FPT UNIVERSITY

Computer project MAN201 CS1302

Authors: Lê Đình Duy
Ngô Tuấn Anh
Nguyễn Việt Tùng
Phạm Cao Bằng
Hoàng Mạnh
Tạ Đăng Khoa

Source code: https://github.com/daemon-Lee/numerical_project.git

1. Newton Method

In numerical analysis, Newton's method, also known as the Newton–Raphson method, named after Isaac Newton and Joseph Raphson, is a root-finding algorithm which produces successively better approximations to the roots (or zeroes) of a real-valued function. The most basic version starts with a single-variable function f defined for a real variable x, the function's derivative f ′, and an initial guess x0 for a root of f. If the function satisfies sufficient assumptions and the initial guess is close, then

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

is a better approximation of the root than x0. Geometrically, (x1, 0) is the intersection of the x-axis and the tangent of the graph of f at (x0, f (x0)): that is, the improved guess is the unique root of the linear approximation at the initial point. The process is repeated as

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

until a sufficiently precise value is reached. This algorithm is first in the class of Householder's methods, succeeded by Halley's method. The method can also be extended to complex functions and to systems of equations.

Example:

function f(x):

$$x^5 - x^4 + 2 \times x^3 - 3 \times x^2 + x - 4$$

Df(X):

$$5 \times x^4 - 4 \times x^3 + 6 \times x^2 - 6 \times x + 1$$

with:

Max iteration = 100

Error tolerance eps = 1e-4

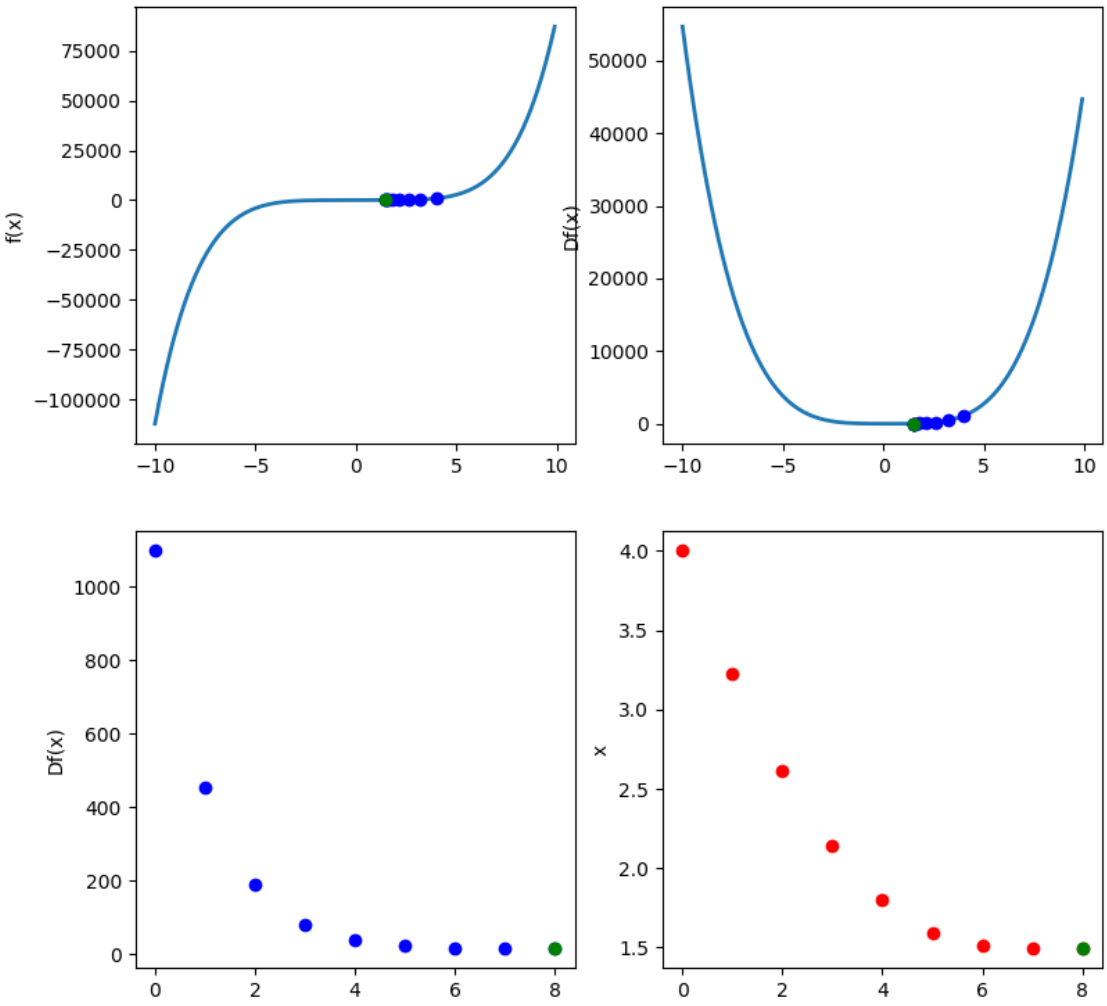
Output of Newton method

Steps	**approximation**
1	4.0
2	3.226982680036463
3	2.6147069857653715
4	2.1408909178607676
5	1.7972980382351813
6	1.5887914164129464
7	1.5090430997274666
8	1.4983657746916184
9	1.4981900316451104

Found solution after 9 iterations.

1.4981900316451104

Result:


f(x)
Df(x)
Df(x)
x

2. Muller Method

Muller's method is a root-finding algorithm, a numerical method for solving equations of the form f(x) = 0. It was first presented by David E.Muller in 1956. Muller's method is based on the secant method, which constructs at every iteration a line through two points on the graph of f. Instead, Muller's method uses three points, constructs the parabola through these three points, and takes the intersection of the x-axis with the parabola to be the next approximation.

Example:

function f(x):

$$x^5 - x^4 + 2 \times x^3 - 3 \times x^2 + x - 4$$

with

p0 = 0.0 p1 = 1.0 p2 = 2.0

Max iteration n = 100

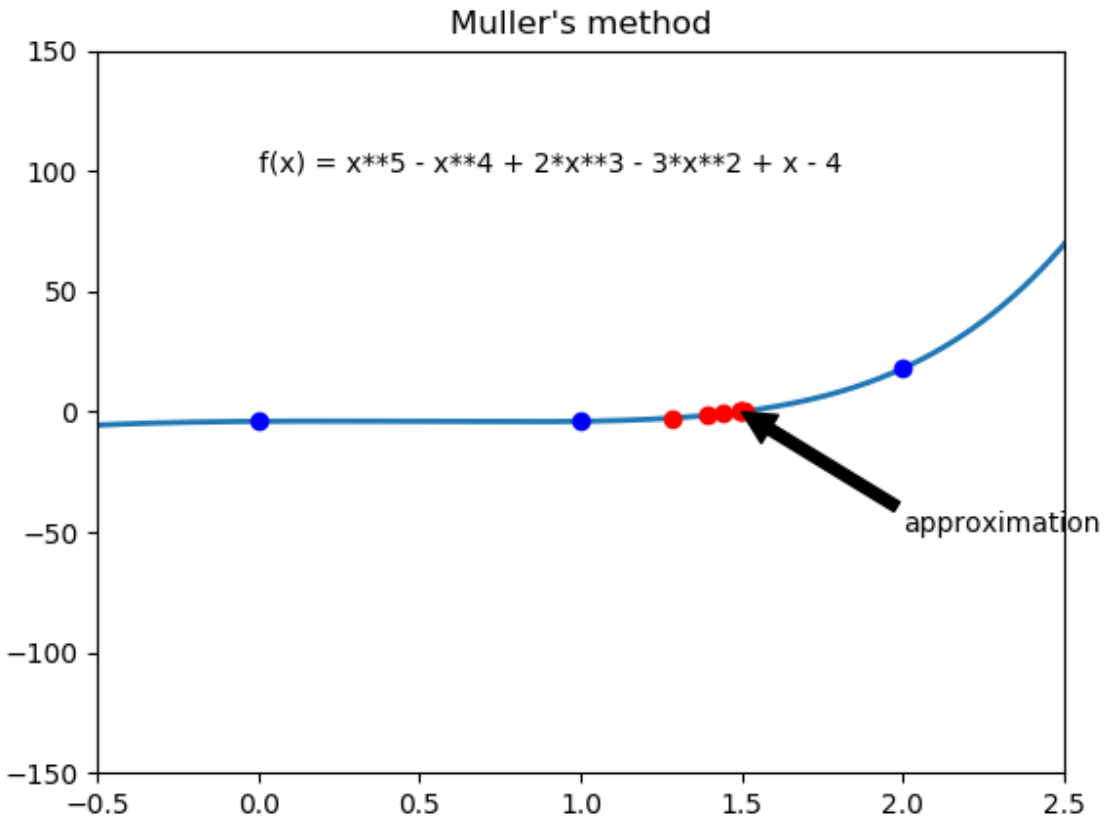
Error tolerance eps = 1e-3

Output of Muller method

Steps	**approximation**
1	1.283349451800640
2	1.391230395437434
3	1.442861630197730
4	1.507322949287101
5	1.496752919304598
6	1.498417889910505

Current approximation at 6 is 1.498417889910505

Result:



3. Jacobi Method

In numerical linear algebra, the Jacobi method is an iterative algorithm for determining the solutions of a diagonally dominant system of linear equations. Each diagonal element is solved for, and an approximate value is plugged in. The process is then iterated until it converges. This algorithm is a stripped-down version of the Jacobi transformation method of matrix diagonalization. The method is named after Carl Gustav Jacob Jacobi.

Example:

for matrix A

4	1	1	0	1
-1	-3	1	1	0
2	1	5	-1	-1
-1	-1	-1	4	0
0	2	-1	1	4

With output b [6.0, 6.0, 6.0, 6.0, 6.0]

inti x [1.0, 1.0, 1.0, 1.0, 1.0]

Max iteration n = 100

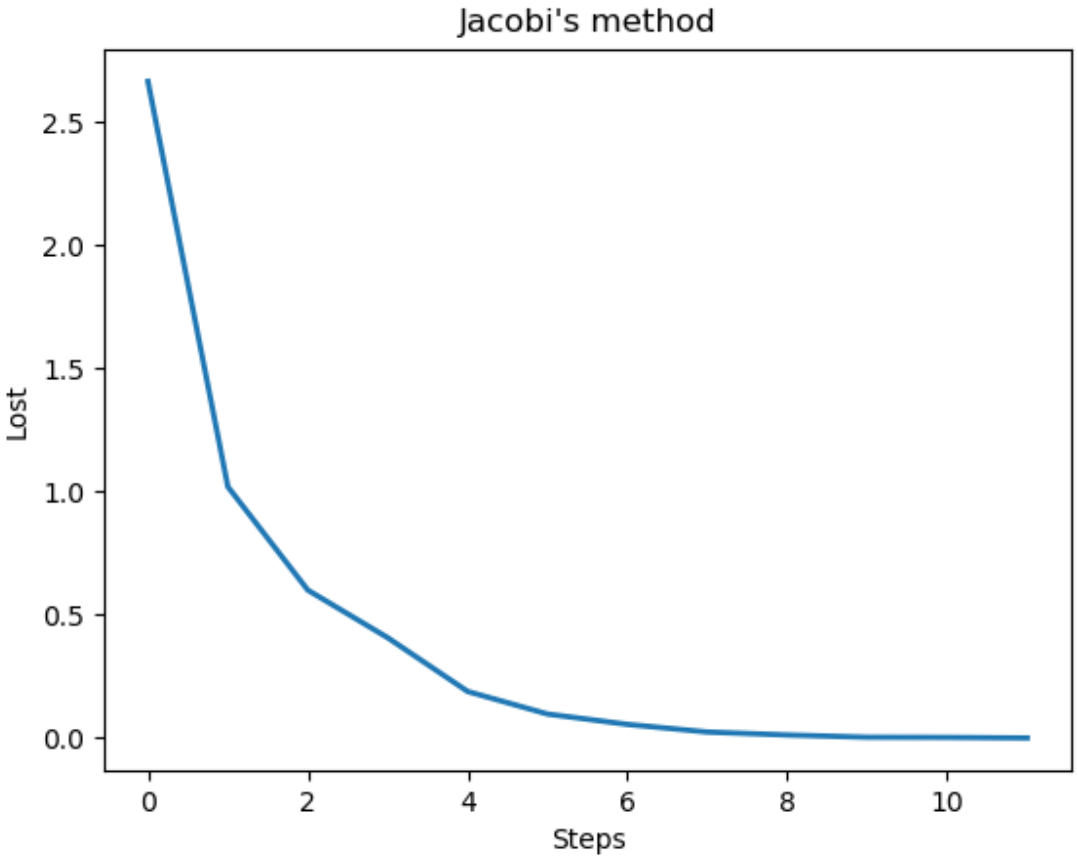
Error tolerance eps = 1e-3

Output of Jacobi method

Steps	**X0**	**X1**	**X2**	**X3**	**X4**
1	0.75	-1.66666667	1.	2.25	1.
2	1.41666667	-1.16666667	1.88333333	1.52083333	2.02083333
3	0.815625	-1.3375	1.575	2.03333333	2.17395833
4	0.89713542	-1.06909722	1.98270833	1.76328125	2.05416667
5	0.75805556	-1.05038194	1.81845486	1.95268663	2.08940538
6	0.78563043	-0.99563802	1.91527257	1.88153212	1.99163303
7	0.77218311	-0.99627525	1.85950846	1.92631624	2.00625412
8	0.78262816	-0.9954528	1.87689588	1.90885408	1.98143568
9	0.78428031	-0.9989594	1.86409725	1.91601781	1.98973685
10	0.78628133	-1.00138842	1.86723069	1.91235454	1.98649956
11	0.78691454	-1.00223203	1.86553597	1.9130309	1.98941325
12	0.7868207	-1.00278256	1.86616942	1.91255462	1.98924228

solve after 12 steps: [0.7868207 -1.00278256 1.86616942 1.91255462 1.98924228]

Result:



4. Gauss-seidel Method

In numerical linear algebra, the Gauss–Seidel method, also known as the Liebmann method or the method of successive displacement, is an iterative method used to solve a linear system of equations. It is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel, and is similar to the Jacobi method. Though it can be applied to any matrix with non-zero elements on the diagonals, convergence is only guaranteed if the matrix is either diagonally dominant, or symmetric and positive definite. It was only mentioned in

a private letter from Gauss to his student Gerling in 1823. A publication was not delivered before 1874 by Seidel.

Example:

For matrix A

4	1	1	0	1
-1	-3	1	1	0
2	1	5	-1	-1
-1	-1	-1	4	0
0	2	-1	1	4

With output b [6.0, 6.0, 6.0, 6.0, 6.0]

Inti x [1.0, 1.0, 1.0, 1.0, 1.0]

Max iteration n = 100

Error tolerance eps = 1e-3

Output of Gauss-seidel method

Steps	**X0**	**X1**	**X2**	**X3**	**X4**
1	1.5	-2.5	1.1	1.525	2.64375
2	1.1890625	-1.52135417	1.86239583	1.88252604	2.25564453
3	0.85082845	-1.03530219	1.89436317	1.92747236	1.98219533
4	0.7828913	-0.98701859	1.87161643	1.91687229	1.98219533
5	0.78330171	-0.998271	1.86614704	1.91279444	1.98747365
6	0.78616258	-1.00240703	1.86606999	1.91245638	1.98960692
7	0.78668253	-1.00271872	1.86628339	1.9125618	1.98978976

solve after 7 steps: [0.78668253 -1.00271872 1.86628339 1.9125618 1.98978976]

Result:

References