

Statistics and Numerical Methods, Tsinghua University

Non-linear Systems

Xuening Bai (白雪宁), Zhuo Chen (陈卓)

Institute for Advanced Study (IASTU) & Department of Astronomy (DoA)



清華大學

Tsinghua University

Oct. 8, 2024

Overview

Find the solution for $f(x) = 0$, or $f(\boldsymbol{x}) = 0$.

But this is much more difficult!

Being non-linear, there can be no/one/multiple/family of solutions.

Problem is easier if the root is known to be between some **bracketing** values.

Usually involves **iteration**, until some convergence is reached.

Success crucially depends on first guess.

The strategy should depend on the properties of f . The better you know about f , the more likely you find more effective ways to solve it.

Examples

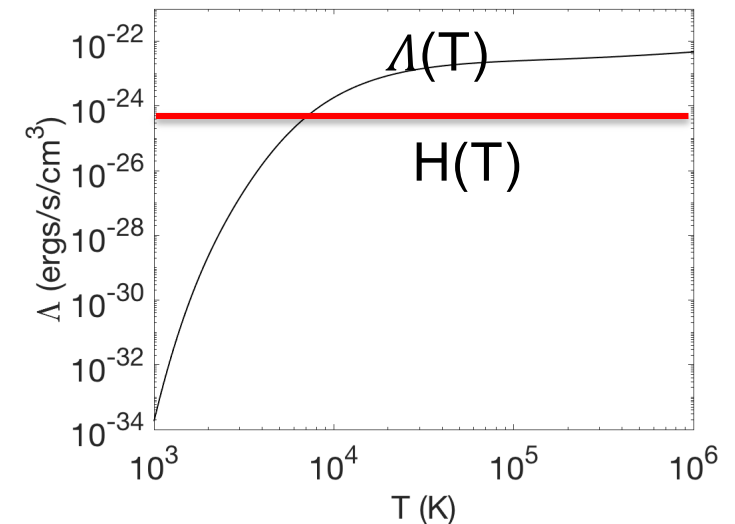
Finding the root of a polynomial with $n \geq 3$ (or 5).

Evaluation of some special functions.

Finding equilibrium solutions in various physical contexts.

The temperature T in interstellar medium depends on heating and cooling rates, and the latter can be a highly non-linear function of T itself.

$$H(T) = \Lambda(T)$$



Minimization/maximization can be considered as a related problem, which also includes many iterative methods for solving sparse systems.

Outline

- Root-finding for a single variable
 - Standard methods when only f is known
 - Newton's method
- Multivariable problems

General considerations

Goal: find the solution for $f(x) = 0$.

We are generally dealing with relatively complex functions (otherwise you'll solve it by hand).

Most of the computational effort lies in evaluating the function $f(x)$.

=> evaluate $f(x)$ as few times as possible.

Fast and efficient:

Rate of convergence: favors methods with higher-order convergence.

Stable and robust:

Usually, higher-order methods are less stable, more sensitive to initial guesses.

Many matured algorithms adopt a hybrid strategy.

Continuity and bisection

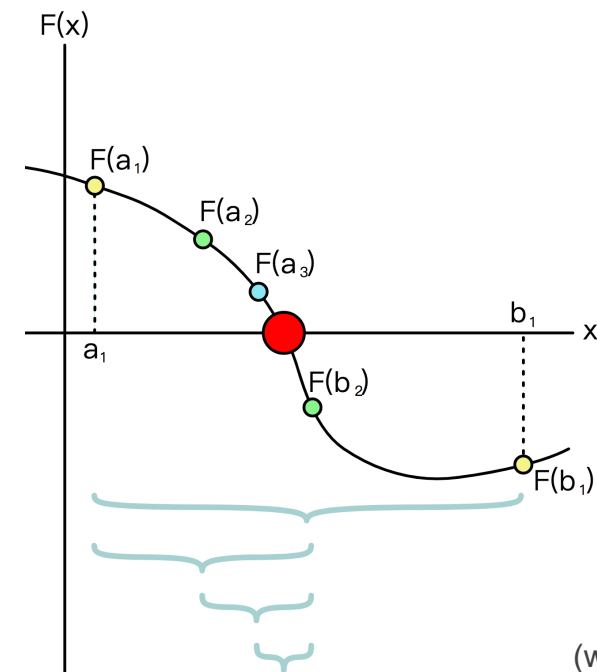
We say a root is **bracketed** in the interval (a, b) if $f(a)$ and $f(b)$ have opposite signs. If f is continuous, then at least one root must lie in this interval.

The **bisection method** is the most robust (never fail), but also somewhat slow.

- Compute the midpoint $c=(a+b)/2$
- Evaluate $f(c)$
- If $f(c) = 0$, then the root is found, and we are done
- Otherwise, choose the new domain:

$[a, c]$ if $f(a)f(c) < 0$

$[c, b]$ if $f(c)f(b) < 0$



Convergence of bisection

The bisection method converges linearly:

$$\epsilon_{n+1} = \frac{1}{2}\epsilon_n \quad \text{This means convergence is actually exponential!}$$

More generally, a method converges to m^{th} power if $\epsilon_{n+1} = \text{const} \times (\epsilon_n)^m$

If the domain contains multiple roots, bisection is guaranteed to converge to one root, depending on the initial guess.

Practical criterion for error tolerance:

Round-off error can suffer from significant amplification in the evaluation of f .

Thus, either set a threshold \gg roundoff, or simply stop after some finite number of iterations (e.g., 40).

Secant method

Can we do better than linear convergence?

Smooth functions are always locally linear: approximate with a straight line.

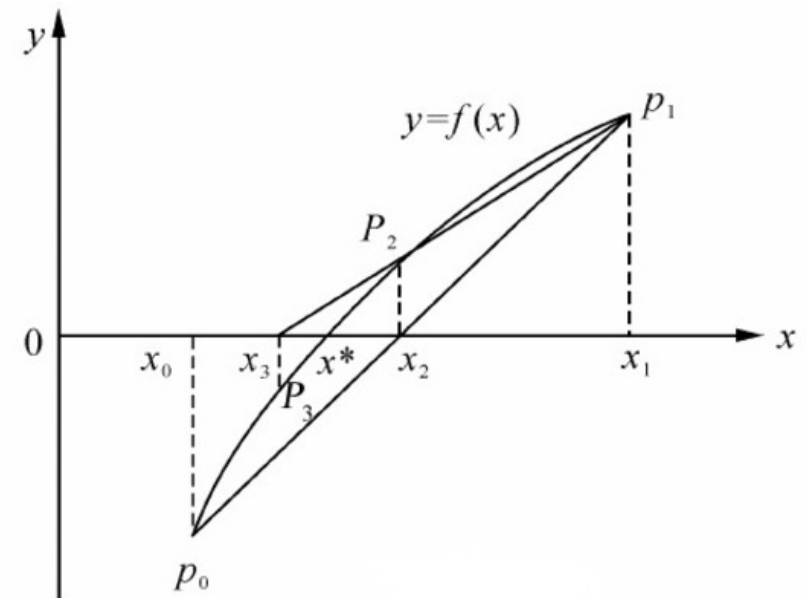
That is, draw a straight line between a and b, and the root will lie near the intersection of this line with the x-axis.

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

Essentially, use secant to approximate derivative (see Newton's method later)

For sufficiently continuous function, it converges near the root as:

$$\lim_{n \rightarrow \infty} \epsilon_{n+1} = \text{const} \times (\epsilon_n)^{1.618}$$

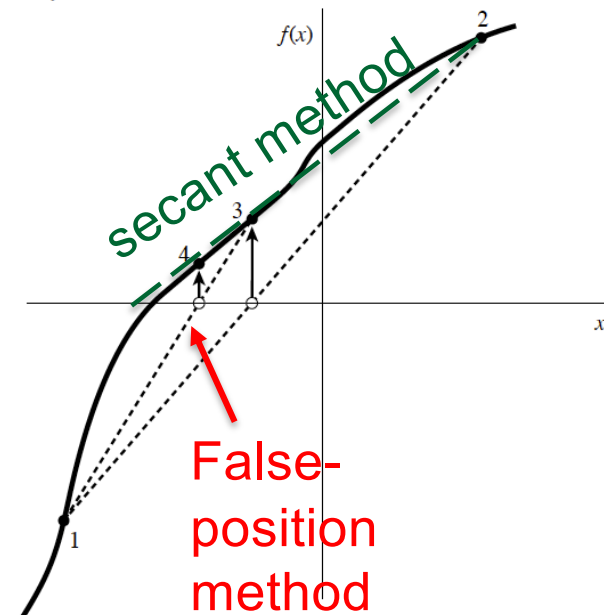


False-position method

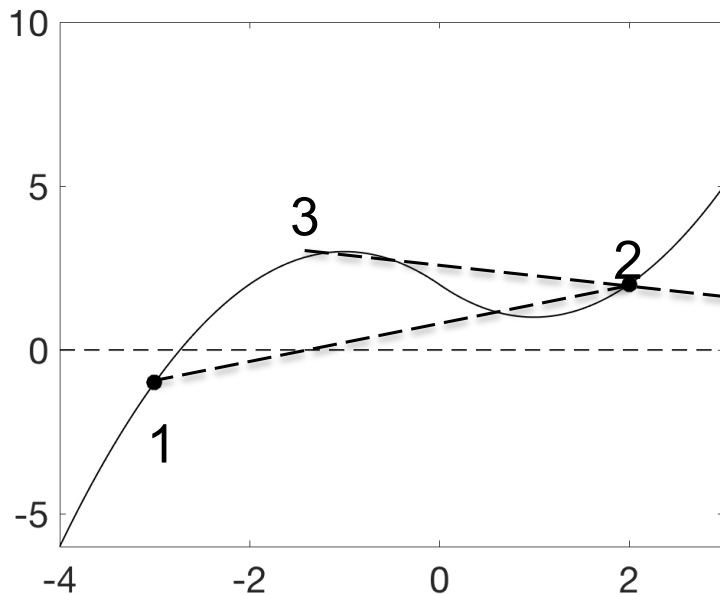
With the Secant method, there is no guarantee that the root remains bracketed during iteration, and it can diverge.

This can be improved by the “false-position method”, which chooses the older point if necessary to ensure that the root is always bracketed.

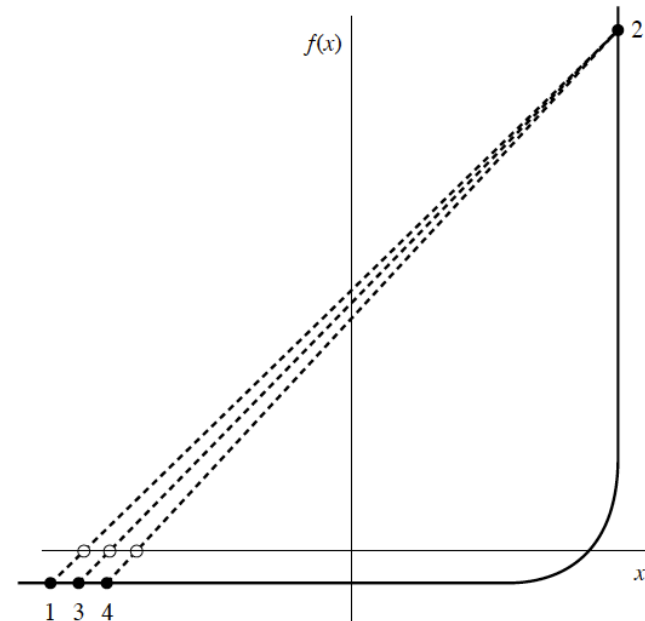
This method is still super-linear, and is more robust, but convergence is slower than the secant method.



They can also get stuck, or fail



Secant method fails but false position method saves the day



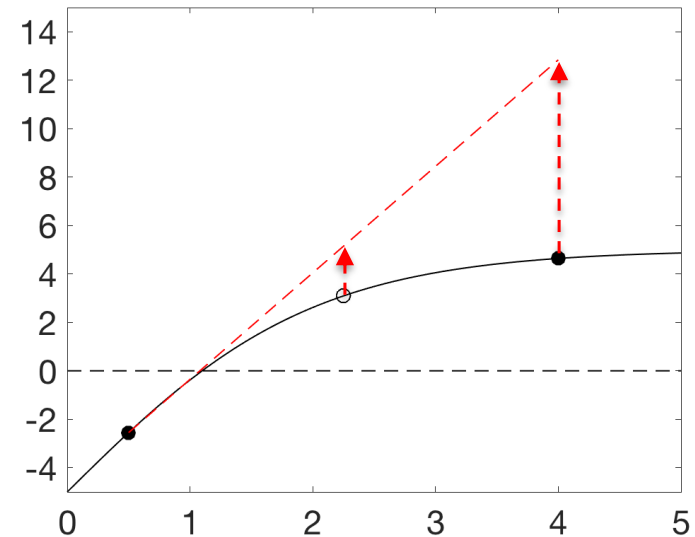
Both secant and false position methods get stuck (much worse than the bisection method)

Ridders' method

A root-finding algorithm based on false-position method.

Use an exponential function to successively approximate the root.

- Choose midpoint $x_m = (x_1 + x_2)/2$.
- Consider a function $h(x) = f(x)e^{ax}$.
- Calculate a so that
$$h(x_m) = [h(x_1) + h(x_2)]/2$$
- Apply false-position method on $h(x)$ to find the next x .



It reaches quadratic convergence for well-behaved functions (but two evaluations per iteration, so actual order of convergence is $\sqrt{2}$).

If not well-behaved, the root remains bracketed (bisection), so convergence is at least as good as bisection.

Towards higher-order interpolation

There are several ways to generalize the secant method, which uses linear interpolation to find the next point.

Generalization to quadratic interpolation gives the **Muller method** (use previous 3 values to construct a parabola).

Converges fast: order of convergence ~ 1.84 .

Can give complex numbers even the solution should be real.

There is also the **inverse quadratic interpolation** method (interpolate f^{-1} , not f):

$$x_{n+1} = \frac{f_{n-1}f_n}{(f_{n-2} - f_{n-1})(f_{n-2} - f_n)}x_{n-2} + \frac{f_{n-2}f_n}{(f_{n-1} - f_{n-2})(f_{n-1} - f_n)}x_{n-1} + \frac{f_{n-2}f_{n-1}}{(f_n - f_{n-2})(f_n - f_{n-1})}x_n,$$

Same rate of convergence near the root (order=1.84).

Can become unstable if not starting near the root (e.g., any of f_{n-2}, f_{n-1}, f_n coincide).

Brent's method

This is a **hybrid method**, that combines root bracketing, bisection, secant, and inverse quadratic interpolation.

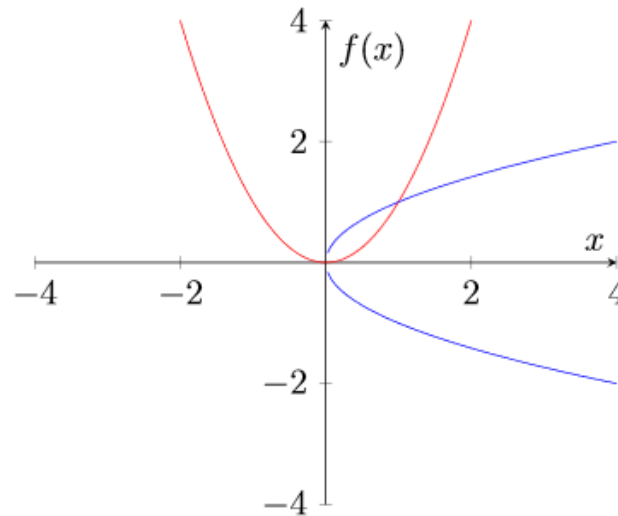
- Start from x_1, x_2 that bracket the root, with $x_3 = x_2$.
- Attempt to use inverse quadratic interpolation among x_1, x_2, x_3 (if two of them are equal, use secant method).
- If the resulting value is beyond the expected bracketing range, or the bracketing interval shrinks too slowly, switch to bisection.
- Repeat until convergence is reached.

See Numerical recipes for a detailed implementation.

This proves to be the most robust and efficient method for 1D root finding, when only a function's values are available.

Brent's method

This is a **hybrid method**, that combines root bracketing, bisection, secant, and inverse quadratic interpolation.



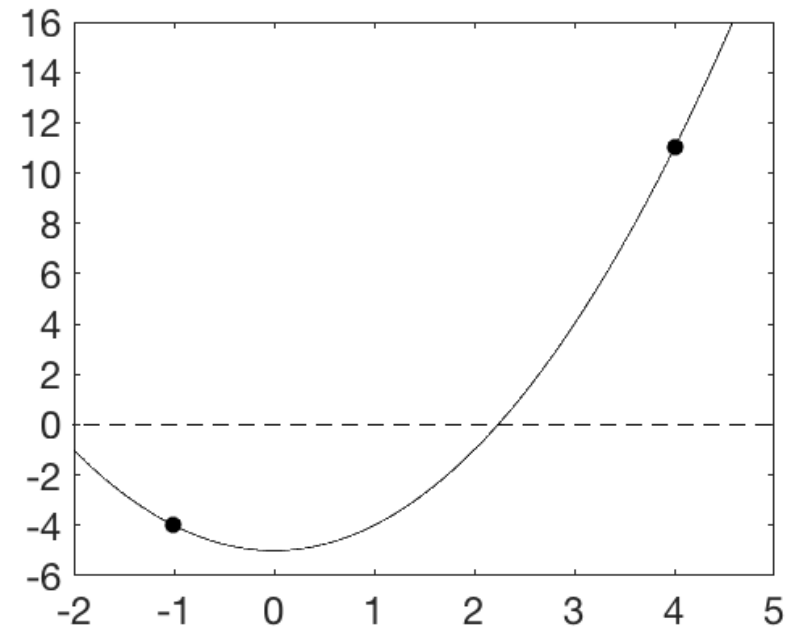
This proves to be the most robust and efficient method for 1D root finding, when only a function's values are available.

Comparing these methods

Solving $y = x^2 - 5$

Initial values: $x_1=-1$, $x_2=4$; tolerance: $1e-14$.

Method	Success?	# iterations
Bisection	Yes	49
Secant	Yes	10
False position	Yes	31
Inverse quadratic	Yes	8
Ridders	Yes	7
Brent	Yes	8

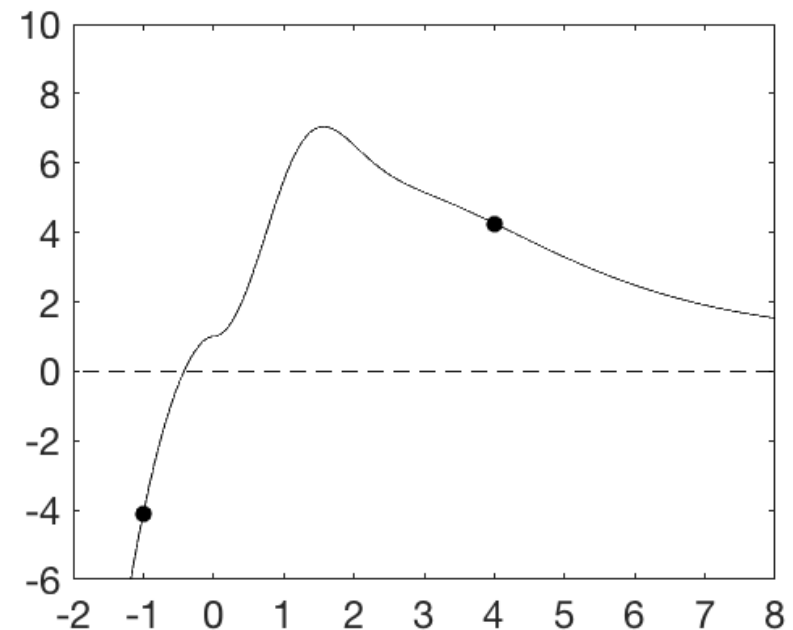


Comparing these methods

Solving $y = 5[\text{sign}(x) \cdot x^2]\{1 + \exp[-(x - 1)^2]\} \cdot \exp[-0.4(x + |x|)] + 1$

Initial values: $x_1=-1$, $x_2=4$; tolerance: $1\text{e-}14$.

Method	Success?	# iterations
Bisection	Yes	49
Secant	No	
False position	Yes	40
Inverse quadratic	No	
Ridders	Yes	6
Brent	Yes	10



Newton-Raphson method

The methods discussed so far, we have only assumed continuity.

If we further know the derivative of the function f , more efficient algorithms can be developed.

Starting from Taylor expansion:

$$f(x + \delta) \approx f(x) + f'(x)\delta + \frac{f''(x)}{2}\delta^2 + \dots$$

To leading order, it suggests an approximation to the root is

$$\delta = -\frac{f(x)}{f'(x)} \quad \text{or} \quad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

This is the **Newton-Raphson method**, or simply **Newton's method**.

Newton-Raphson method: convergence

The error from NR iteration follows the same relation:

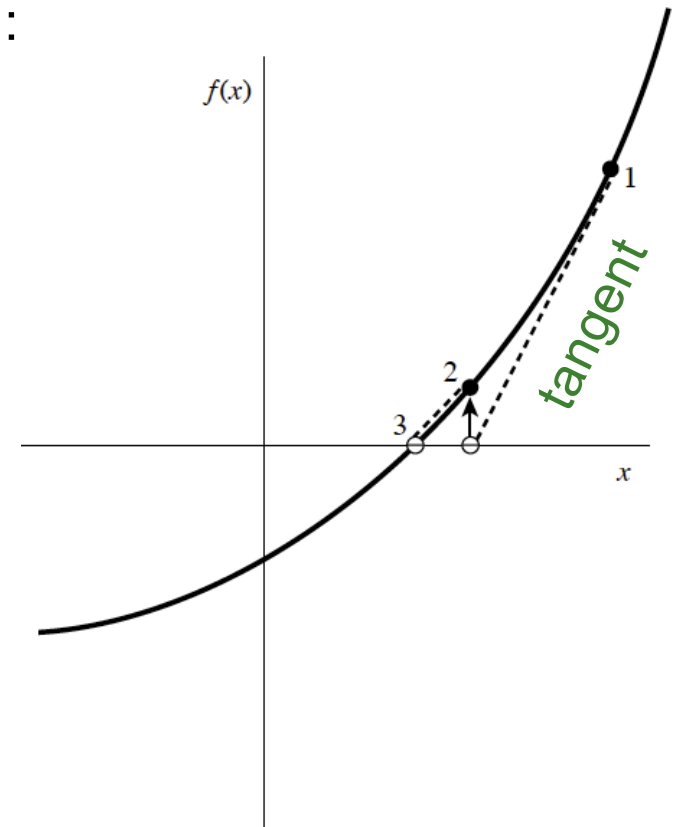
$$\epsilon_{k+1} = \epsilon_k - \frac{f(x_k)}{f'(x_k)}$$

Taylor expanding the function f near its root:

$$f(x_k) = 0 \quad \epsilon_k f'(x^*) + \frac{\epsilon_k^2}{2} f''(x^*) + \dots$$

$$f'(x_k) = f'(x^*) + \epsilon_k f''(x^*) + \dots$$

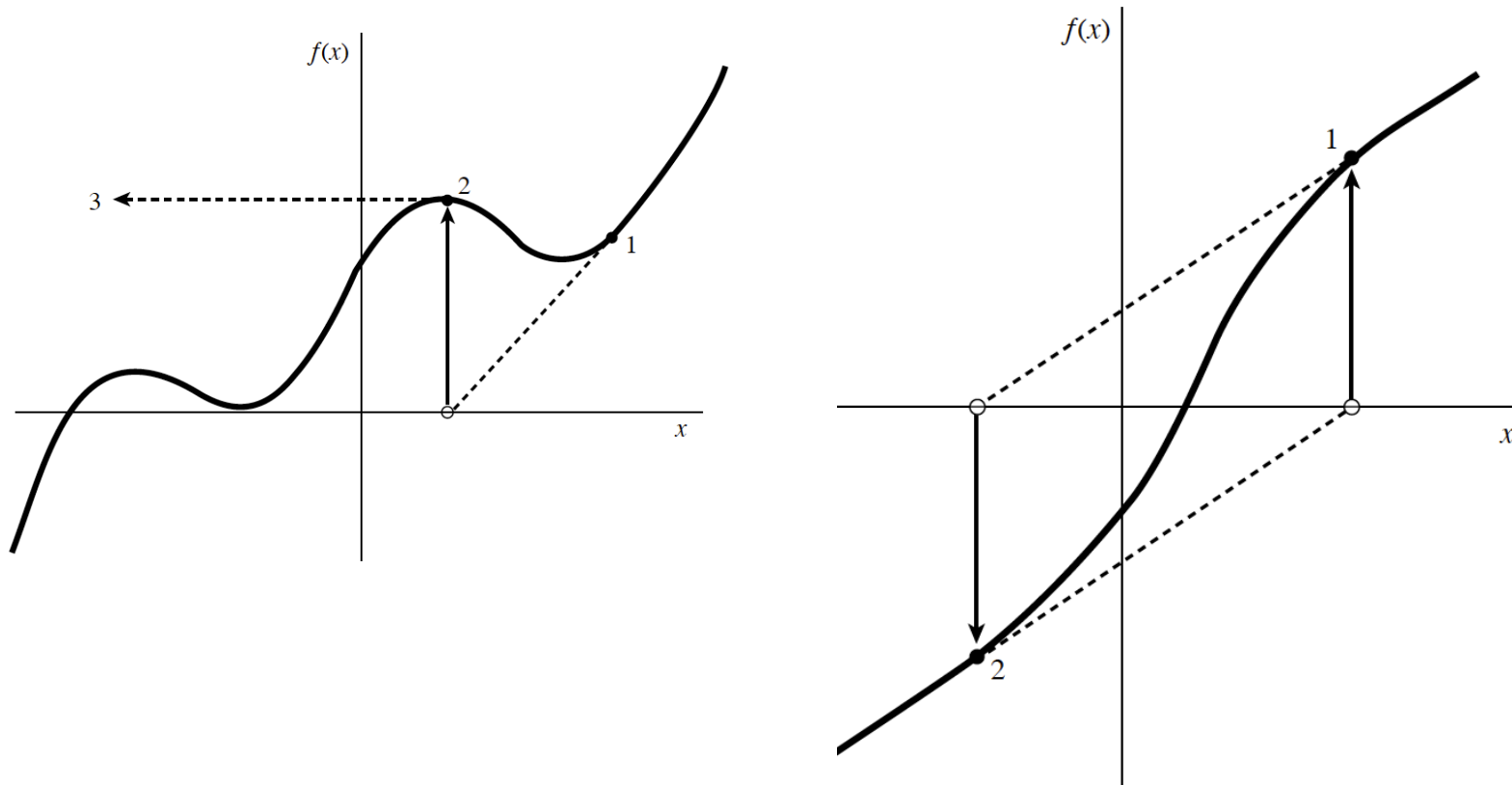
$$\Rightarrow \epsilon_{k+1} = -\epsilon_k^2 \frac{f''(x^*)}{2f'(x^*)}$$



Convergence is quadratic!

Near the root, # of significant digits doubles per iteration!

Newton-Raphson method can fail



When you are far from the root, higher order corrections become important, and Newton's method can easily fail.

Additional remarks

- It is a powerful method to **polish the root**.
- If the root is known to be bracketed, one can again design **hybrid algorithms that combine bisection with Newton's method**, which is both robust and efficient.
- When the derivative exists but not easy to compute, using numerical derivative is generally not recommended. The Brent method would be a better choice.
- It can also be used for finding complex roots.
- It can be generalized to multi-dimensions (see later).
- Higher-order variants also exist. To next order, it is known as **"Halley" method**. More generally, these variants are called **"Householder" methods**.

Discussion

How do calculators evaluate the square root \sqrt{a} ?

Solution: solve $f(x)=x^2-a=0$ by NR iteration.

Starting from an initial guess x_0 , iterate with:

$$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{1}{2}x_n + \frac{a}{2x_n}$$

Any problem?

Need to do division in each iteration: a bit slow...

Better approach: solve $f(x)=x^{-2}-a=0$ by NR iteration, then take the inverse.

Starting from an initial guess x_0 , iterate with:

$$x_{n+1} = x_n + \frac{x_n^{-2} - a}{2x_n^{-3}} = \frac{3}{2}x_n - \frac{1}{2}ax_n^3 \quad \Rightarrow \quad \sqrt{a} \approx 1/x^*$$

Just need to do division once.

Outline

- Root-finding for a single variable
 - Standard methods when only f is known
 - Newton's method
- Multivariable problems

System of non-linear equations

We want to solve: $\mathbf{F}(\mathbf{x}) = \mathbf{0}$

$$\text{or } F_i(x_1, \dots, x_N) = 0 \quad (i = 1, \dots, N)$$

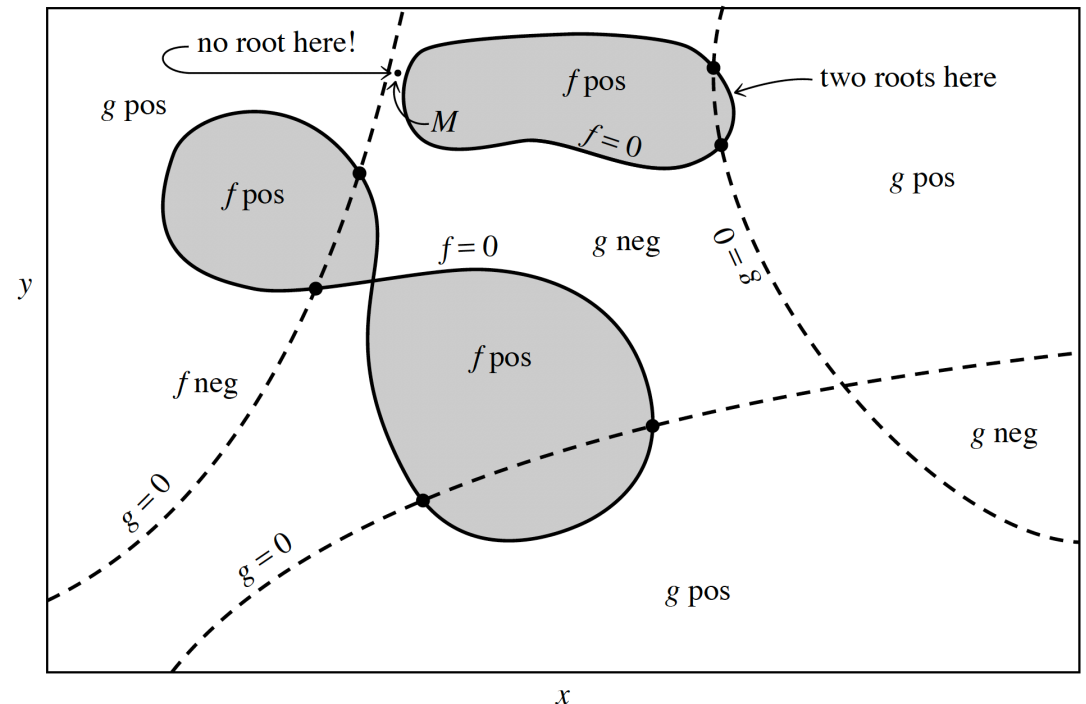
There are no good, general methods for solving systems of more than one nonlinear equations.

Even in 2D case:

$$f(x, y) = 0$$

$$g(x, y) = 0$$

We can end up into arbitrarily complex situation.



Multi-D Newton-Raphson method

Generalization of Newton-Raphson method to multi-D is straightforward.

For \mathbf{x} sufficiently close to a root, we Taylor expand $\mathbf{F}(\mathbf{x})$ to 1st order:

$$\mathbf{F}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{J} \cdot \delta\mathbf{x} + O(\delta\mathbf{x}^2)$$

where $J_{ij} = \frac{\partial F_i}{\partial x_j}$ is the **Jacobian matrix**. (use finite difference if J is difficult to compute analytically)

This suggests one can approximate the root by solving

$$\mathbf{J}(\mathbf{x})\delta\mathbf{x} = -\mathbf{F}(\mathbf{x})$$

At fixed \mathbf{x} , this is a linear system of equations and can be solved by, e.g., LU decomposition.

Iteration just proceeds as $\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \delta\mathbf{x}$

Towards global convergence

Like in 1D, convergence is quadratic near the root, but otherwise, the method can easily fail (e.g., diverge) if the initial guess is poor.

Ideally, we'd love to have a method that is **globally convergent**: guaranteed convergence from any starting point.

This can be partially achieved when **combining Newton-Raphson with a globally convergent strategy**.

An important fact about Newton's method is that its iteration step is **always towards a descent direction** of:


$$f \equiv \frac{1}{2} \mathbf{F} \cdot \mathbf{F}$$

To see this: $\nabla f \cdot \delta \mathbf{x} = -(\mathbf{J}\mathbf{F})^T (\mathbf{J}^{-1}\mathbf{F}) = -\mathbf{F} \cdot \mathbf{F} < 0$

Strategy: **adjust the length of a Newton step to optimally reduce f** .

Line searches and backtracking

Now we write $\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}} + \lambda(\Delta \mathbf{x})$



full Newton's step

Define $g(\lambda) \equiv f[\mathbf{x}_{\text{old}} + \lambda(\Delta \mathbf{x})]$, which guarantees that $g'(0) < 0$.

First check if a full Newton step ($\lambda=1$) manages to reduce f by “appropriate amount”, which requires:

$$g(\lambda) \leq g(0) + \alpha g'(0)\lambda$$

for some small α (e.g. 10^{-4} is sufficient).

If it fails, then “backtrack” to iteratively search for $0 < \lambda \leq 1$ until it is satisfied.

Done by approximating g as quadratic/cubic function of λ and derive its minimum.

Multi-D secant: Broyden's method

Drawback of Newton's method: need Jacobian or its finite difference approximation.

If derivatives not known, and evaluating f is expensive, it becomes prohibitive.

Even analytical Jacobian is available, it can be very costly to evaluate ($N \times N$ matrix).

Solving the linear system is an $O(N^3)$ process, again a bit slow.

Alternative solution:

Generalize the secant method to multi-D (often called quasi-Newton methods).

$$\mathbf{x}_{k+1} = \mathbf{x}_k - B_k^{-1} \mathbf{F}(\mathbf{x}_k) \quad B_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{F}(\mathbf{x}_{k+1}) - \mathbf{F}(\mathbf{x}_k)$$

However, B_{k+1} is underdetermined in multi-D.

Broyden's method: use the previous B_k to update B_{k+1} by making the least changes.

Multi-D secant: Broyden's method

Broyden's method: use the previous B_k to update B_{k+1} by making the least changes.

By minimizing $\|B_{k+1} - B_k\|_{\text{Fro}}$ subject to constraint, Broyden showed

$$B_{k+1} = B_k + \frac{1}{\|\delta \mathbf{x}_k\|_2} (\delta \mathbf{F}_k - B_k \delta \mathbf{x}_k) \times \delta \mathbf{x}_k^T$$

If B_k^{-1} is known, then B_{k+1}^{-1} can be expressed by the Sherman-Morrison formula.
Usually start the iteration from identity matrix.

Thus, the cost of iteration **only takes $O(N^2)$** instead of $O(N^3)$.

Convergence is super-linear near the root. But given it asks for much fewer function evaluations, it is usually much faster than Newton's method.

This method can also be embedded into a global strategy, by further requiring minimizing f similar to that in the global Newton's method.

Summary

■ Root-finding for a single variable

- Standard methods when only f is known

Robust root-finding requires bracketing.

Ridders and Brent (hybrid) methods are recommended.

- Newton's method

Quadratic convergence, should be bracketed for robustness.

■ Multivariable problems

Much more difficult. No general method and no guarantee of success.

Embed Newton-Raphson or Broyden's methods into a global strategy.