NSC1002 Mathematics and Computing: Integrative Tools for Natural Sciences

Beth Wingate
University of Exeter

January 2021

By the end of this workshop you will…

- Have experience performing matrix operations.
- Be able to perform matrix operations using Python's modules.
- Know how to work with complex numbers in Python
- Explore unit tests

Vector and matrices

Remember:

- Vector with n components:

$$\mathbf{v} = \left[V_1, V_2, \ldots, V_n \right]$$

- Matrix of m x n dimensions:
  n is the number of columns
  m is the number of rows

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

a_{ij} is the ij^{th} element of the matrix A.

A simple vector operation: dot product

- The dot product of two vectors of the same length n is given by

$$\mathbf{u} \cdot \mathbf{w} = \sum_{1}^{n} u_i w_i$$

How to program this into Python:

```
v = [1, 2, 3]           # Initialize v
w = [1, 1/2, 1/3]        # Initialize w
N = len(v)               # Compute the length of the vectors
vdotw = 0.               # Initialize vdotw
for i in range(N):       # This loop begins at 0 and ends at N - 1
     vdotw += v[i] * w[i]
```

A simple matrix operation: matrix-matrix product

- We have two matrices **B** and **A**:
  - B has q x n dimensions and the entries are given by b_{ip}.
  - A has n x m dimensions and the entries are given by a_{pj}.

- The product **C = B·A** is given by: $c_{ij} = b_{i1}a_{1j} + \ldots + b_{in}a_{nm} = \sum_{p=1}^{n} b_{ip}a_{pj}$

$$for\ 1 \le i \le q \text{ and } 1 \le j \le m$$

$$\mathbf{B} = \begin{bmatrix} 1 & 2 & 4 \\ 4 & 5 & 8 \end{bmatrix}$$

2 x 3

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 3 & 4 & 5 \\ 0 & 10 & 9 \end{bmatrix}$$

3 x 3

$$\mathbf{C} = \begin{bmatrix} 7 & 49 & 47 \\ 19 & 104 & 101 \end{bmatrix}$$

2 x 3

- Try to program this in Python!!

Python functions for vector and matrix operations: numpy.dot (I)

- Numpy contains many functions to carry out matrix operations. Some important examples for multiplication of vectors and matrices:

- **numpy.dot(A, B)** : dot product of two arrays.
  - A and B are arrays.
  - If A and B are 1-D arrays (vectors), it performs a dot product.
  - If A and B are 2-D arrays (matrices), it performs a matrix multiplication.

Example for 1-D arrays

```
import numpy as np
A = np.array([1, 2, 3])
B = np.array([1, 1/2, 1/3])
C = np.dot(A, B)
print(C)
```

Python functions for vector and matrix operations: numpy.dot (II)

Example for 2-D arrays

```
import numpy as np
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
C = np.dot(A, A)
print(C)
```

It can also multiply two scalars!

Example for scalars

```
import numpy as np
C = np.dot(3, 4)
print(C)
```

Working with complex numbers in Python (I)

- *j* represents the complex number $i = \sqrt{-1}$

```
import numpy as np
z1 = 1. + 1.j  # Don‘t include a * between 1 and j!!
z2 = 1. – 2.j
z3 = z1 + z2
print(‘z3 =‘, z3)
z4 = z1 * z2
print(‘z4 =‘, z4)
```

- To access the real and imaginary parts of the complex variable z, just type z.real and z.imag.

```
print(z1.real)
print(z1.imag)
```

Working with complex numbers in Python (II)

- Complex arithmetic also works for numpy's matrix and vector classes.

```
import numpy as np
F = np.array([[1., 3. + 2.j], [0, 4 - 1.j]])
G = np.array([[2, 4 + 5.j], [5, - 3.j]])
H = F + G
print('F =\n', F)
print('G =\n', G)
print('F + G =\n', H)
```

Other important remarks (I)

Other important functions:

- **numpy.identity**(n) : returns the *n x n* identity array, i.e. a square array of n rows and columns with its main diagonal set to one and the other elements 0.

```
import numpy as np
A = np.identity(2)
print(A)
```

```
[[ 1.  0.]
 [ 0.  1.]]
```

- **numpy.ones**(shape) or **numpy.zeros**(shape): return a new array of given shape (an integer or a sequence of integers) filled with ones or zeros, respectively.

```
B = np.ones((2, 3))
print(B)
```

```
[[ 1.  1.  1.]
 [ 1.  1.  1.]]
```

Other important remarks (II)

- Unit testing is a method to check that the smallest testable parts of a code (units) or application work properly individually. There is an example in your Workshop sheet.
- These tests must be fast and the testing units must be fully independent.
- Python has some tools to carry out a proper unit testing such as the modules pytest, unittest or doctest (out of scope).
- You can simply create a function that checks that a part of your code does what is supposed to do.

Now...

Let’s practice what we have learnt today with the exercises of the worksheet.