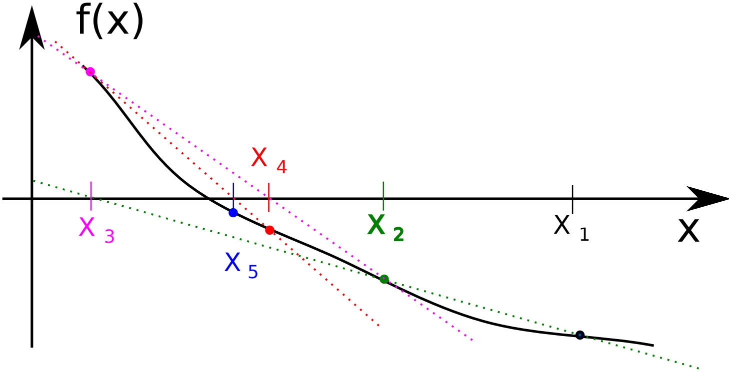
Root finding continued

Eugeniy E. Mikhailov

The College of William & Mary

Lecture 07

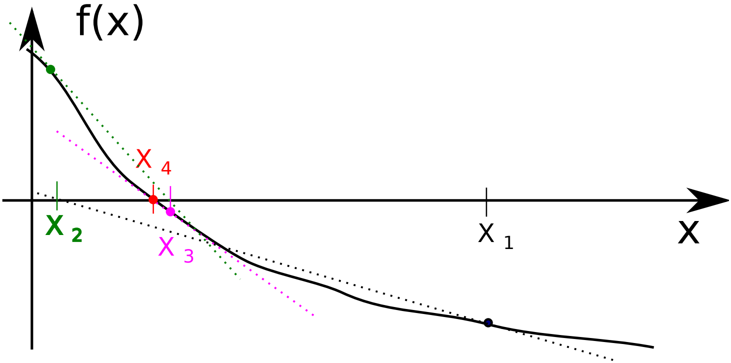
Secant method



$$x_{i+2} = x_{i+1} - f(x_{i+1})\frac{x_{i+1} - x_i}{f(x_{i+1}) - f(x_i)}$$

Need to provide two starting points x_1 and x_2.
Secant method converges with $m = (1 + \sqrt{5})/2 \approx 1.618$

Newton-Raphson method



$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Need to provide a starting points x_1 and the derivative of the function.
Newton-Raphson method converges quadratically ($m = 2$).

Numerical derivative of a function

Mathematical definition

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The initial intent is to calculate it at very small *h*.

Numerical derivative of a function

Mathematical definition

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The initial intent is to calculate it at very small h.
Remember about roundoff errors (HW01).

Numerical derivative of a function

Mathematical definition

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The initial intent is to calculate it at very small *h*.
Remember about roundoff errors (HW01).
For computers with *h* small enough $f(x+h) - f(x) = 0$.

Numerical derivative of a function

Mathematical definition

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The initial intent is to calculate it at very small h.
Remember about roundoff errors (HW01).
For computers with h small enough $f(x+h) - f(x) = 0$.
Let's be smarter. Recall Taylor series expansion

$$f(x+h) = f(x) + \frac{f'(x)}{1!}h + \frac{f''(x)}{2!}h^2 + \cdots$$

Numerical derivative of a function

Mathematical definition

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The initial intent is to calculate it at very small h.
Remember about roundoff errors (HW01).
For computers with h small enough $f(x+h) - f(x) = 0$.
Let's be smarter. Recall Taylor series expansion

$$f(x+h) = f(x) + \frac{f'(x)}{1!} h + \frac{f''(x)}{2!} h^2 + \cdots$$

So we can see

$$f'_c(x) = \frac{f(x+h) - f(x)}{h} = f'(x) + \frac{f''(x)}{2} h + \cdots$$

Here computed approximation and algorithm error

Mathematical definition

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The initial intent is to calculate it at very small h.
Remember about roundoff errors (HW01).
For computers with h small enough $f(x+h) - f(x) = 0$.
Let's be smarter. Recall Taylor series expansion

$$f(x+h) = f(x) + \frac{f'(x)}{1!}h + \frac{f''(x)}{2!}h^2 + \cdots$$

So we can see

$$f'_c(x) = \frac{f(x+h) - f(x)}{h} = f'(x) + \frac{f''(x)}{2}h + \cdots$$

Here computed approximation and algorithm error There is a range of optimal h when both the round off and the algorithm errors are small.

Derivative via Forward difference

$$f_c'(x) = \frac{f(x+h) - f(x)}{h}$$

Algorithm error

$$\varepsilon_{fd} \approx \frac{f''(x)}{2} h$$

Derivative via Forward difference

$$f_c'(x) = \frac{f(x+h) - f(x)}{h}$$

Algorithm error

$$\varepsilon_{fd} \approx \frac{f''(x)}{2} h$$

This is quite bad since error is proportional to h.

Derivative via Forward difference

$$f'_c(x) = \frac{f(x+h) - f(x)}{h}$$

Algorithm error

$$\varepsilon_{fd} \approx \frac{f''(x)}{2} h$$

This is quite bad since error is proportional to *h*.

Example

$$f(x) = a + bx^2$$

Derivative via Forward difference

$$f'_c(x) = \frac{f(x+h) - f(x)}{h}$$

Algorithm error

$$\varepsilon_{fd} \approx \frac{f''(x)}{2} h$$

This is quite bad since error is proportional to h.

Example

$$f(x) = a + bx^2$$

$$f'_c(x) = bxh + bh$$

Derivative via Forward difference

$$f_c'(x) = \frac{f(x+h) - f(x)}{h}$$

Algorithm error

$$\varepsilon_{fd} \approx \frac{f''(x)}{2} h$$

This is quite bad since error is proportional to h.

Example

$$f(x) = a + bx^2$$

$$f_c'(x) = bxh + bh$$

So for small x, the algorithm error dominate our approximation!

Derivative via Central difference

$$f_c'(x) = \frac{f(x+h) - f(x-h)}{2h}$$

Derivative via Central difference

$$f'_c(x) = \frac{f(x+h) - f(x-h)}{2h}$$

Algorithm error

$$\varepsilon_{cd} \approx \frac{f'''(x)}{6} h^2$$

Bonus problem for the homework 03 (5 points)

Plot the $\log_{10}$ of the absolute error of $\sin(x)$ derivative at $x = \pi/4$ calculated with forward and central difference methods vs the $\log_{10}$ the h value. See `loglog` help for ploting with logarithmic axes. The values of h should cover the range $10^{-16}, 10^{-15}, 10^{-14} \cdots 10^{-1}, 1$.

At the low end error will be dominated by round offs and at the higher by the algorithm error.

The minimum of the absolute error indicates optimal values of h.

Ridders method - the variation of false position

Solve $f(x) = 0$ with linear approximation of the function $g(x) = f(x)\exp(hQ)$

1. bracket the root between x_1 and x_2
2. evaluate function in the mid point $x_3 = (x_1 + x_2)/2$
3. find new approximation for the root

$$x_4 = x_3 + sign(f_1 - f_2)\frac{f_3}{\sqrt{f_3^2 - f_1 f_2}}(x_3 - x_1)$$

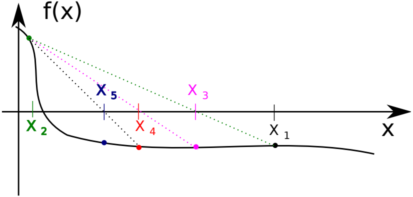
   where $f_1 = f(x_1), f_2 = f(x_2), f_3 = f(x_3)$
4. check if x_4 satisfies convergence condition
5. re bracket the root using
   - x_4 and $f_4 = f(x_4)$
   - whichever of (x_1, x_2, x_3) is closer to x_4 and provides proper bracket.
6. proceed to step 1

Nice parts: x_4 is guaranteed to be inside the bracket, convergence of the algorithm is quadratic $m = 2$. But it requires evaluation of the $f(x)$ twice for f_3 and f_4 thus actually $m = \sqrt{2}$.

Root finding algorithm gotchas

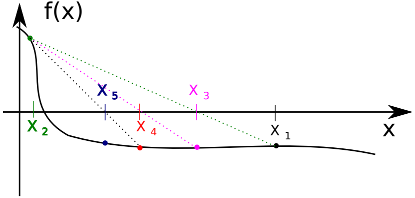
Root finding algorithm gotchas

Bracketing algorithm are bullet proof and will always converge, however false position algorithm could be slow.

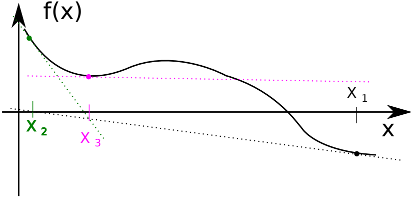


Root finding algorithm gotchas

Bracketing algorithm are bullet proof and will always converge, however false position algorithm could be slow.



Newton-Raphson and secant algorithm are usually fast but starting points need to be close enough to the root.



Root finding algorithms summary

Root bracketing algorithms

- bisection
- false position
- Ridders

Pro

- robust i.e. always converge.

Contra

- usually slower convergence
- require initial bracketing

Non bracketing algorithms

- Newton-Raphson
- secant

Pro

- faster
- no need to bracket (just give a reasonable starting point)

Contra

- may not converge