Introduction to Scientific Computing

with Python

Utku Kaya

Department of Mathematics
University of Kiel, Germany

26 02 2020

Today’s Overview

- NumPy introduction
- Numerical integration

Installing NumPy

NumPy is the fundamental package for scientific computing with Python.

Be sure that NumPy is installed on your computer!

Type on terminal:

```
sudo apt update
sudo apt install python3-pip
pip3 install numpy
```

Import:

```
>>> import numpy as np
```

Vectors and matrices

A vector can be saved as one dimensional array.

```
>>> v = np.array([2,3,4])
>>> print(v)
>>> v.ndim
```

We created an array corresponding to $v = \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}$

Vectors and matrices

A matrix can be saved as two dimensional array.

```
>>> A = np.array([[1., 2., 3.], [4., 5., 6.]])
>>> print(A)
>>> A.ndim
```

We created an array corresponding to $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

Vectors and matrices

The type of an array can be pre-set.

```
>>> A = np.array([[1., 2.], [4., 5.]], dtype = complex)
```

Ask the size of arrays in each dimension

```
>>> v.shape
>>> A.shape
```

It returns a tuple of integers (dim1, dim2, $\cdots$, dimN).

If $v \in \mathbb{R}^n$ then the shape is (n,).

if $A \in \mathbb{R}^{n \times m}$ (n rows and m columns) then the shape is (n,m).

Vectors and matrices

Transpose of A:

```
>>> print(A.T)
```

Create identity matrix $I \in \mathbb{R}^{n \times n}$, $n = 3$:

```
>>> I = np.eye(3)
```

Vectors and matrices

Create an empty array of shape (3,4):

```
>>> C = np.zeros( (3,4) )
>>> print(C)
```

Create an array with integer elements up to 6:

```
>>> t  = np.arange(6)
>>> print(t)
```

Create an array with elements starting from 0, up to 2 with 0.3 increment:

```
>>> w = np.arange( 0, 2, 0.3 )
>>> print(w)
```

Vectors and matrices

Create an array with 100 elements starting from 0, up to 2π:

```
>>> x = np.linspace( 0, 2 * np.pi, 100 )
>>> print(x)
```

Evaluate sinus function for the each element of array x:

```
>>> y = np.sin(x)
>>> print(y)
```

Arithmetic operations

Arithmetic operators are performed on arrays element-wise!

```
>>> a = np.array( [20, 30, 40, 50] )
>>> b = np.arange( 4 )
>>> b
>>> c = a - b
>>> c
>>> b **= 2
>>> b < 5
```

Array operations

```
>>> a = np.array( [1, 1, 4] )
>>> b = np.array( [2, 0, 5])
>>> a * b                    # dot product
>>> np.sum(a)                # summation of elements of a
>>> np.sum(a[1:])            # sum starting from element nr. 1
>>> np.sum(a[:-1])           # sum until element last element
```

Array operations

```
>>> A = np.array( [[1, 1],
... [0, 1]] )
>>> B = np.array( [[2, 0],
... [3, 4]] )
>>> A.dot(B)          # matrix product
>>> A * B             # elementwise product
>>> np.sum(A)         # summation of elements of A
>>> np.sum(A[0, :])  # sum of elements in row zero of A
>>> np.sum(A[1, :])  # sum of elements in row one of A
```

Midpoint rule

Let $a, b \in \mathbb{R},\ N \in \mathbb{Z}_+$. We are interested in approximations of $\int_a^b f(x)dx$ by using summed quadrature formulas. Consider $N+1$ equidistant points $a = x_0 < x_1 < \cdots < x_N = b$ with $h = (b-a)/N$.

Midpoint rule: $$M_h(f) = h \sum_{i=1}^{N} f\left(\frac{x_i + x_{i-1}}{2}\right).$$

Theorem

If $f \in C^2[a,b]$: $\exists \xi \in [a,b]$ such that

$$\int_a^b f(x)dx - M_h(f) = \frac{b-a}{24} h^2 f''(\xi)$$

For proof see Stoer, Bulirsch.

Trapezoidal rule

Trapezoidal rule: $$T_h(f) = h \sum_{i=1}^{N} \frac{f(x_{i-1}) + f(x_i)}{2}$$

Theorem

If $f \in C^2[a,b]$: $\exists \xi \in [a,b]$ such that

$$\int_a^b f(x)dx - T_h(f) = -\frac{b-a}{12} h^2 f''(\xi)$$

Simpson's rule

Simpson's rule: $S_h(f) = \frac{1}{3}T_h(f) + \frac{2}{3}M_h(f)$

Theorem

If $f \in C^4[a,b]$: $\exists \xi \in [a,b]$ such that

$$\int_a^b f(x)dx - S_h(f) = \frac{b-a}{180}h^4 f^{(4)}(\xi)$$

A numerical integration class

See the Exercise sheet 2 for explanation.

Thanks for your attention!