ni_module Package User Manual

Members:
Columba, Lorenzo Miguel L.
Datay, Danica Mae L.
Embuscado, Khayle Anthony L.
Tagalog, Flor-Ann B.

About

This user manual contains instructions on how to use and implement the package on other programs. ni_module package is consists of modules that will help you find the roots of a given equation by inserting the required parameters in each module.

Step 1 Import the ni_module package

In [1]:
```
import numpy as np
# import the package
import ni_module as nm
```

Step 2 Define a given Equation

In [2]:
```
#equation
f = lambda x: x**3 -6*x**2 - 9*x + 54
```

Step 3 choose a desired method to use for finding

The package contains the modules listed below

- Simple Iteration Method
- Newton Rhapson Method
- Bisection Method
- Regula Falsi Method
- secant method

Step 4 Provide the required parameter in the module choosen

simple Iteration Method

This method require the user to give a equation to solve and a initial guess
f is equal to the defined equation and *-5* is the initial guess input by the user

In [13]:
```
#Simple Iteration Method single root
nm.simp_iter(f,-5)
```

```
-176
-70
0
The root is: [-3], found at epoch 2
```

In [4]:
```
#Simple Iteration Method n root
nm.simp_iter_n(f,-3)
```

```
0
40
56
54
40
20
0
-14
-16
0
The root is: [-3, 3, 6], found at epoch 10
```

Newton Rhapson Method

This method require the user to give a equation to solve, and the number of roots to find *f* is equal to the defined equation,1 and 3 are the number of roots to find

In [15]:
```
# single roots newton method
nm.newton(f,1)
```

```
The roots are [-3.], found at epoch: 4
```

In [16]:
```
# n root newton method
nm.newton(f,3)
```

```
The roots are [-3.  3.  6.], found at epoch: 4
```

Bisection Method

This method require the user to give a equation to solve, 2 initial guesses, and the number of roots to find . f is equal to the defined equation,3 and 5 is the 2 initial guesses input by the user, and 2 are the number of roots to that need find

In [17]:
```
#bisection single
nm.bisec(f,-3,5,1)
```

```
The roots  [3.], found at epoch: 0
```

In [18]:
```
# bisection n roots
nm.bisec( f,-5,5,2)
```

```
The roots  [-3.  3.], found at epoch: 0
```

Regula Falsi (False Position Method)

This method require the user to give a equation to solve, and 2 initial guesses and the number of roots to find . f is equal to the defined equation 0 and 5 is the 2 initial guesses input by the user and 1 and 2 are the number of roots to that need find.

In [19]:
```
# false position method single root
nm.false_pos(f, -5, 5, 1)
```

```
The roots are [-3.], at pos: 99
```

In [20]:
```
# false position method n root
nm.false_pos(f, -5, 5, 2)
```

```
The roots are [-3.  3.], at pos: 99
```

Secant Method

This method require the user to give a equation to solve, and 2 initial guesses, and the number of roots to find. f is equal to the defined equation,1 and 3 is the 2 initial guesses input by the user, 1 and 2 are the number of roots to that need find.

In [11]:
```
# Secant method single root
nm.sec_meth(f,1,3,1)
```

```
The roots  [-3.], found at epoch: 9
```

In [21]:
```
# Secant method n root
nm.sec_meth(f,-5,5,2)
```

```
The roots  [-3.  3.], found at epoch: 7
```