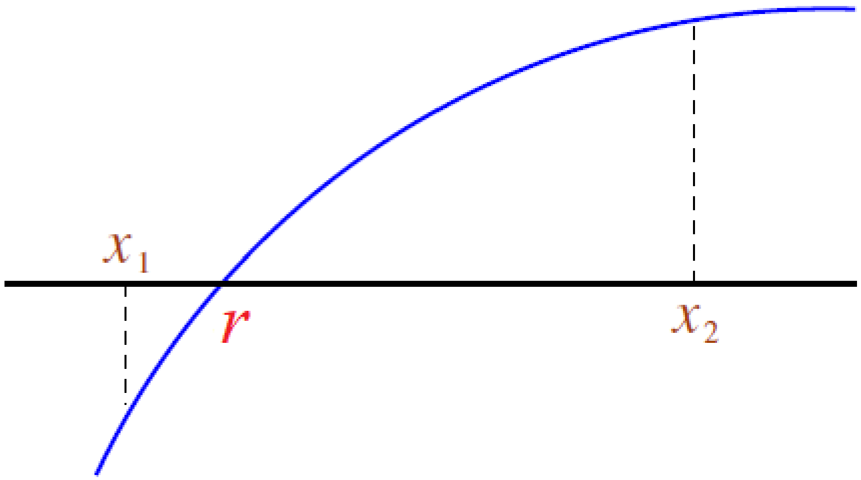
Solving Non-linear Equations

IPCST
Seoul National University

Finding Roots

- Premise
  - A root: r
  - $f(x)$: continuous
    - Real-valued
  - Existence of r
    - $f(r) = 0$



    - $x_1 < x_0 < x_2 \rightarrow f(x_1) \cdot f(x_2) < 0$
    - ➢ For the bisection method

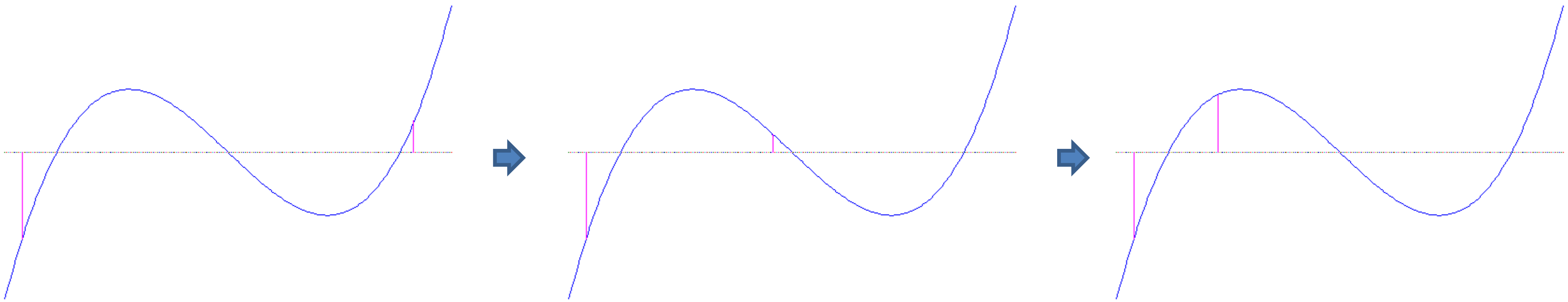
Bisection Method

- Version 1
  1. Take initial x_1 and x_2 as $f(x_1) \cdot f(x_2) < 0$
  2. Given x_1 and x_2, take $x_3 = \dfrac{x_1 + x_2}{2}$
     - Let $x_1 = x_3$ if $\text{sign}(f(x_1)) = \text{sign}(f(x_3))$
     - Let $x_2 = x_3$ if $\text{sign}(f(x_2)) = \text{sign}(f(x_3))$
     - Otherwise: $f(x_3) = 0 \rightarrow$ Stop.
  3. Repeat until $|x_1 - x_2| <$ tolerance
     - ✓ Better set a limit of the iteration number in case it takes too much time

Bisection Method

- Version 2
  1. Take initial x_1 and x_2 as $f(x_1)\cdot f(x_2) < 0$
  2. Given x_1 and x_2, take $x_3 = \dfrac{x_1 + x_2}{2}$
     - Let $x_1 = x_3$ if sign($f(x_1)$) = sign($f(x_3)$)
     - Else, $x_2 = x_3$
  3. Repeat until $f(x_3)$ < tolerance
     - ✓ Better set a limit of the iteration number in case it takes too much time

Bisection Method



- Slow – linear convergence
- Robust
  - Never fails under the premise
  - But it cannot catch some multiple roots.

Bisection Method

- Convergence
  - The error should be equal to or less than the half of the interval.

$$e_k \le \frac{u_k - l_k}{2} = \frac{u_0 - l_0}{2^{k+1}}$$

    - where u_k: kth upper bound, l_k: kth lower bound

  - **Linear convergence**

$$e_{k+1} = O(e_k)$$

Fixed Point Iteration

- Main idea
  - Modifying $f(x) = 0 \rightarrow x = g(x)$
  - The fixed point of $g(x)$ gives the root r because
    $$f(r) = 0 \leftrightarrow r = g(r)$$
- Algorithm
  1. Guess an initial value x_0
  2. Iterate $x_{k+1} = g(x_k)$
  - Stopping criteria
    - $|x_k - x_{k+1}| < (\text{tol.})$ or $|x_k - g(x_k)| < (\text{tol.})$
    - Limit of the iteration number set by you

Fixed Point Iteration

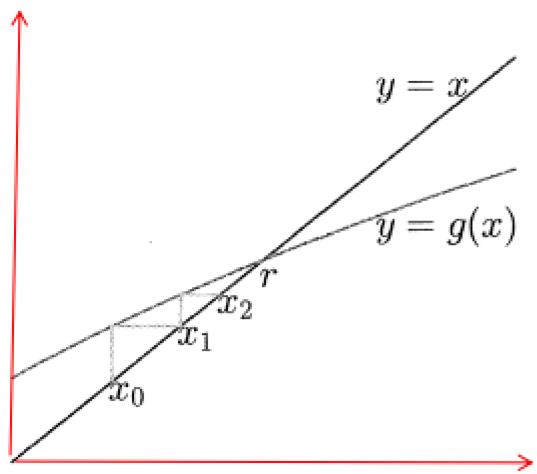
- Convergence
  - Root: r s.t. $r = g(r)$
  - Error: $e_{k+1} = |x_{k+1} - r| = |g(x_k) - r|$
    $= |g(x_k) - g(r)| = |g'(\xi_k)||x_k - r| = |g'(\xi_k)|e_k$
    - where $\xi_k \in (x_k, r)$
  - Linear convergence if $|g'(\xi_k)| < 1$

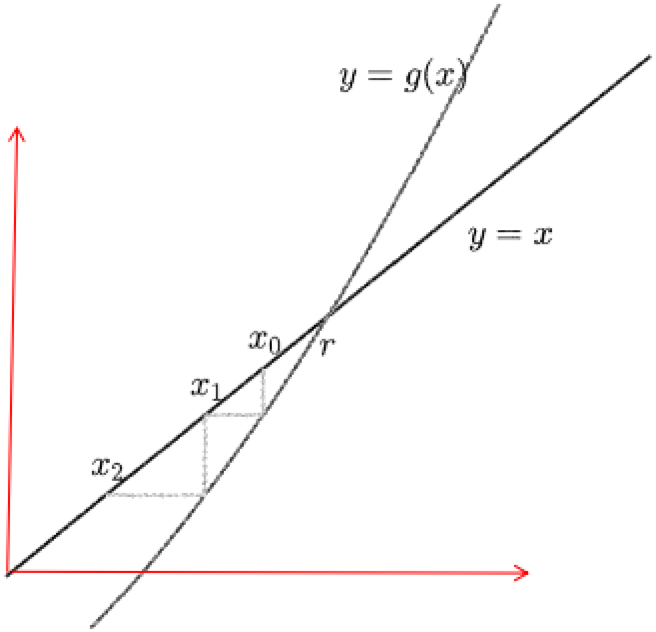
$$e_{k+1} = O(e_k)$$

  - Divergence if $|g'(\xi_k)| > 1$

Fixed Point Iteration

- Convergence condition (Wen Shen p. 102)
  - **There is $\boldsymbol{a > 0}$ s.t. $\boldsymbol{|g'(x)| < 1}$ for every $\boldsymbol{x \in [r-a, r+a]}$,** and the initial guess $x_0 \in [r-a, r+a]$





Figures from Wen Shen

Fixed Point Iteration

- Error estimate
  - Wen Shen pp. 103~104
  - Assume $|g'(x)| \leq M < 1$

$$e_{k+1} \leq Me_k \leq M^2 e_{k-1} \leq \cdots \leq M^{k+1} e_0$$

$$e_0 = |r - x_0| = |r - x_1 + x_1 - x_0| \leq e_1 + |x_1 - x_0|$$
$$\leq Me_0 + |x_1 - x_0|$$

$$\rightarrow e_0 \leq \frac{1}{1-M} |x_1 - x_0|$$

$$\Rightarrow \quad e_k \leq \frac{M^k}{1 - M} |x_1 - x_0|$$

Do It Yourself

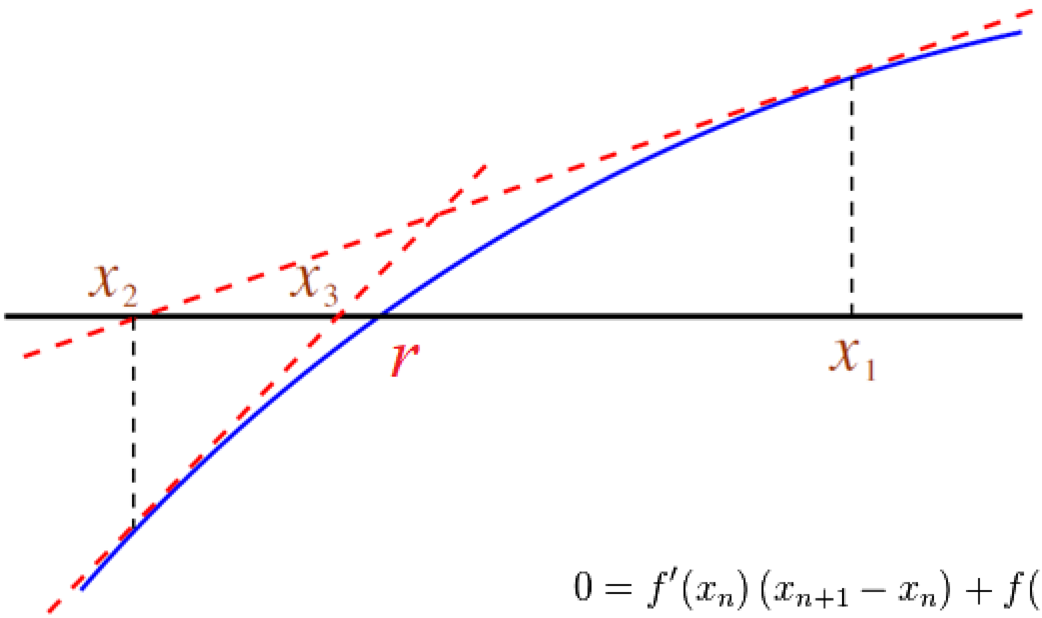
- Find a root of $\cos x = x$ with the fixed point iteration. Get 4 significant digits.
  – Wen Shen Example 5.4 & 5.6

- [After this class]: Try the fixed point iteration for the root of $e^{-x} = \cos x$. Check 4 iteration values for different form of $g(x)$ and initial guesses.
  – Wen Shen 5.7 homework problem 2

Newton's Method

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

- until (error) < (tolerance)



You can regard this
as a kind of
fixed point iteration

$$0 = f'(x_n)\,(x_{n+1} - x_n) + f(x_n).$$

Newton's Method

- Preconditions
  - The function $f(x)$ must be *smooth*.
  - The derivative $f'(x)$ should be *known* or exactly calculated.
  - The initial guess must be *close* to the final result.
- Algorithm
  1. Guess an initial value x_0
  2. Iterate $x_{k+1} = x_k - f(x_k)/f'(x_k)$
  - Stopping criteria
    - $|x_k - x_{k+1}| < (\text{tol.})$ or $|f(x_k)| < (\text{tol.})$
    - Limit of the iteration number set by you

Newton's Method

- Advantages
  - **Best** fixed point iteration
    - *optimal* fixed point iteration if $g'(r) = 0$ where $r = g(r)$
    - Newton method: $g(x) = x - f(x)/f'(x)$ → $g'(r) = 0$

  - *Quadratic* convergence near the root
    - Since x_k is close to r,

$$g(x_k) = g(r) + (x_k - r)g'(r) + \tfrac{1}{2}(x_k - r)^2 g''(\xi_k),\ \xi_k \in (x_k, r)$$

$$\rightarrow\ g(x_k) - g(r) = \tfrac{1}{2}(x_k - r)^2 g''(\xi_k)$$

    - Error: $e_{k+1} = |x_{k+1} - r| = |g(x_k) - g(r)| = \frac{1}{2}e_k^2|g''(\xi_k)|$

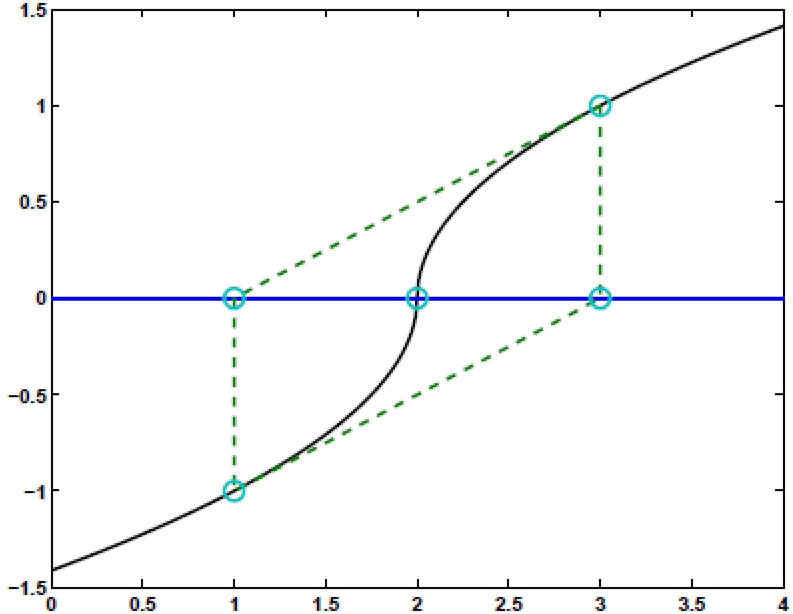
$$e_{k+1} = O(e_k^2)$$

Newton's Method

- Disadvantages of Newton's method
  - Fails at a stationary point ($f'(x) = 0$)
  - If the starting point is far from the root, it can lead to divergence or slow-convergence.
  - Linear convergence near a multiple root
  - Not well-behaved for a non-smooth function
  - There can be a cycle (infinite loop).

Newton's Method

- Disadvantages of Newton's method



An infinite loop

Figure from Moler

Do It Yourself

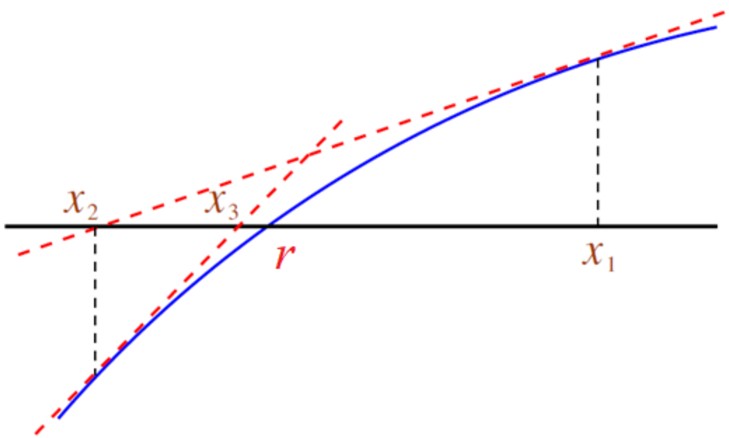
- Find a square-root of a natural number with the Newton's method. Get 5 significant digits.
  – Wen Shen Example 5.7

- [After this class]: Apply the Newton's method to $\mathrm{sgn}(x-2)\sqrt{x-2}=0$. (See a cycle.)
- [After this class]: Try to solve $(x-1)^m=0$
  – Wen Shen 5.7 homework problem 5

參考: Newton's Method

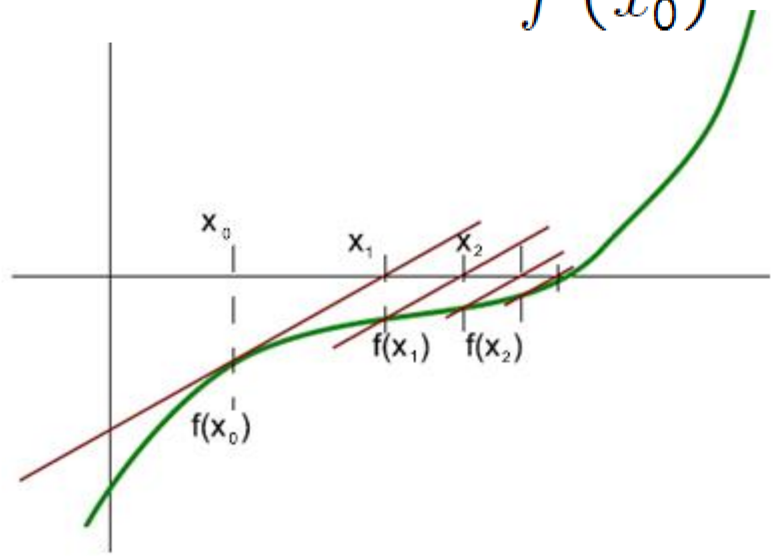
- Original

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



- Simplified

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_0)}$$



- Modified version: calculating $f'(x)$ at every 6~8 steps

參考: Newton's Method

- Modified version for vector-valued functions

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \boldsymbol{\delta}$$

$$\mathbf{J}(\boldsymbol{x}_N)\ \boldsymbol{\delta} = \mathbf{f}(\mathbf{x}_n) \quad \rightarrow \text{linear algebra}$$

➢ $\mathbf{J}(\boldsymbol{x}_N)$: Jacobian of $\mathbf{f}(\mathbf{x})$ at $\mathbf{x}_N$

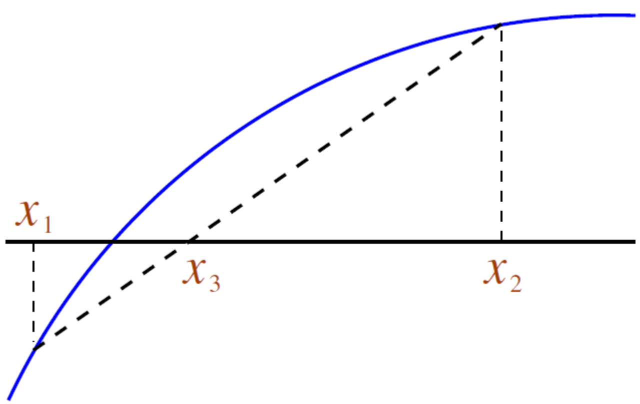
$N = C \cdot m \leq n < C \cdot (m+1)$.

m: integer, C: constant integer 6~8.

Secant Method

- Newton's method with a finite difference approximation based on the two most recent iterates.

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



◆ Advantages
- No need of f'
- One f(x) computing per step

參考: Secant Method

- *Super-linear* convergence
  - ❖Preconditions
    - $f'(x)$ and $f''(x)$ exist and are continuous.
    - The initial values are close to the root.

$$e_{n+1} = \frac{1}{2}\frac{f''(\xi)f'(\xi_n)f'(\xi_{n-1})}{f'(\xi)^3}\, e_n e_{n-1}$$

$$e_{n+1} = O(e_n e_{n-1})$$

- Not for a multiple root, $e_{n+1} = O(e_n^{\phi})$

ϕ : golden ratio

Finding Roots

- Calculation speed
  - Newton ≈ Secant > Bisection
- Stability
  - Bisection > Secant ≈ Newton
- A hybrid method works better
  1. Bisection at first (lower bound a & upper bound b)
  2. Newton or secant (new point x_{i+1})
  3. Accept it if $a < x_{i+1} < b$. Otherwise, calculate it again by bisection (new a, b & x_{i+1})
  4. Repeat 2~3 until it satisfies the convergence criterion

Finding Roots

- Hybrid method 2
  1. Take initial a and b as $f(a) \cdot f(b) < 0$
  2. Given a and b, take x_{i+1} from the secant method
     - Let $a = x_{i+1}$ if $\text{sign}(f(a)) = \text{sign}(f(x_{i+1}))$
     - Else, $b = x_{i+1}$
  3. Repeat until $|a - b| <$ tolerance or $f(x_{i+1}) <$ tolerance

- Hybrid method 3
  1. Bisection for 5-6 iteration to get good x_0
  2. Newton or secant to get r

參考: Optimization

- Root finders can be used to find minima or maxima.
  – Note: local minima or maxima
  1. Solving $f'(x) = 0$
  2. Golden section search
     - Unequal division: $\rho = 2 - \phi$ (ϕ: golden ratio)
     - Check $f(\mathrm{u}) < f(\mathrm{v})$

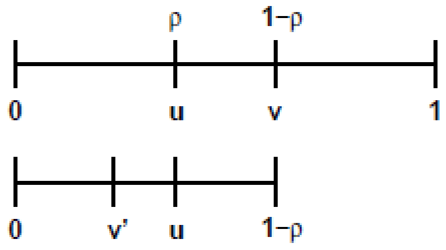


Figure from Moler

References

- Wen Shen,
  An Introduction to Numerical Computation
- C. Moler,
  Numerical Computing with MATLAB
- Wikipedia
- L. O. Jay, “Inexact Simplified Newton Iterations for Implicit Runge-Kutta Methods”, SIAM J. Numer. Anal. **38**, 1369 (2000).

Further Study

- There are many other root finding algorithms
  - Ex.) Muller's method,
    inverse quadratic interpolation,
    Brent-Dekker method
    ……