Sparse Matrices

In this lesson, we will learn about different sparse matrices in Python and the conversion between sparse and dense matrices.

WE'LL COVER THE FOLLOWING

Sparse matrices in Python

The procedure we have used so far to construct the matrix for a is not very efficient. A full matrix is created, which consists mostly of zeros with non-zero values only appearing on diagonals. There are more efficient routines that store what are called **sparse matrices**. In a sparse matrix, only the value and location of non-zero values in a matrix are stored. Functionality for sparse matrices is available in the `scipy` submodule `sparse`.

> 💡 We will import the `scipy.sparse` submodule as `sp`.

```
import scipy.sparse as sp
```

Dense to sparse

When we create a sparse matrix, we have to choose which format it should be stored in. There are many ways to store a sparse matrix. Two most commonly used are:

1. *compressed sparse column* using the `csc_matrix(A)` method.

2. *compressed sparse row* using the `csr_matrix(A)` method.

The following matrix is an example of this:

$$\begin{bmatrix} 2 & 3 & 0 & 0 \\ 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 \end{bmatrix}$$

Let's convert this dense matrix to a sparse matrix. In the first example, we will convert it into a compressed sparse column:

```
import scipy.sparse as sp
import numpy as np

M = np.array([[2, 3, 0, 0], [1, 2, 3, 0], [0, 1, 2, 3], [0, 0, 1, 2]])
print(M)

sparse_column = sp.csc_matrix(M)
print("Compressed Sparse Column")
print(sparse_column)
```

In line 7, the sparse matrix is stored *column-wise*.

Now let's convert it into a compressed sparse row:

```
import scipy.sparse as sp
import numpy as np

M = np.array([[2, 3, 0, 0], [1, 2, 3, 0], [0, 1, 2, 3], [0, 0, 1, 2]])
print(M)

sparse_row = sp.csr_matrix(M)
print("Compressed Sparse Row")
print(sparse_row)
```

In line 7, the sparse matrix is stored *row-wise*.

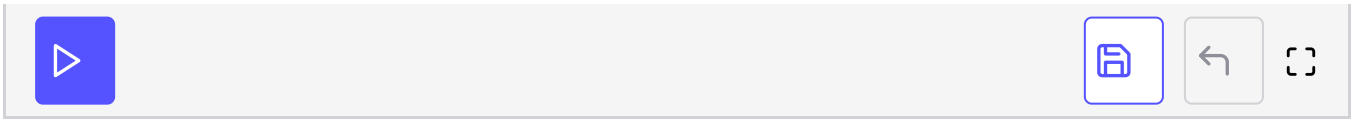
Sparse to dense

`todense()` from the `scipy.sparse` submodule converts a sparse matrix to a dense matrix irrespective of its type. Let's see an implementation of this below:

```
import scipy.sparse as sp
import numpy as np

M = np.array([[2, 3, 0, 0], [1, 2, 3, 0], [0, 1, 2, 3], [0, 0, 1, 2]])
print(M)

sparse_row = sp.csr_matrix(M)
print("Compressed Sparse Row")
print(sparse_row)
print("Dense Matrix")
print(sparse_row.todense())
```



Solving equations with sparse matrices

The solution to the system of equations $Ax = b$ is obtained with the `spsolve()` function of the `scipy.sparse.linalg` module.

> 💡 **Note:** The multiplication sign does not do term-by-term multiplication for sparse matrix objects.

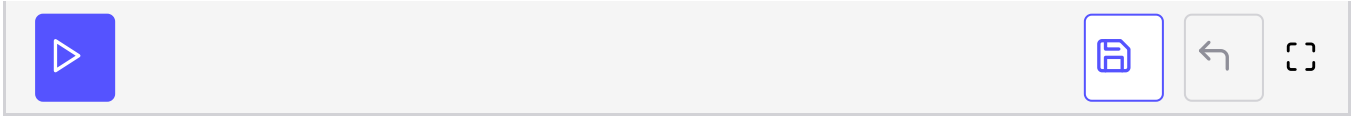
```
import numpy as np
import scipy.sparse as sp
from scipy.sparse.linalg import spsolve

M = np.array([[2, 3, 1], [3, 4, 2], [1, 1, -1]])
print(M)
A = sp.csc_matrix(M)    # conversion to sparse matrix
print(A)

b = np.array([17, 25, 6])
print(b)

x = spsolve(A, b)       # solving the system of linear equation
print(x)

print(A * x)        # checking the solution
```



> 💡 All the computations that we performed on dense matrices, can be performed on sparse matrices as well.

Let's test your knowledge of this chapter with a quiz.