



Lemon

Lemon Package Documentation

- [Methods](#)

MEMBERS

L.A. Guy © 2021

R. Surio Jr. © 2021

M.A. Sustento © 2021

T.J. Vallarta © 2021



Lemon

METHODS

- [B_force\(\)](#)
- [B_forcex\(\)](#)
- [Newton\(\)](#)
- [Bisection\(\)](#)
- [Falsi\(\)](#)
- [Secant\(\)](#)



Lemon

`lemon.b_force(func, n_roots, epochs, x0, x1, tol=1e-5)`

Definition: Returns the roots of an equation given in *func* using the brute force method (simple iterative method).

Parameters: **func:** *array_like*

Input arrays with equation.

n_roots: *int*

The desired number of roots.

epochs: *int*

The desired number of iterations.

x0: *int*

Starting value.

x1: *int*

Starting value.

tol: *float*

Tolerance.

Returns: **Roots:** *List*

A list containing the roots of the equation.

Epochs: *int*

An integer containing the number of iterations where the root is found.

Examples:

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from lemon_package import BruteForceIteration as BFI
>>> func = lambda x: x**2 - 1
>>> roots,end_epoch = BFI.b_force(func,no_roots = 4,epochs = 100000,x0 = -10,x1 = 10)

>>> print("The roots are",roots, "found at", end_epoch)
The roots are [-1. 1.] found at 90000
>>> |
```



Lemon

`lemon.b_forcex(func, n_roots, epochs, x0 = 0)`

Definition: Returns the roots of an equation given in *func* using the brute force method (in terms of *x*).

Parameters: **func:** *array_like*

Input arrays with equation.

n_roots: *int*

The desired number of roots.

epochs: *int*

The desired number of iterations.

x0: *int*

Starting value.

Returns: **Roots:** *List*

A list containing the roots of the equation.

Epochs: *int*

An integer containing the number of iterations where the root is found.

Examples:

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from lemon_package import BruteForceIntermsX as BFIX
>>> sample1 = lambda x: 2*x**4 + 3*x**3 - 11*x**2 - 9*x + 15
>>> f1 = lambda x: (2*x**4 + 3*x**3 - 11*x**2 + 15)/9
>>> f2 = lambda x: ((2*x**4 + 3*x**3 - 9*x + 15)/11)**(1/2)
>>> f3 = lambda x: ((-2*x**4 + 11*x**2 + 9*x - 15)/3)**(1/3)
>>> f4 = lambda x: ((-3*x**3 + 11*x**2 + 9*x - 15)/2)**(1/4)
>>> func = [f1,f2,f3,f4]
>>> roots,end_epoch = BFIX.b_forcex(func,no_roots = len(func),epochs = 100)
>>> print("The roots are",roots, "found at", end_epoch)
The roots are [1.  +0.j 1.732+0.j] found at 16
```



Lemon

`lemon.newton(func, n_roots, epochs, tol = 1.0e-05, inits)`

Definition: Returns the roots of an equation given in *func* using the newton-raphson method.

Parameters: **func:** *array_like*

Input arrays with equation.

n_roots: *int*

The desired number of roots.

epochs: *int*

The desired number of iterations.

tol: *float*

Tolerance.

inits: *int*

Return values within a given interval.

Returns: **Roots:** *List*

A list containing the roots of the equation.

Epochs: *int*

An integer containing the number of iterations where the root is found.

Raises: **RuntimeWarning**

When encountered invalid value in log or double_scalars.

Examples:

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from lemon_package import Newton as nwtn
>>> import numpy as np
>>> func = lambda x: np.sin(2*x)-np.cos(2*x)
>>> roots,epoch = nwtn.newton(func,n_roots = 5,epochs = 100, tol = 1.0e-05,inits = np.arange(0,6.28))
>>> print("The roots are {}, found at epoch: {}".format(roots,epoch))
The roots are [0.393 1.963 3.534 5.105 8.247], found at epoch: 2
```



Lemon

lemon.bisection(func, i1, i2, n_roots, epochs, n_move, tol = 1.0e-06)

Definition: Returns the roots of an equation given in *func* using the bisection method.

Parameters: **func:** *array_like*

Input arrays with equation.

i1: *int*

Intervals where the root is expected.

i2: *int*

Intervals where the root is expected.

n_roots: *int*

The desired number of roots.

epochs: *int*

The desired number of iterations.

n_move: *int*

Increments or decrements of moving interval.

tol: *float*

Tolerance.

Returns: **Roots:** *List*

A list containing the roots of the equation.

Epochs: *int*

An integer containing the number of iterations where the root is found.

Raises: **RuntimeWarning**

When encountered invalid value in log and divide by zero in log.

Examples:

```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from lemon_package import Bisection as bscn
>>> func = lambda x: 2*x**4 + 3*x**3 - 11*x**2 - 9*x + 15
>>> root,end_bisect = bscn.bisection(func,i1 = -10,i2 = 1,n_roots = 5,epochs = 100,n_move = 50,tol = 1.0e-06,)
>>> print("The roots are {}, at bisection: {}".format(root,end_bisect))
The roots are [-2.5 -1.732 1. 1.732], at bisection: 0
```



Lemon

`lemon.falsi(func, a, b, n_roots, epochs, n_move, tol = 1.0e-06,)`

Definition: Returns the roots of an equation given in *func* using the regula falsi (false position) method.

Parameters: **func:** *array_like*

Input arrays with equation.

a: *int*

Intervals where the root is expected.

b: *int*

Intervals where the root is expected.

n_roots: *int*

The desired number of roots.

epochs: *int*

The desired number of iterations.

tol: *float*

Tolerance.

n_move: *int*

Increments or decrements of moving interval.

Returns: **Roots:** *List*

A list containing the roots of the equation.

Epochs: *int*

An integer containing the number of iterations where the root is found.

Raises: **RuntimeWarning**

When encountered invalid value in log.

Examples:

```
Python 3.7.9 Shell
File Edit Shell Debug Options Window Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from lemon_package import Falsi as fls
>>> import numpy as np
>>> func = lambda x: x**2 + np.cos(x)**2-4*x
>>> root,pos = fls.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))
The roots are [0.25 3.85], at pos: 0
```



Lemon

`lemon.secant(func, a, b, n_roots, epochs, n_move, tol = 1.0e-06)`

Definition: Returns the roots of an equation given in *func* using the secant method.

Parameters: **func:** *array_like*

Input arrays with equation.

a: *int*

Intervals where the root is expected.

b: *int*

Intervals where the root is expected.

n_roots: *int*

The desired number of roots.

epochs: *int*

The desired number of iterations.

n_move: *int*

Increments or decrements of moving interval.

tol: *float*

Tolerance.

Returns: **Roots:** *List*

A list containing the roots of the equation.

Epochs: *int*

An integer containing the number of iterations where the root is found.

Raises: **RuntimeWarning**

When encountered invalid value in log.

Examples:

```
Python 3.7.9 Shell
File Edit Shell Debug Options Window Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from lemon_package import Secant as sct
>>> import numpy as np
>>> func = lambda x: 2*x**4 + 3*x**3 - 11*x**2 - 9*x + 15
>>> root,end_epoch = sct.secant(func,a = -10,b = 1,n_roots = 4,epochs = 100,n_move = 40,tol = 1.0e-06)
>>> print("The roots are {}, found at epoch: {}".format(root,end_epoch))
The roots are [-2.5 -1.732 1. 1.732], found at epoch: 8
```