Building and Solving Linear Equations

In this lesson, you will learn how to build and solve a system of linear equations in Python.

WE'LL COVER THE FOLLOWING ^

- Building linear equations
- Solving the equations

The linear algebra module of NumPy, linalg, provides a number of functionalities, including the ability to solve systems of linear equations. This is done using the matrix method.

Suppose we have the following system of linear equations:

$$2x + 3y + z = 17$$

 $3x + 4y + 2z = 25$
 $x + y - z = 6$

We need to convert these to the form AX = b and solve the equation for X.

$$\begin{bmatrix} 2 & 3 & 1 \\ 3 & 4 & 2 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 17 \\ 25 \\ 6 \end{bmatrix}$$

Building linear equations

Before we can solve the system of linear equations, it is important to build these equations in a format that Python understands, by using any one of the array creation methods. Let's see an example of this:

```
print(A)
print(b)
```

Solving the equations

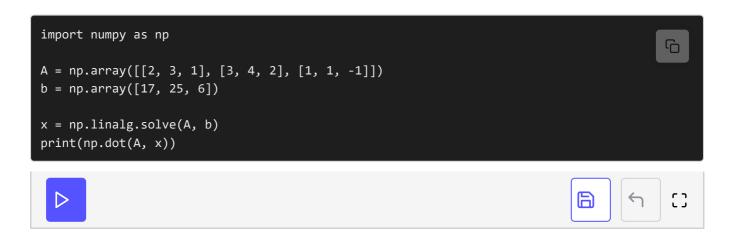
The system may be solved with the solve method, which is part of the linalg subpackage of numpy. The solve method takes, as input, a two-dimensional array (the matrix) and a one-dimensional array (the right-hand side) and returns the solution.

```
import numpy as np

A = np.array([[2, 3, 1], [3, 4, 2], [1, 1, -1]])
b = np.array([17, 25, 6])

x = np.linalg.solve(A, b)
print(x)
```

To check whether the solution is correct, we need to multiply the matrix stored in the array A with the solution we are checking, which is x. We can use the np.dot() method to perform matrix multiplication on its input arguments. The order of input arguments does not matter as long as the matrices are compatible.



Since the result equals matrix **b**, the solution is correct

The next lesson discusses how to find eigenvectors and eigenvalues of a matrix.