NUMERICAL ANALYSIS

Project

MAY 16, 2022

DHRUV MAHESHWARI
2019A7PS0020U

Acknowledgements

I owe a huge debt of gratitude to everyone who assisted me in this endeavour.

My sincere gratitude to my project supervisor, Dr. K Kumar for providing me with the opportunity to work on such an educational project, for believing in me and supporting me throughout the project.

I also wish to express my heartfelt gratitude to Prof. Srinivasan Madapusi, Director of BITS Pilani, Dubai Campus, who has always steered me in the correct direction.

Index

Python

Python is a general-purpose, high-level, interpreted programming language.

It was designed by Guido van Rossum and released in 1991.

Its design philosophy prioritizes code readability and makes extensive use of indentation; hence it is one of the most widely used programming language.

Figure 1: Python logo

Python is garbage-collected and dynamically typed. It supports a variety of programming paradigms, including structured (especially procedural) programming, object-oriented programming, and functional programming. Because of its extensive standard library, it is often referred to as a "batteries included" language.

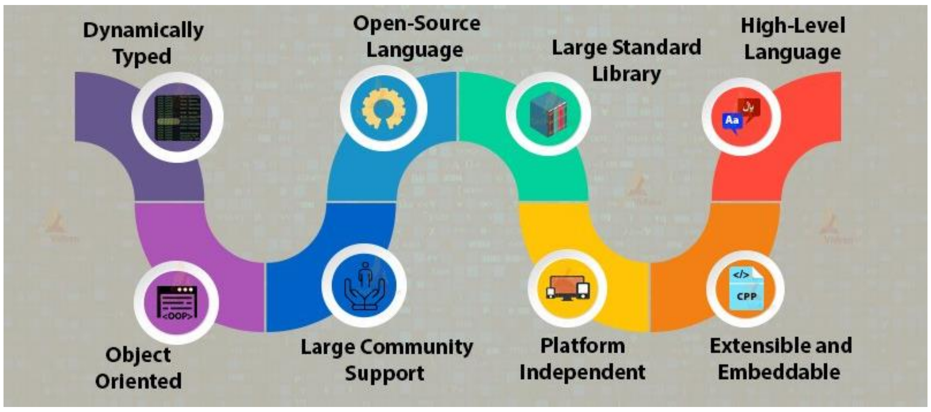


Figure 2: Python features

Bisection method

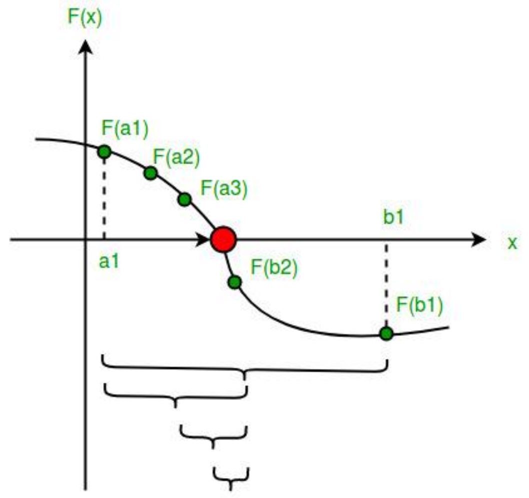


Figure 3: Graphical representation of Bisection method

To locate the roots of a polynomial problem, we utilize the bisection method. It divides and divides the interval in which the root of the equation is located.

It operates by gradually closing the gap between the positive and negative periods until the proper solution is found. By averaging the positive and negative intervals, this approach narrows the gap. It's a basic procedure that takes a long time.

Let us consider a continuous function "f" which is defined on the closed interval [a, b], is given with f(a) and f(b) of different signs. There exists a point x belongs to (a, b) for which f(x) = 0.

Algorithm:

For any continuous function f(x),

- Find two points, say a and b such that a < b and f(a)* f(b) < 0
- Find the midpoint of a and b, say "c" ie c = $\frac{(a+b)}{2}$
- c is the root of the given function if f(c) = 0; else follow the next step
- Divide the interval [a, b] – If f(c)*f(a) <0, there exist a root between c and a – else if f(c) *f (b) < 0, there exist a root between c and b
- Repeat above three steps until f(c) = 0.

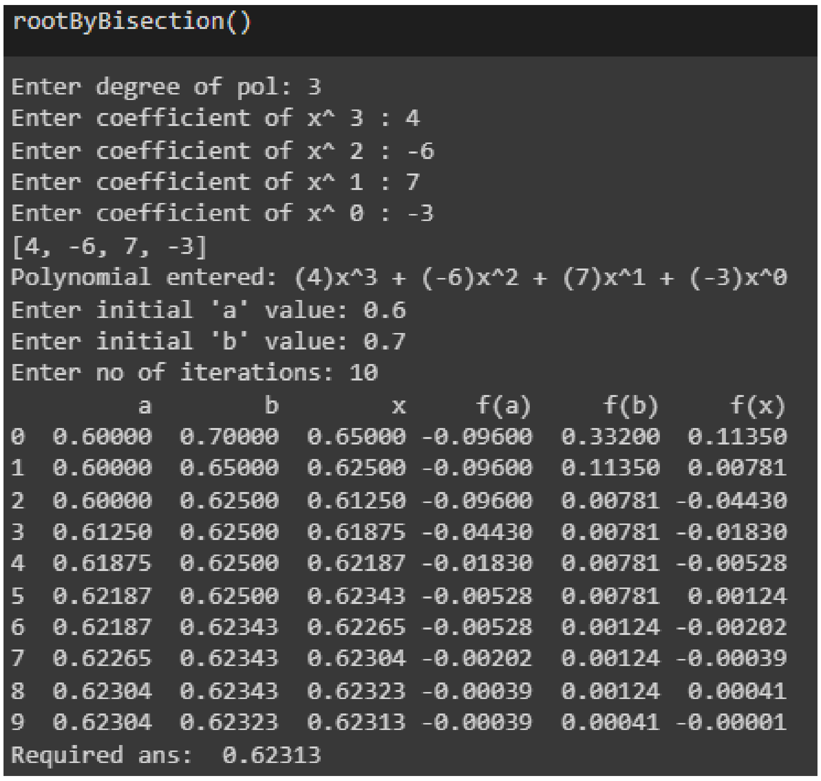
<u>Uses of finding real root by method of bisection:</u>

It helps in finding x such that f(x) = 0. It helps in finding solution to any mathematical function/model to solve a real life problem like calculating speed vector, force, momentum, current, magnetism etc.

<u>Ques:</u> We are given a function f(x) = 4x3 – 6x2 + 7x – 3 where 'x' is the distance and 'f(x)' is the velocity. We have to find out after traveling how much distance will velocity or f(x) be equal to zero. Through our previous tests, the physicist estimates the value of x to be between 0.6 and 0.7. Can we help him?

The code can find out x for f(x) = 0 where f(x) is any polynomial with degree n.

We have to input polynomial, initial approximations a and b and number of iterations required.



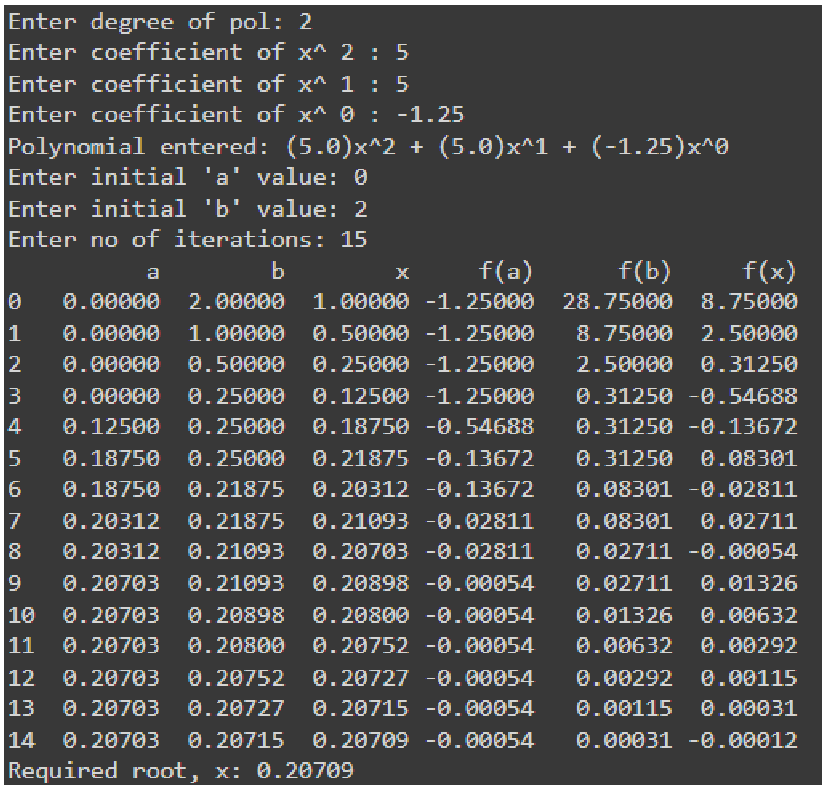
```
rootByBisection()

Enter degree of pol: 3
Enter coefficient of x^ 3 : 4
Enter coefficient of x^ 2 : -6
Enter coefficient of x^ 1 : 7
Enter coefficient of x^ 0 : -3
[4, -6, 7, -3]
Polynomial entered: (4)x^3 + (-6)x^2 + (7)x^1 + (-3)x^0
Enter initial 'a' value: 0.6
Enter initial 'b' value: 0.7
Enter no of iterations: 10
         a        b        x     f(a)     f(b)     f(x)
0  0.60000  0.70000  0.65000 -0.09600  0.33200  0.11350
1  0.60000  0.65000  0.62500 -0.09600  0.11350  0.00781
2  0.60000  0.62500  0.61250 -0.09600  0.00781 -0.04430
3  0.61250  0.62500  0.61875 -0.04430  0.00781 -0.01830
4  0.61875  0.62500  0.62187 -0.01830  0.00781 -0.00528
5  0.62187  0.62500  0.62343 -0.00528  0.00781  0.00124
6  0.62187  0.62343  0.62265 -0.00528  0.00124 -0.00202
7  0.62265  0.62343  0.62304 -0.00202  0.00124 -0.00039
8  0.62304  0.62343  0.62323 -0.00039  0.00124  0.00041
9  0.62304  0.62323  0.62313 -0.00039  0.00041 -0.00001
Required ans:  0.62313
```

Figure 4: Finding root for f(x) = 4x3 -6x2 +7x -3

Ques 2: A stone is thrown in a vertically upward direction with a velocity of 5 m/s^2.If the acceleration of the stone during its motion is 10m/s^2 in the downward direction. How much time will it take to reach a height of 1.25 m?

Here,

s = 1.25 m
u = 5 m/s
g = -10 m/s^2
we have to find t by using $s = ut + {}^1/_2 gt^2$
arranging we get, ${}^1/_2 10t^2 + 5t - 1.25 = 0$



```
Enter degree of pol: 2
Enter coefficient of x^ 2 : 5
Enter coefficient of x^ 1 : 5
Enter coefficient of x^ 0 : -1.25
Polynomial entered: (5.0)x^2 + (5.0)x^1 + (-1.25)x^0
Enter initial 'a' value: 0
Enter initial 'b' value: 2
Enter no of iterations: 15
          a        b        x      f(a)      f(b)     f(x)
0   0.00000  2.00000  1.00000 -1.25000  28.75000  8.75000
1   0.00000  1.00000  0.50000 -1.25000   8.75000  2.50000
2   0.00000  0.50000  0.25000 -1.25000   2.50000  0.31250
3   0.00000  0.25000  0.12500 -1.25000   0.31250 -0.54688
4   0.12500  0.25000  0.18750 -0.54688   0.31250 -0.13672
5   0.18750  0.25000  0.21875 -0.13672   0.31250  0.08301
6   0.18750  0.21875  0.20312 -0.13672   0.08301 -0.02811
7   0.20312  0.21875  0.21093 -0.02811   0.08301  0.02711
8   0.20312  0.21093  0.20703 -0.02811   0.02711 -0.00054
9   0.20703  0.21093  0.20898 -0.00054   0.02711  0.01326
10  0.20703  0.20898  0.20800 -0.00054   0.01326  0.00632
11  0.20703  0.20800  0.20752 -0.00054   0.00632  0.00292
12  0.20703  0.20752  0.20727 -0.00054   0.00292  0.00115
13  0.20703  0.20727  0.20715 -0.00054   0.00115  0.00031
14  0.20703  0.20715  0.20709 -0.00054   0.00031 -0.00012
Required root, x: 0.20709
```

Figure 5: Finding t

Gauss Jacobi method

One of the iterative approaches for approximating the solution of a system of n linear equations in n variables is the Jacobian method. The Jacobi iterative method is an iterative procedure used in numerical linear algebra to find solutions to diagonally dominant systems of linear equations. For each diagonal element, an approximate value is filled in using this procedure. The process is iterated until it converges. The Jacobi transformation process of matrix diagonalization was the name given to this approach at first. Simultaneous displacement method is another name for the Jacobi method.

Let there be n equations of the for

$$
\begin{array}{c}
a_{11}x_1 + a_{12}x_2 + \cdots a_{1n}x_n = b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots a_{2n}x_n = b_2 \\
\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
a_{n1}x_1 + a_{n2}x_2 + \cdots a_{nn}x_n = b_n
\end{array}
$$

The solution to variables x_i is given by:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \ldots, n.$$

<u>Uses of Gauss Jacobi method:</u>

It helps to converge out initial assumption to convergence in solving a system of linear equations. As linear equations describe relationships between two variables in the physical world, make predictions, calculate rates etc, Gauss Jordan helps to solve n variable equations very quickly.

<u>Ques 1:</u> Solving Kirchhoffs Circuit Law:

We have to find the current I_1, I_2 and I_3 in the circuit shown below.

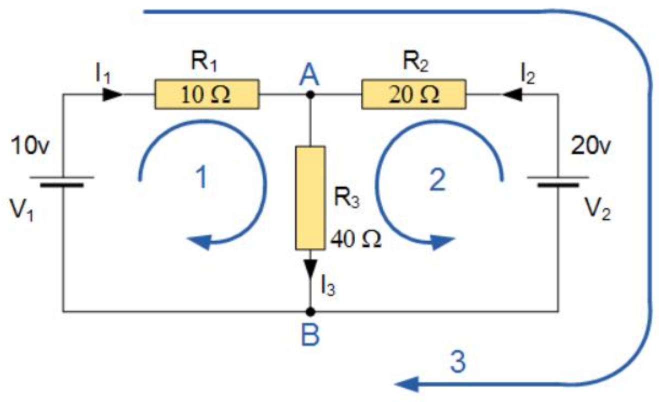
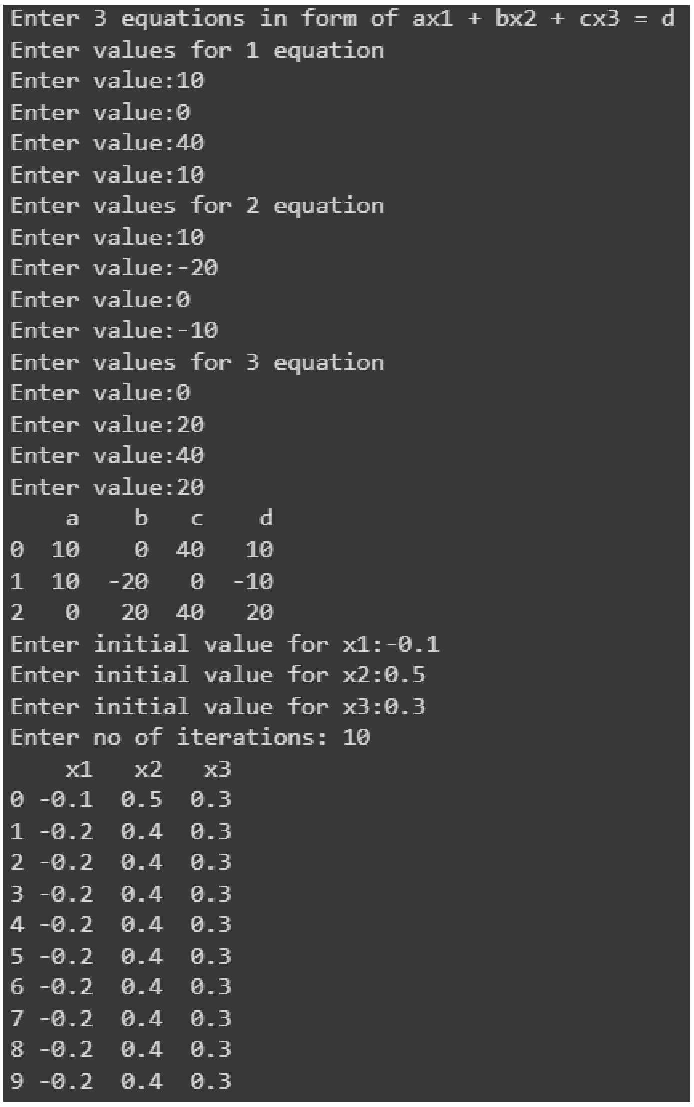


Figure 6: Electrical circuit having currents I1, I2, I3

Solving 3 variable equation:

- Loop1: $10I_1 + 0I_2 + 40I_3 = 10$ (1)
- Loop 2: $0I_1 + 20I_2 + 40I_3 = 20$ (3)
- Loop3: $10I_1 - 20I_2 + 0x_3 = 10-20 = -10$ (2) for diagonally dominant

Let initial values be $I_1 = -0.1$, $I_2 = 0.5$, $I_3 = 0.3$



```
Enter 3 equations in form of ax1 + bx2 + cx3 = d
Enter values for 1 equation
Enter value:10
Enter value:0
Enter value:40
Enter value:10
Enter values for 2 equation
Enter value:10
Enter value:-20
Enter value:0
Enter value:-10
Enter values for 3 equation
Enter value:0
Enter value:20
Enter value:40
Enter value:20
    a    b   c    d
0  10    0  40   10
1  10  -20   0  -10
2   0   20  40   20
Enter initial value for x1:-0.1
Enter initial value for x2:0.5
Enter initial value for x3:0.3
Enter no of iterations: 10
    x1   x2   x3
0 -0.1  0.5  0.3
1 -0.2  0.4  0.3
2 -0.2  0.4  0.3
3 -0.2  0.4  0.3
4 -0.2  0.4  0.3
5 -0.2  0.4  0.3
6 -0.2  0.4  0.3
7 -0.2  0.4  0.3
8 -0.2  0.4  0.3
9 -0.2  0.4  0.3
```

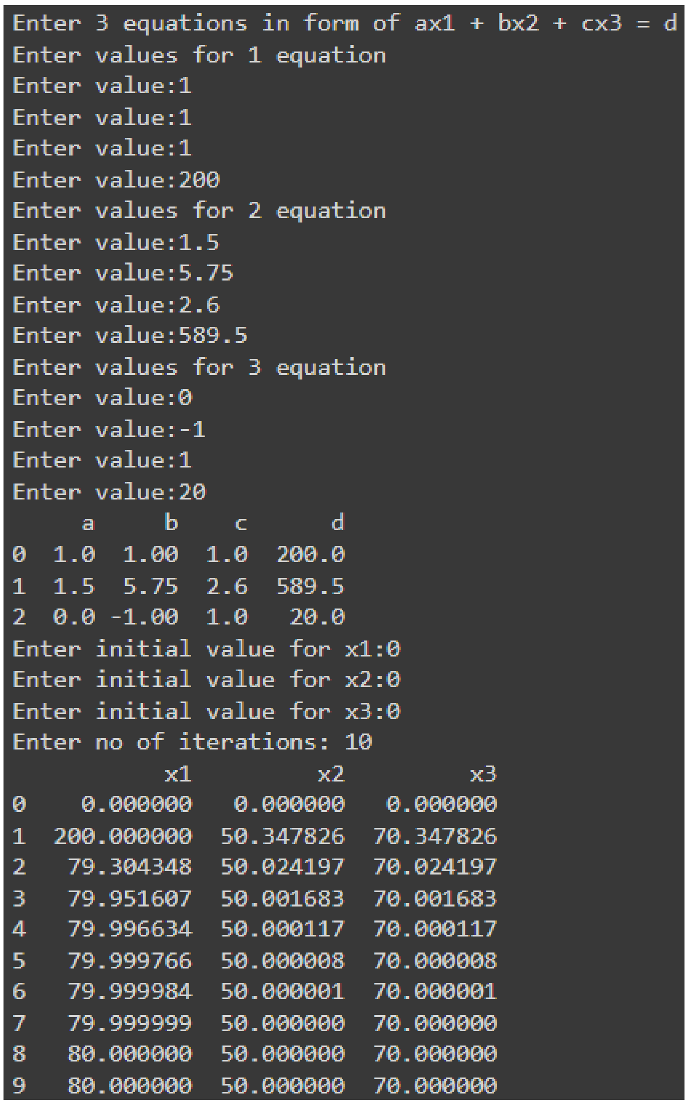
Figure 7: Solving KCL using Gauss Jacobi method

Ques 2: Billy's Restaurant ordered 200 flowers for Mother's Day. They ordered carnations at \$1.50 each, roses at \$5.75 each, and daisies at \$2.60 each. They ordered mostly carnations, and 20 fewer roses than daisies. The total order came to \$589.50. How many of each type of flower was ordered?

Let,
x = no of carnation
y = no of roses
z = no of daisies
ATQ,

x + y + z = 200
1.5x + 5.75y + 2.6z = 589.5
z – 20 = y => -y + z =20

Let initial values x=y=z=0



```
Enter 3 equations in form of ax1 + bx2 + cx3 = d
Enter values for 1 equation
Enter value:1
Enter value:1
Enter value:1
Enter value:200
Enter values for 2 equation
Enter value:1.5
Enter value:5.75
Enter value:2.6
Enter value:589.5
Enter values for 3 equation
Enter value:0
Enter value:-1
Enter value:1
Enter value:20
     a     b    c      d
0  1.0  1.00  1.0  200.0
1  1.5  5.75  2.6  589.5
2  0.0 -1.00  1.0   20.0
Enter initial value for x1:0
Enter initial value for x2:0
Enter initial value for x3:0
Enter no of iterations: 10
           x1         x2         x3
0    0.000000   0.000000   0.000000
1  200.000000  50.347826  70.347826
2   79.304348  50.024197  70.024197
3   79.951607  50.001683  70.001683
4   79.996634  50.000117  70.000117
5   79.999766  50.000008  70.000008
6   79.999984  50.000001  70.000001
7   79.999999  50.000000  70.000000
8   80.000000  50.000000  70.000000
9   80.000000  50.000000  70.000000
```

Figure 8: Finding no of flowers of each category

Carnations = 80, Roses = 50, Daisies =70

References:

1. Code for above problems can be found at: https://colab.research.google.com/drive/12CgB6G8800rXiFYCjNN55u-DeOPk9iUt?usp=sharing (please use BITS email ID only)
2. https://en.wikipedia.org/wiki/Root-finding_algorithms
3. https://byjus.com/maths/bisection-method/
4. https://en.wikipedia.org/wiki/Jacobi_method