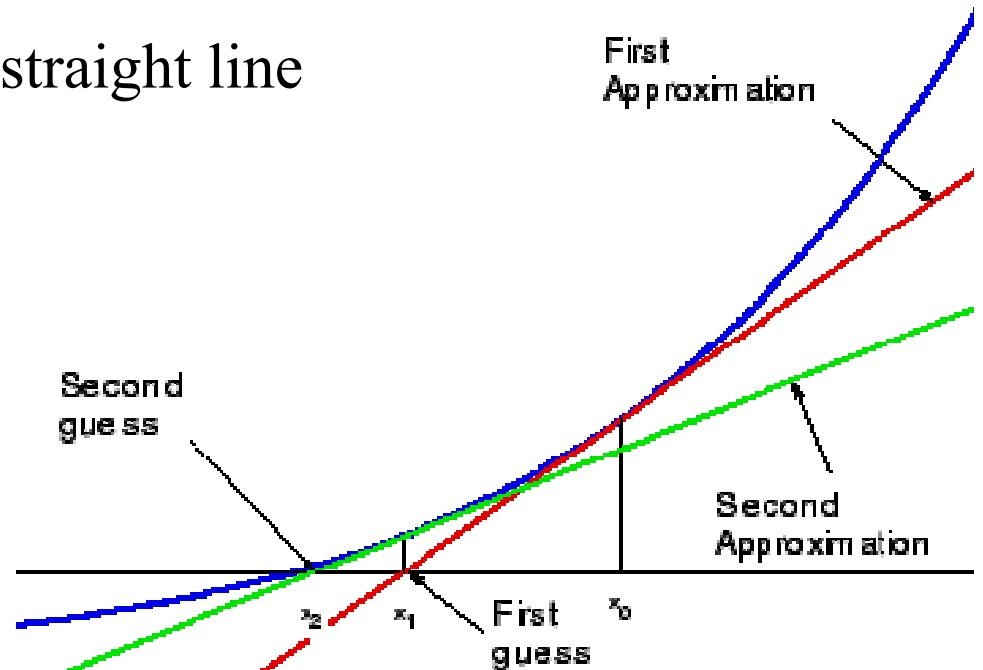
$$x_{k+1} = x_k - f(x_k)(x_k - x_{k-1}) / (f(x_k) - f(x_{k-1}))$$

- Choose next point and repeat
- Convergence is superlinear
- $e_{k+1} = c \cdot e_k^\phi$
- $\phi = (5^{1/2} + 1) / 2 = 1.62 \dots$



# Newton's method

- Several ways to derive: We choose Taylor series
- I want  $f(x_*)=0$
- But I have  $f(x_k)$  which is not zero
- Let me guess that  $f(x_k+h)$  will be zero
- $f(x_k+h)=f(x_k)+hf'(x_k)+O(h^2)$
- Ignore terms of  $O(h^2)$ 
  - Approximate curve locally as straight line
  - When will this not work?
- Solve  $f(x_k)+hf'(x_k)=0$  for  $h$
- So  $h=-f(x_k)/f'(x_k)$
- So  $x_{k+1}=x_k+h=x_k-f(x_k)/f'(x_k)$
- Repeat until convergence



# Convergence analysis

- For iterative algorithms, we want to know how the error decreases after each iteration
- We also want to check how much each iteration costs
- Optimal algorithm is the one that achieves a given error for a given cost
- Algorithm    convergence    function evaluations per step
- Bisection    Linear    One
- Secant    Superlinear    One
- Newton    Quadratic    Two
- Which method is better?
- Define better:
- Bisection guaranteed to converge, but slow
- Secant one evaluation per step Newton: two
- Newton quadratic convergence Secant super linear
- Newton needs derivative, which may be unavailable

## Comparing convergence

- Suppose cost of function evaluations for derivative and function are similar
- Then let Newton method converge in  $n$  steps to error  $\tau$
- So  $e_0^{2n} \leq \tau$
- Secant will require  $s$  steps to ensure  $e_0^{1.62s} \leq \tau$
- So cost of Newton is  $2n$  while that of secant is  $s$
- Which is larger?
- Take logs:  $2n \log e_0 \leq \log \tau$
- So  $2n \geq |\log \tau| / |\log e_0|$        $n \geq (2)^{-1} (|\log \tau| / |\log e_0|)$
- For secant:  $s \geq (1.62)^{-1} (|\log \tau| / |\log e_0|)$
- So  $\text{Cost}_{\text{Newton}} / \text{Cost}_{\text{Secant}} = (2/1.62) = 1.44$
- So Secant is cheaper!

# Infinite cycles

- Newton's method could iterate forever!

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

- cycles back and forth around a point  $a$  if

$$x_{n+1} - a = -(x_n - a)$$

- This happens if

$$x - a - \frac{f(x)}{f'(x)} = -(x - a)$$

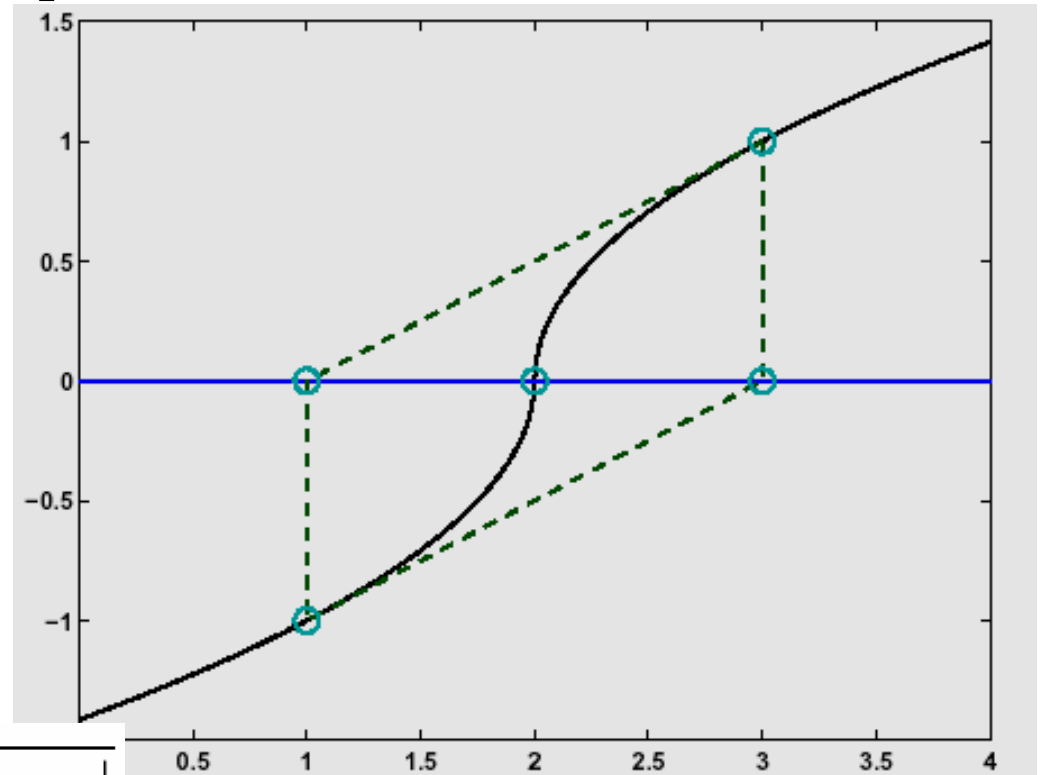
- Rewrite as an ODE for  $f$

$$\frac{f'(x)}{f(x)} = \frac{1}{2(x - a)}$$

- Solution

$$f(x) = \text{sign}(x - a) \sqrt{|x - a|}$$

- Such cycles could exist with secant methods as well.



# Inverse Quadratic Interpolation

- Secant method fits a straight line to predict zero from two previous values
- We could instead fit a parabola to predict the zero from three values!
- However parabola may not intersect x axis (straight line will always)
  - In this case roots will be complex
- Idea of inverse quadratic interpolation
  - Fit a parabola  $x=f(y^2)$  instead of a parabola  $y=f(x^2)$
  - Evaluate it at 0
- Problem: polynomial interpolation needs the points (here function values) to be distinct
- Cannot guarantee this!
- So method may not converge
- However near solution it converges very rapidly

```
k = 0;
fa=f(a)
fb=f(b)
fc=f(c)
while abs(c-b) > eps*abs(c)
    x = polyinterp([fa,fb,fc],[a,b,c],0)
    a = b;fa=fb;
    b = c;fb=fc;
    c = x;fc=f(x);
    k = k + 1;
end
```

## Guaranteed methods: Zeroin

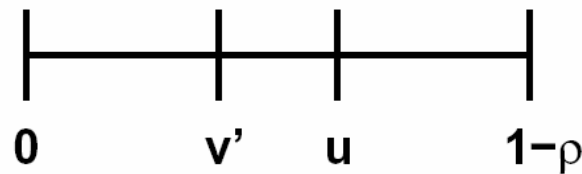
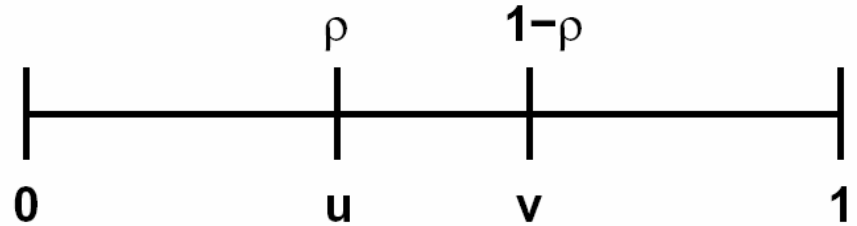
- Start with  $a$  and  $b$  so that  $f(a)$  and  $f(b)$  have opposite signs.
- Use a secant step to give  $c$  between  $a$  and  $b$ .
- Repeat until  $|b - a| < \varepsilon |b|$  or  $f(b) = 0$ .
  - Arrange  $a$ ,  $b$ , and  $c$  so that
    - $f(a)$  and  $f(b)$  have opposite signs.
    - $|f(b)| < |f(a)|$
    - $c$  is the previous value of  $b$ .
- If  $c \neq a$ , consider an IQI step.
- If  $c = a$ , consider a secant step.
- If the IQI or secant step is in the interval  $[a,b]$ , take it.
- If the step is not in the interval, use bisection.

# Optimization analog of bisection

- Optimization involves finding maximum and minimum of functions
- At these point first derivative vanishes
- So optimization typically involves use of differential methods
- Here we consider an algorithm like bisection
- Suppose we are given an interval  $[a,b]$  and have to find the minimum in this interval
- We could look at  $f(a)$ ,  $f(b)$  and  $f((a+b)/2)$
- Even if  $f((a+b)/2) < f(a)$  and  $f((a+b)/2) < f(b)$  don't know if  $[a,(a+b)/2]$  or  $[(a+b)/2,b]$  contains the minimum
- Could divide domain into three regions
- $f(a)$ ,  $f(b)$ ,  $f((a+b)/3)$ , and  $f(2(a+b)/3)$ .
- Then we know which interval  $[a,2(a+b)/3]$  or  $[(a+b)/3, b]$  contains the minimum

# Golden Search

- Let  $[0, 2/3]$  be the reduced domain
- At next step we cannot reuse our function evaluation at  $1/3$  (which is the mid-point of our interval)
- Instead we must evaluate the function at  $2/9$  and  $4/9$ .
- Thus each iteration requires two function evaluations.
- Can we instead choose points (not at  $1/3$  and  $2/3$ ) but at some other points  $\rho$  and  $1-\rho$ , so that the point can be reused in the next step
- So  $\rho/(1-\rho) = (1-\rho)/\rho$
- $1-2\rho + \rho^2 = \rho$
- $\rho^2 - 3\rho + 1 = 0$
- $\rho = (3 \pm (9-4)^{1/2})/2$
- $\rho = 1 + (1-5^{1/2})/2 = 2-\phi = 0.382..$
- Length of interval is reduced by a factor of  $\phi - 1 \simeq 0.618$  each step
  - So to converge to machine epsilon we require  $52/0.618 \simeq 75$  steps



# Improved Golden Search: `fminbd`

- As the search proceeds, we will have three points in the interval with the minimum
- Fit a parabola and find the minimum
- If the minimum is within the interval, we can choose it as the next point
- To stop: recall near a minimum derivative vanishes
- So  $f(x) = f(x_*) + b(x - x_*)^2$
- Let  $x - x_* = \delta$  and  $f(x_*) = a$   $f(x) = a + b\delta^2$
- If the interval  $\delta$  is as small as machine  $\varepsilon$ , then the change in the value of  $f$  will be of the order of machine  $\varepsilon^2$ .
- SO it is not computable
- Rather change can at most be about the square root of machine  $\varepsilon$
- This is employed in Matlab function `fminbd` and in the book software function `fminbx`

# Systems of Nonlinear equations

- Analog for 1d bisection: Too hard to find bracketed zero
- Analog for Newton is what is used
- Derivative is now  $\nabla f$

**Example:** Let the function  $f(x)$  be defined by

$$\begin{aligned}f_1(x_1, x_2) &= x_1^3 + \cos(x_2) \\f_2(x_1, x_2) &= x_1x_2^2 - x_2^3.\end{aligned}$$

Then in solving  $f(x) = 0$ , we look for a point  $x_1, x_2$  where both  $f_1$  and  $f_2$  are zero.

The derivative of the function is called the **Jacobian matrix**. It is a matrix  $J(x)$  defined by

$$J(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 3x_1^2 & -\sin(x_2) \\ x_2^2 & 2x_1x_2 - 3x_2^2 \end{bmatrix}$$

# Multidimensional Newton's Method

- $f_1(\mathbf{x}) = f_1(x_1, x_2, \dots, x_N)$
- $f_2(\mathbf{x}) = f_2(x_1, x_2, \dots, x_N)$
- ...
- $f_M(\mathbf{x}) = f_M(x_1, x_2, \dots, x_N)$
  
- $f_1(\mathbf{x}+\mathbf{h}) = f_1(x_1, x_2, \dots, x_N) + \nabla f_1 \cdot \mathbf{h}$
- ...
- $f_M(\mathbf{x}+\mathbf{h}) = f_M(x_1, x_2, \dots, x_N) + \nabla f_M \cdot \mathbf{h}$
  
- $\nabla f_1 = [\partial f_1 / \partial x_1 \quad \partial f_1 / \partial x_2 \quad \dots \quad \partial f_1 / \partial x_N]$
- ...
- $\nabla f_M = [\partial f_M / \partial x_1 \quad \partial f_M / \partial x_2 \quad \dots \quad \partial f_M / \partial x_N]$
  
- **$\mathbf{f}(\mathbf{x}+\mathbf{h}) = \mathbf{f}(\mathbf{x}) + \mathbf{Jh} = \mathbf{0}$**
- Solve  **$\mathbf{Jh} = -\mathbf{f}$**  to find the step
- **$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{h}$**