Secant Method

The secant method may seem similar to false position method but instead of looking for two point embracing the root, the roots of secant lines are found consecutively until the approximate root of the given function is approached. Consequently, the two initial values of x do not have to be on both sides of the expected root.

The figure illustrates the method by using the first three successive roots x_3, x_4 and x_5 of the secant lines AB, BC and DE, respectively.
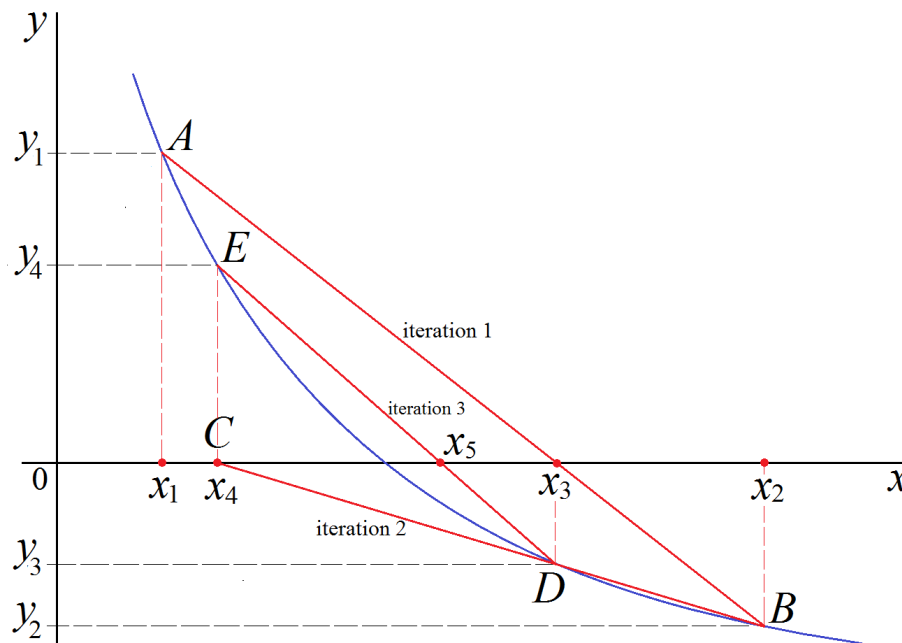

Figure The first three iterations of secant method

The equation of the first three iterations are

$$x_3 = x_2 - \frac{x_2 - x_1}{y_2 - y_1} y_2$$

$$x_4 = x_3 - \frac{x_3 - x_2}{y_3 - y_2} y_3$$

$$x_5 = x_4 - \frac{x_4 - x_3}{y_4 - y_3} y_4$$

This sequence will continue until the condition of convergence is satisfied at x_i where

$$x_i = x_{i-1} - \frac{x_{i-1} - x_{i-2}}{y_{i-1} - y_{i-2}} y_{i-1}$$

Although this method requires two initial values to start iterations, it utilizes the algorithm of the iteration methods.

Example 4.6

Find the roots of the following equation:

$$2x^2 - 5x + 3 = 0$$

Solution

Let's write the equation as a function

$$y(x) = 2x^2 - 5x + 3$$

The function of the method can be written as following

```python
def secant(fn, x = list(), tol=1.0E-9, maxiter = 100):
    [x1,x2] = x
    for iteration in range(maxiter):
        xnew = x2 - (x2 - x1)/(fn(x2) - fn(x1)) * fn(x2)
        if abs(xnew - x2) < tol: break
        x1 = x2
        x2 = xnew
    else:           # this else belongs to the for-loop
        print('Limit of iterations reached without solution.')
    return xnew, iteration


# definition of the function
f = lambda x: 2*x**2 - 5*x + 3
# call of the solver
soln, iters = secant(f, [0,.8], 0.000001)
# output of the results
print('The root : %0.6f' % soln)
print('The number of iterations : %d' % iters)
```

x1 and x2 are included in a Python list

The output is

```
The root : 1.000000
The number of iterations : 7
```

The second root can be found if the initial value array is entered as [2, 3]:

```
The root : 1.500000
The number of iterations : 8
```

In this code, values of x1 and x2 are sent to the function as a Python list. This is a common method in numerical computing functions to pass arguments of multiple components as a Python list or a NumPy array.