Lab 2 Report

Amlan Nayak MS18197

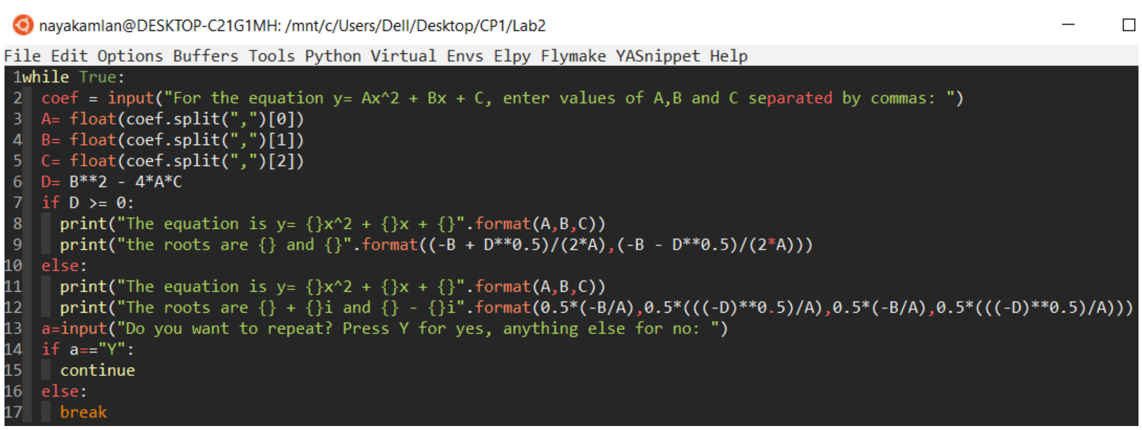
27 Jan 2021

1 A simple code to get the roots of a quadratic equation.

1.1 Algorithm and Discussion

Quadratic roots formula was used directly. When the discriminant was positive, the program gives real roots. When negative, it gives complex roots.

1.2 Code and Output




```python
while True:
    coef = input("For the equation y= Ax^2 + Bx + C, enter values of A,B and C separated by commas: ")
    A= float(coef.split(",")[0])
    B= float(coef.split(",")[1])
    C= float(coef.split(",")[2])
    D= B**2 - 4*A*C
    if D >= 0:
        print("The equation is y= {}x^2 + {}x + {}".format(A,B,C))
        print("the roots are {} and {}".format((-B + D**0.5)/(2*A),(-B - D**0.5)/(2*A)))
    else:
        print("The equation is y= {}x^2 + {}x + {}".format(A,B,C))
        print("The roots are {} + {}i and {} - {}i".format(0.5*(-B/A),0.5*(((-D)**0.5)/A),0.5*(-B/A),0.5*(((-D)**0.5)/A)))
    a=input("Do you want to repeat? Press Y for yes, anything else for no: ")
    if a=="Y":
        continue
    else:
        break
```

Figure 1: Intput Code in Python in emacs

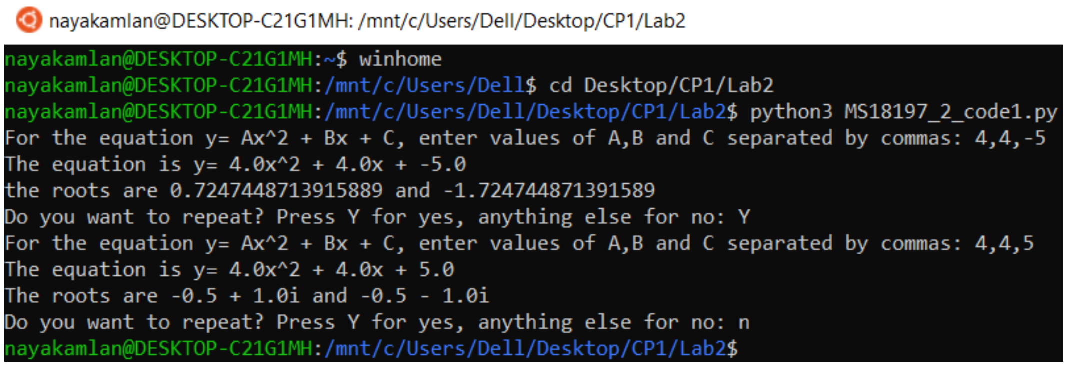


Figure 2: Output

2 Root of equation for $f(x) = x^3 - 3x + 1$ on [0,1] using bisection method.

2.1 Algorithm and Discussion

The upper and lower bounds are given such that there exists a root in between them. The midpoint of both is found. The product of values of function at midpoint and lower bound is found out. If positive, the root does not lie in between them. The midpoint becomes the new lower bound and the whole process described above is repeated.

In the case product of values of function at midpoint and lower bound is negative, the midpoint becomes the new upper bound and the next midpoint is found. The process repeats as described above.

This whole process works inside a while loop which ensures once the values of successive midpoint are sufficiently close enough, the loop terminates.

2.2 Code and Output

```
nayakamlan@DESKTOP-C21G1MH: /mnt/c/Users/Dell/Desktop/CP1/Lab2
File Edit Options Buffers Tools Python Virtual Envs Elpy Flymake YASnippet Help
print("Find root of equation for f(x)=x^3 - 3x + 1 on [0,1] using bisection method")
up=1
low=0
def f(x):
    return x**3 - 3*x + 1
mid= 1
mid_before=10
while abs(mid-mid_before)/abs(mid) > 0.001:
    mid_before=mid
    mid=(low+up)/2
    if f(mid)==0:
        break
    elif f(mid)*f(low)>0:
        low= mid
    else:
        up=mid

print("The value of root is : {} ".format(mid))
```

Figure 3: Python code in emacs

```
nayakamlan@DESKTOP-C21G1MH:/mnt/c/Users/Dell/Desktop/CP1/Lab2$ python3 MS18197_2_code2.py
Find root of equation for f(x)=x^3 - 3x + 1 on [0,1] using bisection method
The value of root is : 0.347412109375
```

Figure 4: Output

3 A program that determines he solution of equation $f(x) = 8 - 4.5(x - sin(x))$ by using bisection method. Solution should have tolerance of less than 0.001 rad.

3.1 Algorithm

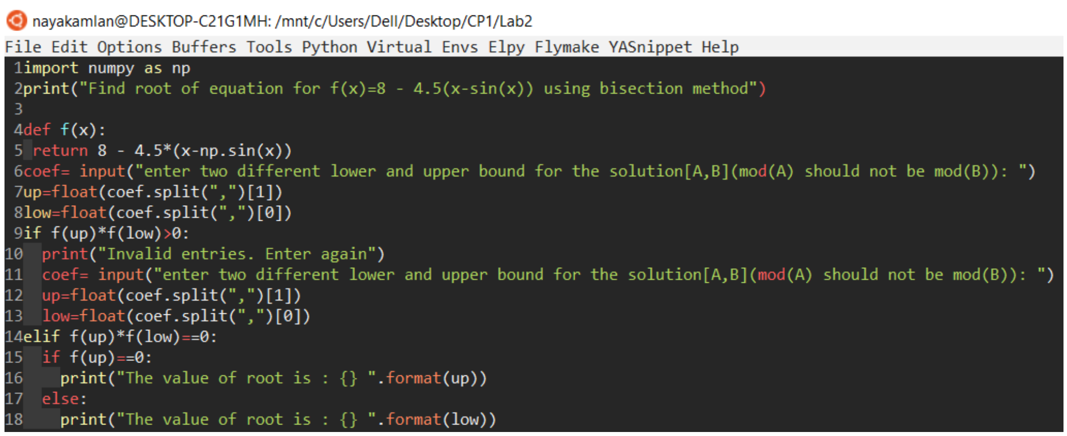
Here, the user is allowed to enter the bounds. If the root does not lie in between the bounds, the code discard the inputs and ask the user to fill the values again.

The upper and lower bounds are chosen such that there exists a root in between them. The midpoint of both is found. The product of values of function at midpoint and lower bound is found out. If positive, the root does not lie in between them. The midpoint becomes the new lower bound and the whole process described above is repeated.

In the case product of values of function at midpoint and lower bound is negative, the midpoint becomes the new upper bound and the next midpoint is found. The process repeats as described above.

This whole process works inside a while loop which ensures once the values of successive midpoint are sufficiently close enough(difference less than 0.001 rads), the loop terminates.

3.2 Code and Output



nayakamlan@DESKTOP-C21G1MH: /mnt/c/Users/Dell/Desktop/CP1/Lab2

File Edit Options Buffers Tools Python Virtual Envs Elpy Flymake YASnippet Help

```
import numpy as np
print("Find root of equation for f(x)=8 - 4.5(x-sin(x)) using bisection method")

def f(x):
 return 8 - 4.5*(x-np.sin(x))
coef= input("enter two different lower and upper bound for the solution[A,B](mod(A) should not be mod(B)): ")
up=float(coef.split(",")[1])
low=float(coef.split(",")[0])
if f(up)*f(low)>0:
  print("Invalid entries. Enter again")
  coef= input("enter two different lower and upper bound for the solution[A,B](mod(A) should not be mod(B)): ")
  up=float(coef.split(",")[1])
  low=float(coef.split(",")[0])
elif f(up)*f(low)==0:
  if f(up)==0:
    print("The value of root is : {} ".format(up))
  else:
    print("The value of root is : {} ".format(low))
```

Figure 5: Code in Python in emacs

```python
mid = 1
mid_before=10

while abs(mid-mid_before)/abs(mid) > 0.001:
        mid_before=mid
        mid = (up+low)/2
        if (f(mid) == 0.0):
                break
        elif (f(mid)*f(low) < 0):
                up = mid
        else:
                low = mid
print("The value of root is : {} ".format(mid))
```

Figure 6: Code in Python in emacs

```
nayakamlan@DESKTOP-C21G1MH:/mnt/c/Users/Dell/Desktop/CP1/Lab2$ python3 MS18197_2_code3.py
Find root of equation for f(x)=8 - 4.5(x-sin(x)) using bisection method
enter the lower and upper bound for the solution[A,B]: 0,5
The value of root is : 2.430419921875
nayakamlan@DESKTOP-C21G1MH:/mnt/c/Users/Dell/Desktop/CP1/Lab2$ python3 MS18197_2_code3.py
Find root of equation for f(x)=8 - 4.5(x-sin(x)) using bisection method
enter the lower and upper bound for the solution[A,B]: -50,51
The value of root is : 2.4310455322265625
```

Figure 7: Output

4 Write code for equation $x^3 - x - e^x - 2 = 0$ having root between [2,3] using Secant method, Regula-Falsi method, Newton-Raphson method and Chebyshev method

4.1 Algorithm

Our starting interval is such that $f(a)f(b) < 0$ where a and b are lower and upper bounds respectively

- Secant Method - We compute $f(c)$ where c is given by:

  $$c = a - f(a)\frac{b-a}{f(b)-f(a)}$$

  This is the secant line. The method described is repeated again till we reach a certain level of accuracy. Finally we get the final intercept of the secant line on the x-axis.

- Regula-Falsi Method - Our starting interval is such that $f(a)f(b) < 0$ where a and b are lower and upper bounds. We define a new variable c which is given:

  $$c = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

  If $f(c)f(a) > 0$, the new bounds become $[c, b]$. If not, the new bounds become $[a, c]$. This process continues for a fixed number of iterations and it finally outputs an approximate root at the end of a loop. And in case during the iterations $f(c) = 0$, c is our approximate root.

- Newton-Raphson Method - In this method, we use the tangent line. If we guess a x_0 which is near the root of $f(x)$, then we can use the tangent line at and compute the x-intercept of the tangent line. The x-intercept x_1 is given by $x_0 - \frac{f(x_0)}{f'(x_0)}$. We then continue this process for a fixed number of iterations using the relation:

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

  The solution eventually converges to the root.

- Chebyshev method - The chebyshev method is very similar to Newton Raphson method. Here the iteration relation is given by:

  $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{1}{2}\frac{f(x_n)^2}{f'(x_n)^3}f''(x_n)$$

4.2 Code and Output

```python
import numpy as np
print("Find root of equation for f(x)=x**3 - x - exp(x) - 2 in the interval [2,3]")
def f(x):
 return x**3 - x - np.exp(x) - 2
def f_1(x):
 return 3*(x**2) - 1 - np.exp(x)
def f_2(x):
 return 6*x - np.exp(x)
```

Figure 8: Code in Python in emacs

```python
def secant():
  xn1 = 2
  xn = 3
  x = xn1
  while(abs(f(x)) > 0.00001):
    x = xn - ((f(xn) * (xn - xn1)) / (f(xn) - f(xn1)))
    xn1 = xn
    xn = x
  print("Root through Secant Method is {}".format(x))
```

Figure 9: Secant method Python code in emacs

```python
def falsi():
  x_0=2
  x_1=3
  x_2= 5
  i=0
  while i < 20:
    x_2 = (x_0 * f(x_1) - x_1 * f(x_0))/ (f(x_1) - f(x_0))
    if f(x_2)==0:
      break
    if f(x_2)*f(x_0)<0:
      x_1=x_2
    else:
      x_0=x_2
    i=i+1
  print("Root through Regula-falsi Method is {}".format(x_2))
```

Figure 10: Regula-Falsi method Python code in emacs

```python
def newton():
    x_0=2
    x_1=3
    if abs(f(x_0))>=abs(f(x_1)):
        x_2=x_1
    else:
        x_2=x_0
    i=0
    while i < 51:
        x_3 = x_2 - f(x_2)/f_1(x_2)
        x_2=x_3
        i=i+1
    print("Root through Newton Raphson Method is {}".format(x_3))
```

Figure 11: Newton Raphson method Python code in emacs

```python
def cheby():
    x_0=2
    x_1=3
    if abs(f(x_0))>=abs(f(x_1)):
        x_2=x_1
    else:
        x_2=x_0
    i=0
    while i < 21:
        x_3 = x_2 - f(x_2)/f_1(x_2) - 0.5*(f(x_2)**2 / f_1(x_2)**3)*f_2(x_2)
        x_2=x_3
        i=i+1
    print("Root through chebyshev Method is {}".format(x_3))
```

Figure 12: Chebyshev method Python code in emacs

```python
while True:
  method= input("type 1 for Secant Method, 2 for Regula-falsi method, 3 for Newton Raphson method and 4 for Chebyshev method: ")
  if method=="1":
    secant()
  elif method=="2":
    falsi()
  elif method=="3":
    newton()
  elif method=="4":
    cheby()
  else:
    print("Invalid Entry")
  a=input("Enter Y for trying again. Anything else to exit: ")
  if a=="Y":
    continue
  else:
    break
```

Figure 13: Code in Python in emacs

```
nayakamlan@DESKTOP-C21G1MH:/mnt/c/Users/Dell/Desktop/CP1/Lab2$ python3 MS18197_2_code4.py
Find root of equation for f(x)=x**3 - x - exp(x) - 2 in the interval [2,3]
type 1 for Secant Method, 2 for Regula-falsi method, 3 for Newton Raphson method and 4 for Chebyshev method: 1
Root through Secant Method is 2.682726514174302
Enter Y for trying again. Anything else to exit: Y
type 1 for Secant Method, 2 for Regula-falsi method, 3 for Newton Raphson method and 4 for Chebyshev method: 2
Root through Regula-falsi Method is 2.682726514174302
Enter Y for trying again. Anything else to exit: Y
type 1 for Secant Method, 2 for Regula-falsi method, 3 for Newton Raphson method and 4 for Chebyshev method: 3
Root through Newton Raphson Method is 2.682726514174302
Enter Y for trying again. Anything else to exit: Y
type 1 for Secant Method, 2 for Regula-falsi method, 3 for Newton Raphson method and 4 for Chebyshev method: 4
Root through chebyshev Method is 2.682726514174302
Enter Y for trying again. Anything else to exit: s
nayakamlan@DESKTOP-C21G1MH:/mnt/c/Users/Dell/Desktop/CP1/Lab2$
```

Figure 14: Output