Bisection Method

```
import numpy as np
```

```
def bisection(func,i1,i2,n_roots,epochs,n_move,tol = 1.0e-06):
  x_roots = []
  fpoint = []
  spoint = []

  for i in range(n_move): #Moving intervals
    i1+=0.25
    i2+=0.25
    fpoint.append(i1)
    spoint.append(i2)

  for (i1,i2) in zip (fpoint,spoint):
    y1, y2 = func(i1), func(i2)
    root = None
    end_bisect = 0
    if np.sign(y1) == np.sign(y2):
      pass #Root are not in this interval
    else:
      for bisect in range(epochs):
        midp = np.mean([i1,i2])
        y_mid = func(midp)
        y1 = func(i1)
        if np.allclose(0,y1, tol):
          root = i1
          x_roots.append(root)
          final_roots = np.unique(np.around(x_roots,3))
          final_roots = final_roots[:n_roots]
          end_bisect = bisect
          break
        if np.sign(y1) != np.sign(y_mid): #root is in first-half interval
          i2 = midp
        else: #root is in second-half interval
          i1 = midp

  return final_roots,end_bisect

func = lambda x: 2*x**4 + 3*x**3 - 11*x**2 - 9*x + 15
root,end_bisect = bisection(func,i1 = -10,i2 = 1,n_roots = 5,epochs = 100,n_move = 50,tol = 1
print("The roots are {}, at bisection: {} ".format(root,end_bisect))
```

```
The roots are [-2.5   -1.732  1.     1.732], at bisection: 0
```

Regula Falsi Method

```
def falsi(func,a,b,n_roots,epochs,n_move,tol = 1.0e-06):
  x_roots = []
  fpoint = []
  spoint = []

  for i in range(n_move):
    a+=0.25
    b+=0.25
    fpoint.append(a)
    spoint.append(b)
  for (a,b) in zip (fpoint,spoint):
    y1, y2 = func(a), func(b)
    root = None
    pos = 0
    if np.allclose(0,y1): root = a
    elif np.allclose(0,y2): root = b
    elif np.sign(y1) == np.sign(y2):
      print("Root are not in this interval")
    else:
      for pos in range(epochs):
        c = b - (func(b)*(b-a))/(func(b)-func(a)) ##false root
        if np.allclose(0,func(c), tol):
          root = c
          x_roots.append(root)
          final_roots = np.unique(np.around(x_roots,3))
          final_roots = final_roots[:n_roots]
          break
        if np.sign(func(a)) != np.sign(func(c)): b,y2 = c,func(c)
        else: a,y1 = c,func(c)
  return final_roots, pos

func = lambda x: x**2 + np.cos(x)**2-4*x
root,pos = falsi(func,a = -1,b = 0,n_roots = 10,epochs = 100,n_move = 35,tol = 1.0e-06)
print("The roots are {}, at pos: {} ".format(root,pos))
```

```
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
```

```
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
Root are not in this interval
The roots are [0.25 3.85], at pos: 0
```

Secant Method

```
def secant(func,a,b,n_roots,epochs,n_move,tol = 1.0e-06):
  x_roots = []
  fpoint = []
  spoint = []

  for i in range(n_move):
    a+=0.25
    b+=0.25
    fpoint.append(a)
    spoint.append(b)

  for (a,b) in zip (fpoint,spoint):
    root = None
    end_epoch =  0
    for epoch in range(epochs):
      c = b - (func(b)*(b-a))/(func(b)-func(a))
      if np.allclose(b,c):
        root = c
        x_roots.append(root)
        final_roots = np.unique(np.around(x_roots,3))
        final_roots = final_roots[:n_roots]
        end_epoch = epoch
        break
      else:
        a,b = b,c

  return final_roots, end_epoch

func = lambda x: np.log(x**2-2*x-1)*(x**2-3)
root,end_epoch = secant(func,a = -3,b = 1,n_roots = 4,epochs = 100,n_move = 23,tol = 1.0e-06)
print("The roots are {}, found at epoch: {} ".format(root,end_epoch))
```

```
The roots are [-1.732 -0.732  2.732], found at epoch: 3
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:29: RuntimeWarning: invalid
```