Solving nonlinear equations

Outline

- problem of finding the roots of nonlinear equation
- bissection method
- order of convergence of iterative methods
- Regula Falsi (False Point) method
- Secant Method
- Newton-Raphson method
- system of nonlinear equations
  (multidimensional Newton-Raphson method)

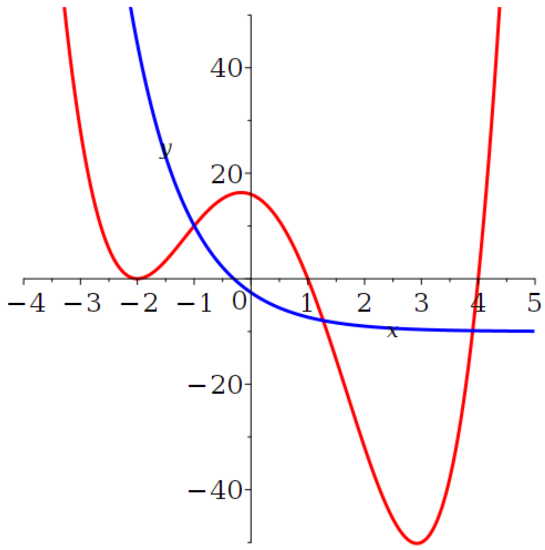
Roots/zeros of nonlinear equation

Let the function f(x) be nonlinear i.e. its Taylor expansion, fininte or not, contains higher terms than linear

$$f : \mathbb{R} \to \mathbb{R}$$
$$f(x) = a_0 + a_1 x + \underbrace{a_2 x^2 + a_x x^3 + \ldots}_{nonlinear}$$

and we seek the solution of nonlinear equation such that the set of discrete points (roots, zeros) fullfill

$$f(x) = 0 \quad \Longleftrightarrow \quad x \in \{x^{(1)}, x^{(2)}, \ldots, x^{(k)}\}, \; x \in \mathbb{R}$$



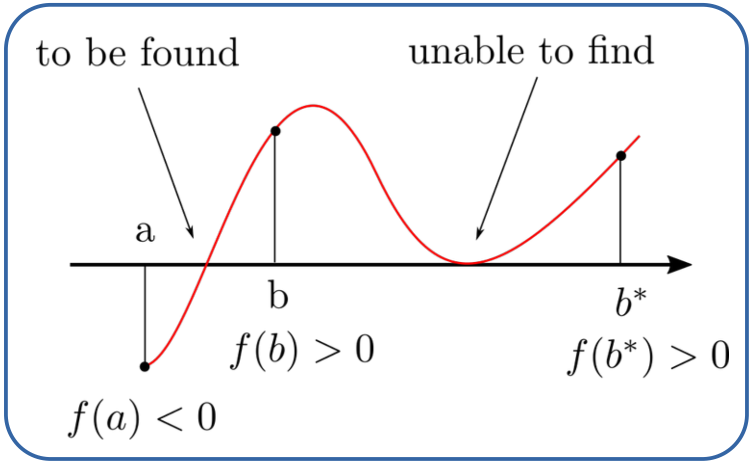
- generally, there are no analytic methods allowing to solve such problems exactly (besides low degree polynomials),
- solutions are found iteratively, we guess the first approximation and then update it in hope we get closer to given root

$$x_0 \to x_1 \to x_2 \to \ldots \to x_n$$

- the **stopping criterion** is as usual, iteration is interupted when two subsequent approximations are closer to each other than given threshold ε

$$|x_{k+1} - x_k| < \varepsilon$$

Bissection method (two -point method)



Algorithm is very intuitive

- define the region of expected localization of single root (**of odd order**)

$$x \in [x_a, x_b] \implies f(x_a) \cdot f(x_b) < 0$$

- half the segment

$$x_c = \frac{x_b - x_a}{2}$$

0-th iteration

$$x_a = x_{a_0}$$
$$x_b = x_{b_0}$$

- choose segment at which ends function value has opposite signs

$$f(x_a) \cdot f(x_c) < 0 \implies \text{replace: } x_b \leftarrow x_c$$
$$f(x_c) \cdot f(x_b) < 0 \implies \text{replace: } x_a \leftarrow x_c$$

new narrowed segment

$$x \in [x_{a_1}, x_{b_1}]$$

- repeat procedure of segment halving k-times until convergence condition is met

$$|x_{b_k} - x_{a_k}| < \varepsilon \implies x = \frac{x_{b_k} - x_{a_k}}{2} + O\left(\frac{1}{2}\Delta_k\right)$$

- the pace of convergence is defined by width of narrowing segment which isolates the root

$$\Delta_k = \frac{x_{b_0} - x_{a_0}}{2^k}$$

BISSECTION METHOD

```
input: x_a, x_b, itmax, ε

do
    k ← k + 1

    x_c ← (x_a + x_b) / 2

    if f(x_a) · f(x_c) < 0 then
        x_b ← x_c
    else
        x_a ← x_c
    end if

    !save approximated solution
    x ← x_c

    Δ ← |x_a − x_b|

while k < itmax & Δ > ε
```

Convergence of iterative methods

Set of subsequent approximations converges to the solution if

$$\lim_{k\to\infty} x_k = a, \quad f(a) = 0$$

with the error of approximation in k-th iteration

$$\varepsilon_k = a - x_k$$

The method is p-order at point x=a, if it converges to exact solution and following conditions are fullfilled

$$p \geq 1$$

$$\lim_{k\to\infty} \frac{|x_{k+1} - a|}{|x_k - a|^p} = \lim_{k\to\infty} \frac{|\varepsilon_{k+1}|}{|\varepsilon_k|^p} = C \neq 0$$

factor C is asymptotic error constant

$$|\varepsilon_{k+1}| = C|\varepsilon_k|^p$$

in other words, the larger is value of C,
the faster decreases the error in next iterations

example

$$\varepsilon_k = 0.01$$

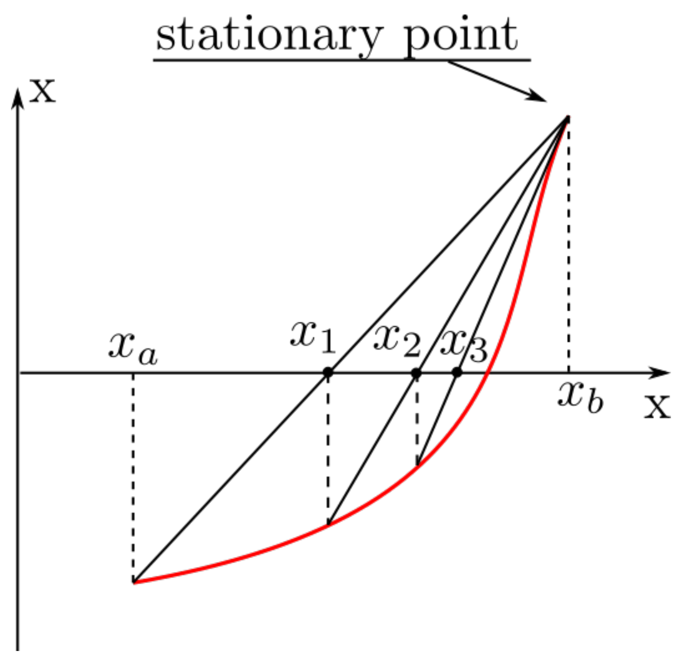
$$p = 1 \implies \varepsilon_{k+1} = 0.01$$

$$p = 2 \implies \varepsilon_{k+1} = 0.0001$$

Regula Falsi or False position method (two-point method)

In this method we linearize the function over the region of isolation of root (it is false assumption which gives the Latin name to the method)

Algorithm



- draw stright line between the end points of isolation segment

$$y(x) = ax + b$$

$$y(x) = \frac{f(x_b) - f(x_a)}{x_b - x_a}(x - x_b) + f(x_b)$$

- find intersection of stright line with y=0,
  it is approximate solution in present iteration (k+1)-th

$$x_{k+1} = x_b - f(x_b)\frac{x_b - x_a}{f(x_b) - f(x_a)}$$

- replace one of the segment ends by new approximation and repeat procedure iteratively

for **convex** function x_b is stationary

$$\frac{d^2 f}{dx^2} > 0 \qquad [x_a, x_b] \Longrightarrow x_a \leftarrow x_{k+1}$$

for **concave** function x_a is stationary

$$\frac{d^2 f}{dx^2} < 0 \qquad [x_a, x_b] \Longrightarrow x_b \leftarrow x_{k+1}$$

Remark: Regula Falsi is 1-order method (slow convergence)

Regula Falsi

$$\text{input}: \ x_a \, , \ x_b \, , \ \text{itmax} \, , \ \varepsilon \, , \ x = x_a$$

$$\text{do}$$

$$k \leftarrow k + 1$$

$$x_c \leftarrow x_b - f(x_b)\frac{x_b - x_a}{f(x_b) - f(x_a)}$$

one of end point is stationary but we needn’t to know which one, it is chosen automatically

$$\text{if} \ f(x_a) \cdot f(x_c) < 0 \ \text{then}$$

$$x_b \leftarrow x_c$$

$$\text{else}$$

$$x_a \leftarrow x_c$$

$$\text{end if}$$

$$\text{!save approximated solution}$$

$$x \leftarrow x_c$$

$$\Delta \leftarrow |x - x_c|$$

$$\text{while} \ k < itmax \ \& \ \Delta > \varepsilon$$

Secant method (two-point method)

It is derived from Regula Falsi scheme, both points x_a and x_b are replaced by the last two approximations

Regula Falsi:
$$x_{k+1} = x_b - f(x_b)\frac{x_b - x_a}{f(x_b) - f(x_a)}$$

replacement:
$$\begin{cases} x_b = x_k \\ x_a = x_{k-1} \end{cases}$$

Secant method:
$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

Remarks:

- order of Regula Falsi $p = 1$
- order of Secant method $p = \dfrac{1+\sqrt{5}}{2} \approx 1.618$
- during the course of iterative process we must monitor if the set of $|f(x_k)|$ is decreasing, otherwise calculations should be restarted with different two initial points

Secant Method

input: x_0, x_1, itmax, ε

do

$k \leftarrow k+1$

$x_2 \leftarrow x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$

!save data for next iteration

$x_0 \leftarrow x_1$

$x_1 \leftarrow x_2$

$\Delta \leftarrow |x_1 - x_0|$

!save approximated solution

$x \leftarrow x_1$

while $k < itmax$ & $\Delta > \varepsilon$

Example: finding positive value root of nonlinear equation with **Regula Falsi** and **Secant** methods

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

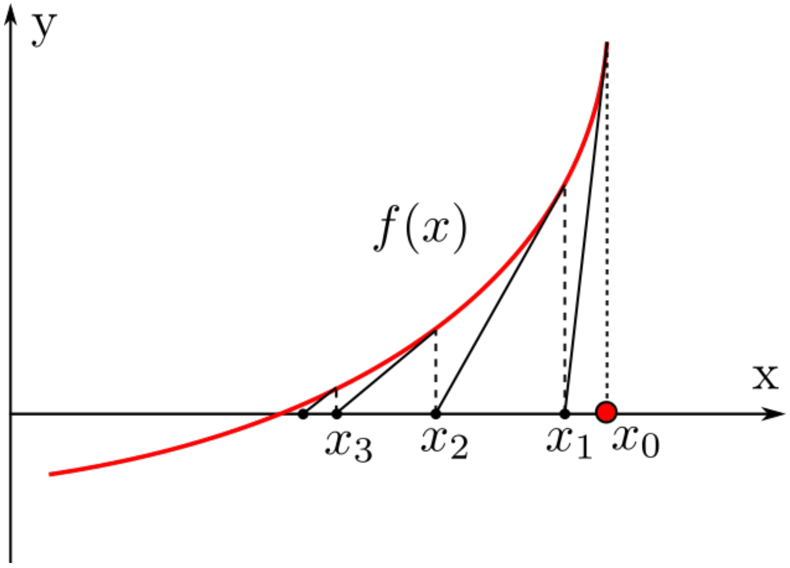
starting points are the same in both methods:

$$x_1 = \pi/2$$

$$x_2 = \pi$$

	Regula Falsi	**Secant method**
x_3	1.75960	1.75960
x_4	1.84420	1.93200
x_5	1.87701	1.89242
x_6	1.88895	1.89543
x_7	1.89320	**1.89549**
x_8	1.89469	
x_9	1.89521	
x_{10}	1.89540	
x_{11}	1.89546	
x_{12}	1.89548	
x_{13}	**1.89549**	

Newton-Raphson method (one-point metod)



Algorithm

- choose a point x_0, at this point calculate the tangent of nonlinear function f(x)

$$y(x) = ax + b$$

$$y(x) = f(x_0) + (x - x_0) f'(x_0), \qquad f'(x_0) = \left. \frac{d^2 f}{dx^2} \right|_{x=x_0}$$

- find the point x at which tangent value is 0

$$y(x) = 0 \quad \Rightarrow \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

- choose x_1 as a new approximated solution and repeat procedure with iterative equation

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

Remarks:

- Newton-Raphson method is 2 order (the fastest method)
- it requires the knowledge of the first derivative, this could be calulated analytically (best option) or approximated with other methods (finite difference, dual numbers etc)
- it can find both, even and odd, parity roots

Newton-Raphson Method

```
input: x, itmax, ε

do
      k ← k + 1
      x1 ← x − f(x)/f'(x)

      Δ ← |x − x1|

      !save approximate solution
      x ← x1

while k < itmax & Δ > ε
```

Example: find root of nonlinear function with Regula Falsi, Secant and Newton-Raphson methods

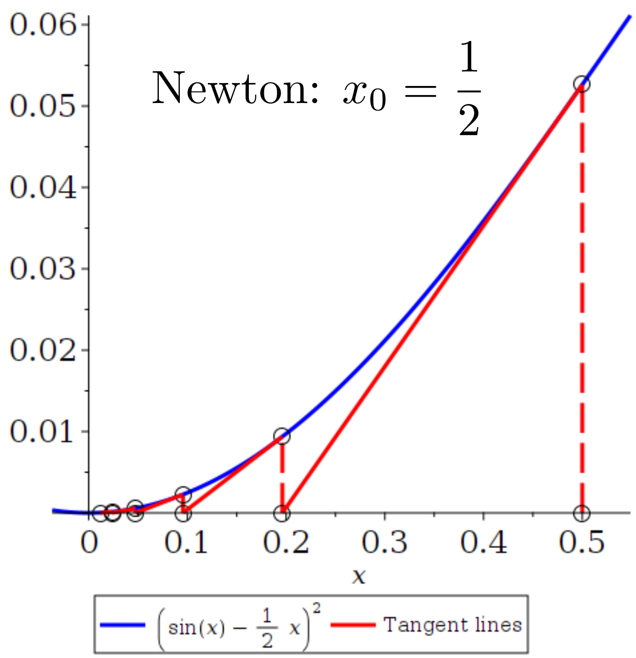
$$f(x) = \left(\sin(x) - \frac{x}{2}\right)^2 = 0$$

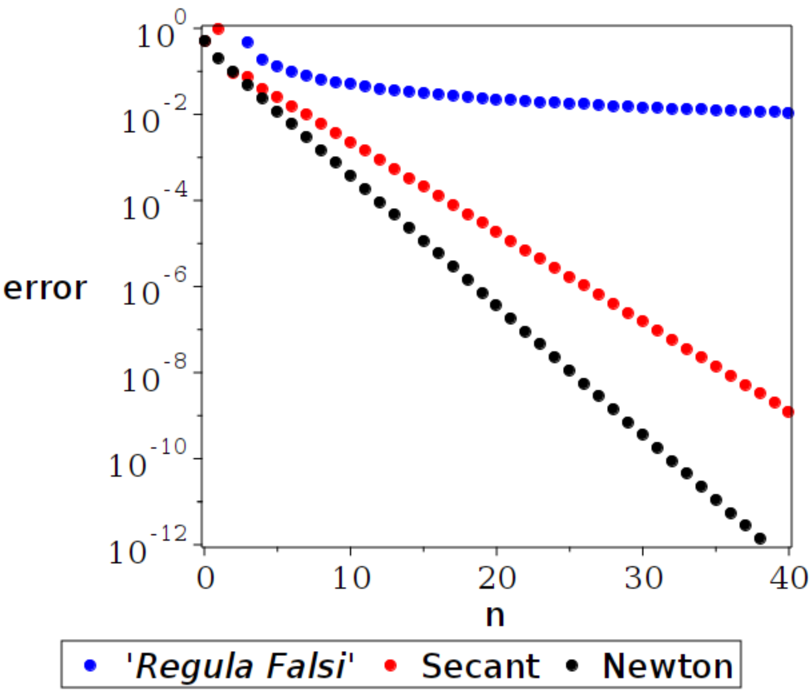
$$x_{exact} = 0$$

$$\text{Newton: } x_0 = \frac{1}{2}$$

$$\text{Regula Falsi: } x_0 = -\frac{1}{4}, \quad x_{stationary} = \frac{1}{2}$$

$$\text{Secant: } x_0 = \frac{1}{2}, \quad x_1 = 1$$





Systems of nonlinear equations and multidimensional Newton-Raphson method

Let's consider system of n nonlinear equations depending on n variables

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \quad\vdots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases} \longrightarrow \quad \vec{f}(\vec{x}) = \vec{0} \quad \longrightarrow \quad \vec{x} = \vec{f}^{-1}(\vec{x}) = \vec{g}(\vec{y})$$

vector notation

we assume inverse function exists

We may find Taylor expansion of inverse function around some point
(e.g. given in k-th iteration) near the exact solution

$$\vec{x}_k = [x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}] \quad \Longleftrightarrow \quad \vec{y}_k = [y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)}]$$

from Taylor series we retain only terms up to first order

$$\vec{x} \approx \vec{g}(\vec{y}_k) + \sum_{j=1}^{n} \left.\frac{\partial \vec{g}(\vec{y})}{\partial\, y_j}\right|_{y_j = y_j^{(k)}} \left(y_j - y_j^{(k)}\right)$$

now we assume the point at which we approximate the function would be an exact solution
(actually it is not the truth)

$$\vec{y} = 0 \quad \Longrightarrow \quad \vec{x} \approx \vec{x}_k - \sum_{j=1}^{n} \left.\frac{\partial \vec{g}(\vec{y})}{\partial\, y_j}\right|_{y_j = y_j^{(k)}} y_j^{(k)}$$

The last equation would be interpreted as an iterative matrix equation, it is a multidimensional counterpart of Newton-Raphson method

$$\vec{x}_{k+1} \approx \vec{x}_k - D_y \vec{y}_k$$

$$D_y = [d_{i,j}] \quad \Longrightarrow \quad d_{i,j} = \left. \frac{\partial g_i(\vec{y})}{\partial\, y_j} \right|_{y_j = y_j^{(k)}}$$

The last problem we must solve before we could use this iterative formula is to find the derivatives of unknown inverse function.

In order to derive matrix D we consider differential of vector function **f** and its inverse **g**

differential of **f**

$$d\vec{f} = \left(\nabla_{\vec{x}} \vec{f}\right) d\vec{x}$$

$$d\vec{f} = D_x d\vec{x} \qquad D_x^{-1} \cdot /$$

$$d\vec{x} = D_x^{-1} d\vec{f}$$

differential of **g**

$$d\vec{x} = (\nabla_{\vec{y}} \vec{g})\, d\vec{y}$$

$$d\vec{x} = D_y d\vec{y}$$

we know how to calculate these derivatives

$$d\vec{f} \equiv d\vec{y} \quad \Longrightarrow \quad D_y = D_x^{-1} = \left[\frac{\partial f_i}{\partial x_j}\right]^{-1} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}^{-1}$$

Finaly we get the multidimensional version of iterative Newton-Raphson method

$$\vec{x}_{k+1} = \vec{x}_k - \left(\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \dots & \dots & \dots & \dots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}^{-1} \begin{bmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_n(\vec{x}) \end{bmatrix} \right)_{\vec{x}=\vec{x}_k}$$

Remarks:

- method works effectively only if the starting point is chosen near exact solution, finding the solution is not guaranteed
- stopping criterion must be fullfilled by every element of solution vector **x**
- if solution can’t be find during the iterative process, it must be restarted with different starting vector **x**
- matrix of derivatives must be recalculated in every iteration
- insted of comupting the inverse matrix, system of linear equation should be solved for example using LU decoposition (matrix is not symmetric)

$$D_x^{-1} \vec{f} = \vec{\Delta}_x \qquad \Longrightarrow \qquad D_x \vec{\Delta}_x = \vec{f}$$