Python to find roots of an equation using Secant Method

Numerical Methods

Python to find roots of an equation using Secant Method

In this, you will learn how to find the roots of equations using Secant Method in Python Programming.

Secant Method

The second method is used to find the origin of the equation
F (x) = 0. It starts from two different estimates, x1 and x2 for the root. It is a repetition process with linear interpolation to a source.

The iteration stops if the difference between the two intermediate values is less than the convergence factor.

Steps:

1. x1, x2, E, n // E = convergence indicator

2. Calculate f (x1), f (x2)

3. if (f (x1) * f (x2) = E); // Repeat the loop until convergence

4. Print the value of 'x0' // value of the root
5. Print the 'n' // iterations
else
6. Print "Source not found"

In this Python program, x0 & x1 are the two initial estimation values, e is the tolerable error and f (x) is the actual non-linear function whose root is obtained using the second method. Variable x2 has approximately root in each step.

Example 1: Program to find the solution of equation x2-9 using secant method

Python Code:

```
from pylab import *
def secant(f,x0,x1,eps):
    f_x0=f(x0)
    f_x1=f(x1)
    iteration_counter=0
    while abs(f_x1)>eps and iteration_counter<100:
        try:
            x2 = float(f_x1-f_x0)/(x1-x0)
            x = x1-float(f_x1)/x2
        except:
            print("error")
        x0 = x1
        x1 = x
        f_x0=f_x1
        f_x1=f(x1)
        iteration_counter+=1
    if abs(f_x1)>eps:
        iteration_counter-=1
    return x, iteration_counter
def f(x):
    return x**2-9
x0 = 10
x1 = x0 - 1
solution, no_iterations=secant(f,x0,x1,eps=0.00001)
if no_iterations>0:
    print("number of function call: %d" %(2+no_iterations))
    print("solution is:", solution)
else:
    print("solution is not found")
```

Output:

```
number of function call: 9
solution is: 3.000000000915234
```

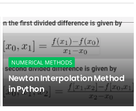
Example 2: Program to find the solution of equation x3-4 using secant method

Python Code:

```
from pylab import *
def secant(f,x0,x1,eps):
    f_x0=f(x0)
    f_x1=f(x1)
    iteration_counter=0
    while abs(f_x1)>eps and iteration_counter<100:
        try:
            x2 = float(f_x1-f_x0)/(x1-x0)
            x = x1-float(f_x1)/x2
        except:
            print("error")
        x0 = x1
        x1 = x
        f_x0=f_x1
        f_x1=f(x1)
        iteration_counter+=1
    if abs(f_x1)>eps:
        iteration_counter-=1
    return x, iteration_counter
def f(x):
    return x**3-4
x0 = 10
x1 = x0 - 1
solution, no_iterations=secant(f,x0,x1,eps=0.00001)
if no_iterations>0:
    print("number of function call: %d" %(2+no_iterations))
    print("solution is:", solution)
else:
    print("solution is not found")
```

Output:

```
number of function call: 13
solution is: 1.5874010554986935
```

PREVIOUS
Regula Falsi Method in Python

NEXT
Jacobi Method in Python and Numpy