**Root finding**

This article will address how to solve a non-linear equation in python. In python, there are a lot of methods available to solve non-linear equations. Here we are using scipy.fsolve to solve a non-linear equation. There are two types of equations available, Linear and Non-linear.

**Fsolve**

It is a function in a scipy module that returns the roots of non-linear equations.

**Syntax**

scipy.optimize.fsolve (func, x0, args=(), fprime=None, full\_output=0, col\_deriv=0, xtol=1.49012e-08, maxfev=0, band=None, epsfcn=None, factor=100, diag=None)

Parameters

func: It is a function that takes an argument and returns the value.

x0: ndarray, It is a starting estimate for the root of fun(x)=0.

args: Tuple, it is an extra argument to the function, optional.

fprime: It is a function to compute the Jacobian of function with derivatives.

full\_output: It is a boolean value and it is optional.

col\_deriv: It is a boolean value and it is also optional. Specify whether the Jacobian function computes derivatives down the columns

xtol: It is float, optional.

maxtev: It is int and the maximum number of calls to the function, optional.

band: It is a tuple and optional parameter. If set to a two-sequence containing the number of sub- and super-diagonals within the band of the Jacobi matrix, the Jacobi matrix is considered banded

epsfcn: It is float and optional.

factor: It is float and it determines the initial step bounds.

diag: It is a scale factor for the variables.

**Returns**

x: ndarray, It is a solution.

infodict: It is a dictionary of optional values with the keys.

nfev number of function calls

njev number of Jacobian calls

fvec function evaluated at the output

fjac the orthogonal matrix

r upper triangle matrix

qtf the vector

ier: An integer flag.

msg: If no solution is found, mesg details the cause of failure.