Programming for Essential Digital Skills, Part 2

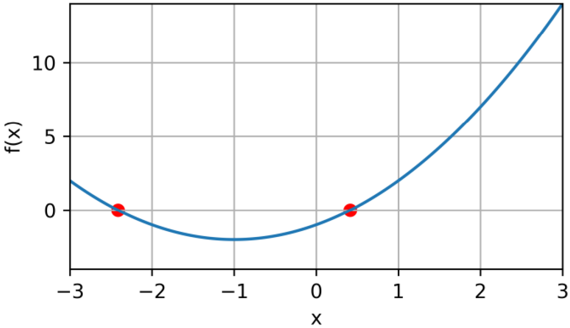
Pieter Kleer

2024

Chapter 8

Root finding

Consider $f(x) = x^2 + 2x - 1$. A root x of the function f is a point that satisfies $f(x) = 0$.



SciPy: Scientific Computing with Python

Subpackages

SciPy is organized into subpackages covering different scientific computing domains. These are summarized in the following table:

Subpackage	Description
`cluster`	Clustering algorithms
`constants`	Physical and mathematical constants
`fft`	Discrete Fourier transforms
`fftpack`	Fast Fourier Transform routines (legacy)
`integrate`	Integration and ordinary differential equation solvers
`interpolate`	Interpolation and smoothing splines
`io`	Input and Output
`linalg`	Linear algebra
`ndimage`	N-dimensional image processing
`odr`	Orthogonal distance regression
`optimize`	Optimization and root-finding routines

Solving the equation $f(x) = 0$ using `fsolve()`

Solving the equation $f(x) = 0$ using `fsolve()`

```
import scipy.optimize as optimize

def f(x):
    return x**2 + 2*x - 1

guess = 3
f_zero = optimize.fsolve(f,guess)[0]

print("A root of the function f is given by", f_zero)
```

```
A root of the function f is given by 0.41421356237309503
```

Solving the equation $f(x) = 3$

Suppose we want to solve $f(x) = 3$. How to do this with root finding?

Solving the equation $f(x) = 3$

Suppose we want to solve $f(x) = 3$. How to do this with root finding?

- If we define $g(x) = f(x) - 3$, then $g(x) = 0$ if and only if $f(x) = 3$.

Solving the equation $f(x) = 3$

Suppose we want to solve $f(x) = 3$. How to do this with root finding?

- If we define $g(x) = f(x) - 3$, then $g(x) = 0$ if and only if $f(x) = 3$.

```
def g(x):
    return f(x) - 3

guess = 4
f_zero = optimize.fsolve(g,guess)[0]

print("A number x satisfying f(x) = 3, is given by", f_zero)
```

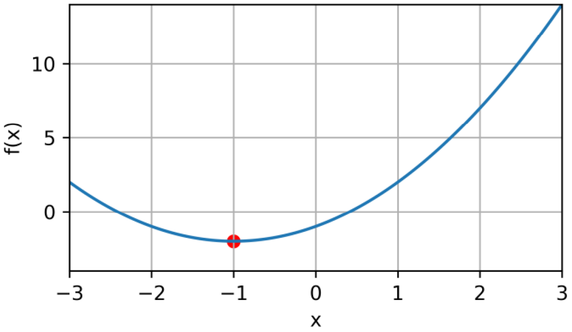
```
A number x satisfying f(x) = 3, is given by 1.2360679774998171
```

Solving the equation $f(x) = c$

```python
def solve_eq(f,c,guess):
    # This function returns the solution to f(x) = c using
    # fsolve() on the function g(x) = f(x) - c

    def g(x):
        return f(x) - c

    x = optimize.fsolve(g,guess)[0]
    return x
```

Minimizing a function f

Consider $f(x) = x^2 + 2x - 1$. Minimum of f is a point x for which $f(x)$ is smallest.



Computing a minimum of f using `fmin()`

```
import scipy.optimize as optimize

def f(x):
    return x**2 + 2*x - 1

guess = 1
minimum = optimize.fmin(f,guess)
```

```
Optimization terminated successfully.
         Current function value: -2.000000
         Iterations: 19
         Function evaluations: 38
```

```
print('The minimum of the function f is attained at x = ', minimum)
```

```
The minimum of the function f is attained at x =  [-1.]
```

Computing a minimum of f using `fmin()`

```
import scipy.optimize as optimize

def f(x):
    return x**2 + 2*x - 1

guess = 1
minimum = optimize.fmin(f,guess,disp=False)[0]

print('The minimum of the function f is attained at x = ', minimum)
```

```
The minimum of the function f is attained at x =  -1.00000000000018
```

Computing a minimum of f using `fmin()`

```
import scipy.optimize as optimize

def f(x):
    return x**2 + 2*x - 1

guess = 1
minimum = optimize.fmin(f,guess,disp=False)[0]

print('The minimum of the function f is attained at x = ', minimum)
```

```
The minimum of the function f is attained at x =  -1.0000000000000018
```

Note: `fmin()` might return a “local” minimum, which is not the true minimum of the function (Classroom Exercise 1).

Matplotlib: Data visualization

Matplotlib is a package that can be used for data visualization

- For this we use the `matplotlib.pyplot` (sub)package ...
- ... which we usually import under the name `plt`

How are functions plotted in Python?

1. Create a vector of x-values, e.g.,

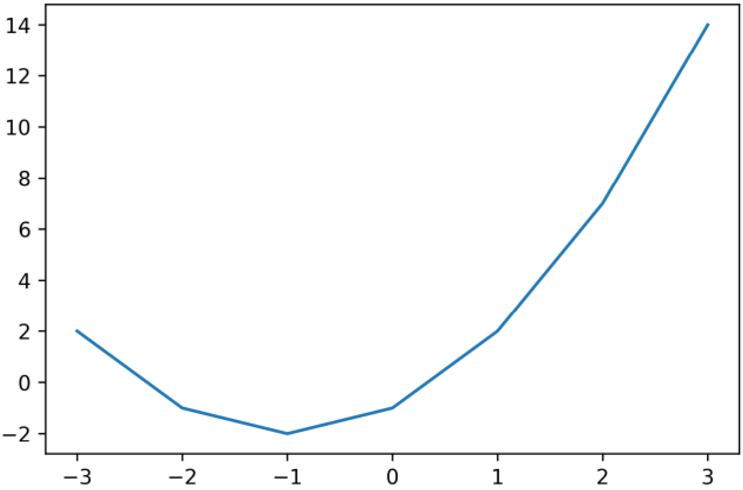
$$x = [-3, -, 2, -1, 0, 1, 2, 3].$$

2. Compute the function values

$$[f(-3), f(-2), f(-1), f(0), f(1), f(2), f(3)] = [2, -1, -2, -1, 2, 7, 14].$$

3. Draw the points $(x_i, f(x_i))$ and connect them with line segments.

Resulting Python plot



Plotting a “smooth” line

Increase the number of points in x to get a smoother line using `np.linspace()`.

- Command `np.linspace(a,b,k)` plots k evenly spaced points in interval $[a, b]$.

Plotting a "smooth" line

Increase the number of points in x to get a smoother line using `np.linspace()`.

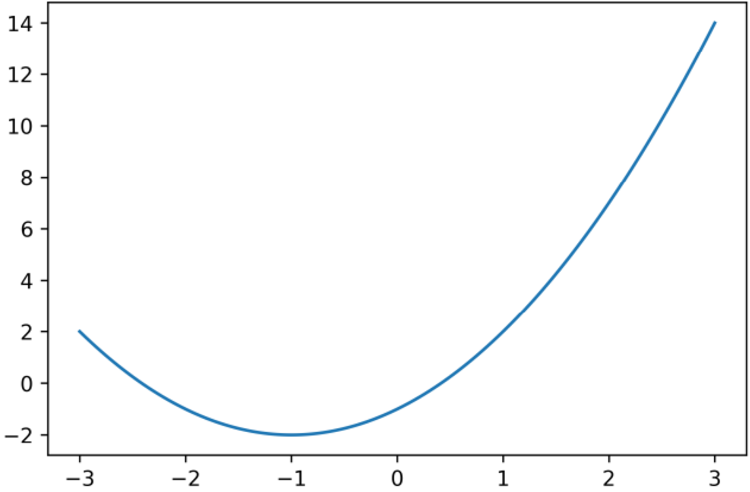
- Command `np.linspace(a,b,k)` plots k evenly spaced points in interval $[a, b]$.

```
import numpy as np

a = 0
b = 1
k = 11

x = np.linspace(a,b,k)
print(x)
```

```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ]
```

Resulting “smoothed” Python plot

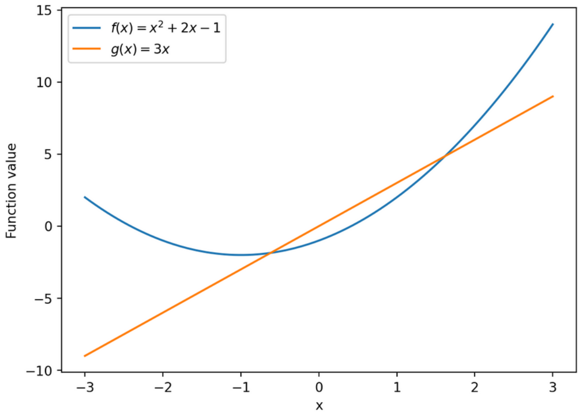
Using `x = np.linspace(-3,3,600)`



Adding legend to plot

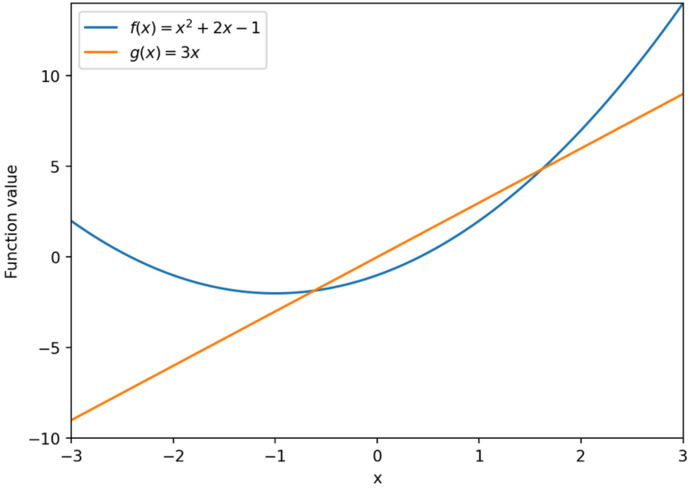
Use label-argument in `plt.plot()` in combination with `plt.legend()` at the end ...

- ... and `plt.xlabel('x')` and `plt.ylabel('Function value')` for axis labels.



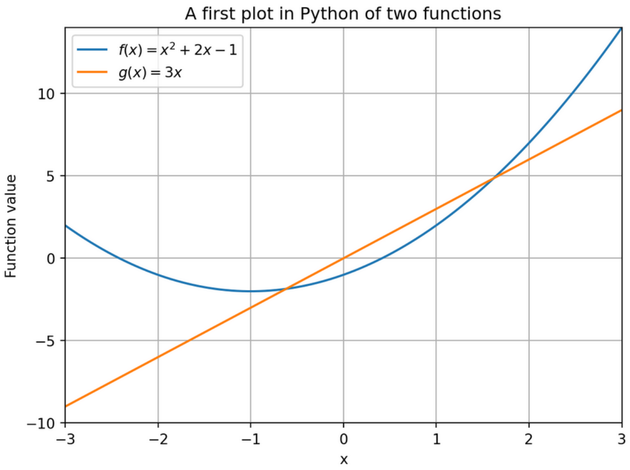
Fixing axes ranges

Use `plt.xlim(-3,3)` and `plt.ylim(-10,14)` to fix range of horizontal/vertical axis, resp.



Adding title and grid

- Use `plt.title('A first plot of two functions')` to add title
- Use `plt.grid()` to add grid.



Classroom Exercise 1

Consider the function $f(x) = \frac{9}{10}x^4 - 3x^3 - \frac{7}{2}x^2 + 12x + 3$.

- Plot this function with horizontal axis range $[-6, 6]$, and vertical axis range $[-15, 15]$.
- Find four roots of this function with `fsolve()` by trying out different initial guesses.
- Find a minimum of this function with `fmin()` by using initial guesses -1 and 2. Are both solutions actual minima of the function?

Chapter 7 (remainder)

Matrix Multiplication

- Suppose we want to calculate $C = AB$, where:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 & 2 \\ 3 & 2 & 1 \\ 1 & 3 & 1 \end{pmatrix}$$

Matrix Multiplication

- Suppose we want to calculate $C = AB$, where:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 3 \end{pmatrix} \qquad B = \begin{pmatrix} 2 & 1 & 2 \\ 3 & 2 & 1 \\ 1 & 3 & 1 \end{pmatrix}$$

- Row i and column j of C is given by:

$$c_{ij} = \sum_{k=1}^{3} a_{ik} b_{kj}$$

For example:

$$c_{21} = \sum_{k=1}^{3} a_{2k} b_{k1} = a_{21} b_{11} + a_{22} b_{22} + a_{23} b_{33} = 2 \times 2 + 3 \times 3 + 1 \times 1 = 14$$

Defining Matrices with Lists:

- We can define matrices as nested lists.
- Each element of the list is a list which represents a row of the matrix.

```
A = [
    [1, 2, 3],
    [2, 3, 1],
    [3, 1, 3]
]

B = [
    [2, 1, 2],
    [3, 2, 1],
    [1, 3, 1]
]
```

Matrix Multiplication without Numpy

```
C = [
    [0, 0, 0],
    [0, 0, 0],
    [0, 0, 0]
]
for i in range(3):
    for j in range(3):
        for k in range(3):
            C[i][j] += A[i][k] * B[k][j]
for row in C:
    print(row)
```

```
[11, 14, 7]
[14, 11, 8]
[12, 14, 10]
```

Matrix Multiplication with Numpy

```python
import numpy as np
A = np.array(A)
B = np.array(B)
np.dot(A, B)
```

```
array([[11, 14,  7],
       [14, 11,  8],
       [12, 14, 10]])
```