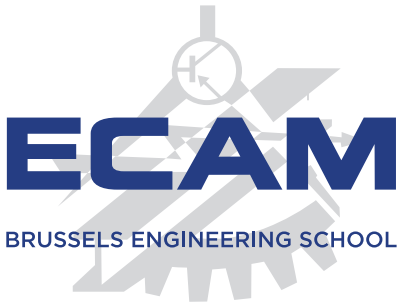


NC301μ Introduction to
Numerical Computing with SciPy

Session 2

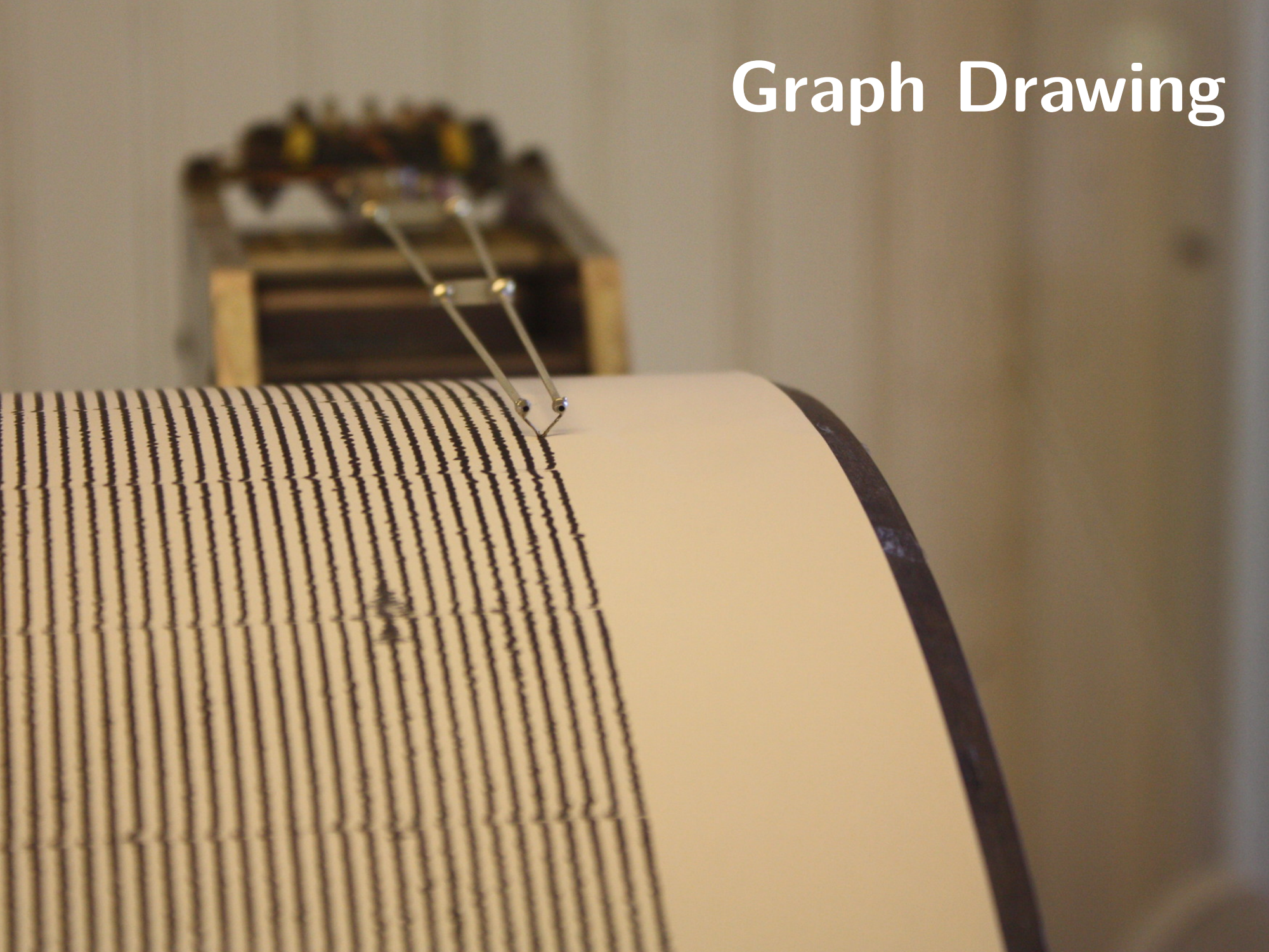
Plotting with Matplotlib and Numerical Analysis with SciPy

Sébastien Combéfis

Winter 2020

Objectives

- Using the `matplotlib` library to draw plots
  - Drawing 2D graphs of functions in the plane on a figure
  - Customising the visual appearance of the rendered graphs
- Discovering how to perform numerical analyses
  - Using the main functions of the `scipy` library
  - Polynomial, integration, optimisation, and interpolation


Graph Drawing

`matplotlib` Library

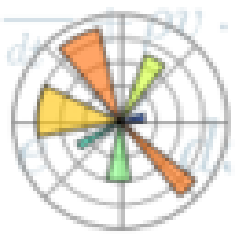
- Drawing of interactive or exportable 2D graphs

  Generation of quality figures ready for publication

- Many features are available in the `matplotlib` library
  - Drawing of curves
  - Histograms
  - 3D drawing is also possible
  - ...

- Open source code available on GitHub

  https://github.com/matplotlib/matplotlib



Drawing a Function (1)

- Using the `plot` function to draw points in the plane

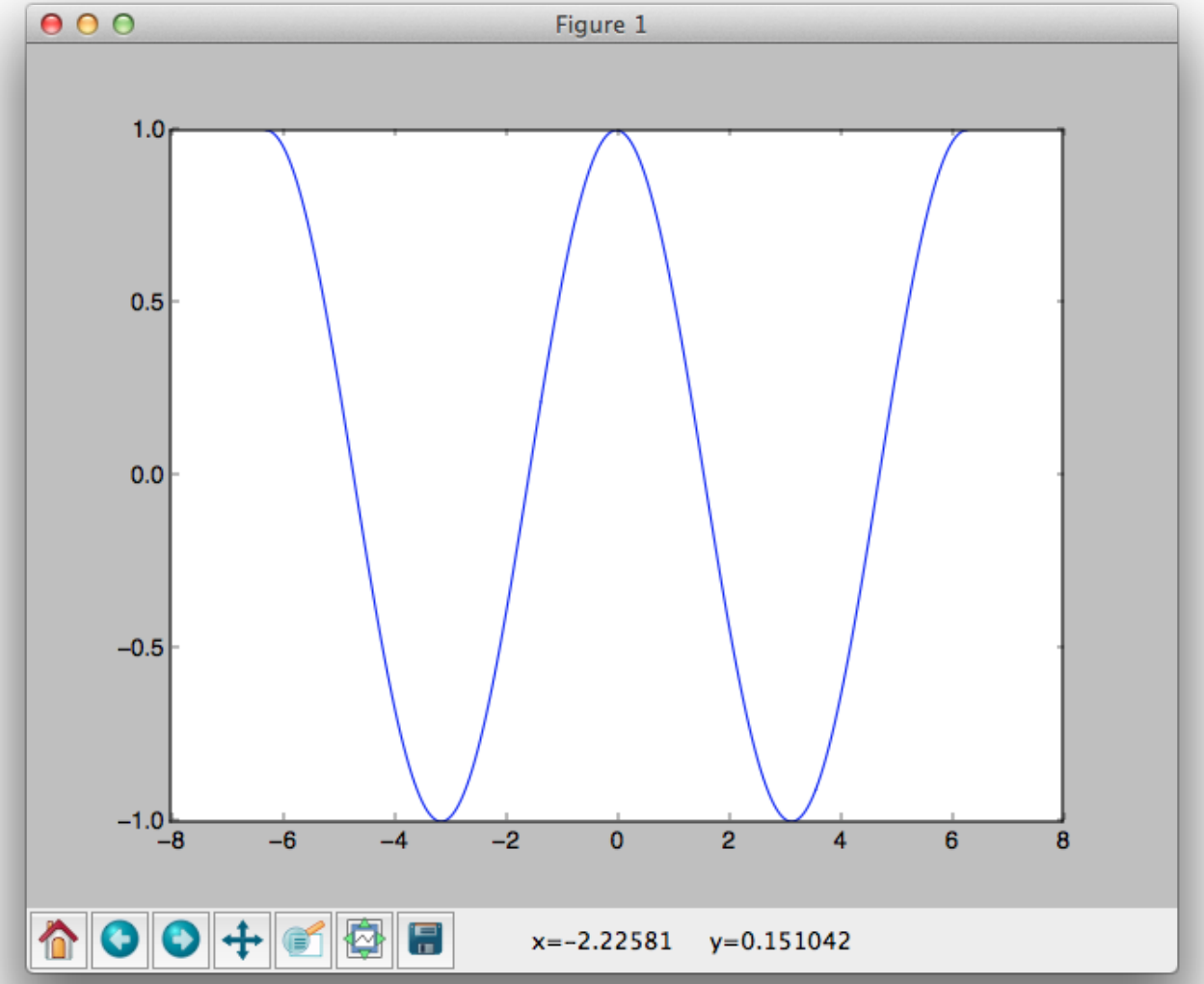
  Creating vectors of data with `numpy` for x and y axes

- Opening the drawing window with the `show` function

  Possible to have several drawings in the same window

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  x = np.linspace(-2*np.pi, 2*np.pi, 256, endpoint=True)
5  y = np.cos(x)
6
7  plt.plot(x, y)
8  plt.show()
```

Drawing a Function (2)



Drawing Configuration (1)

- Configuration of the curves

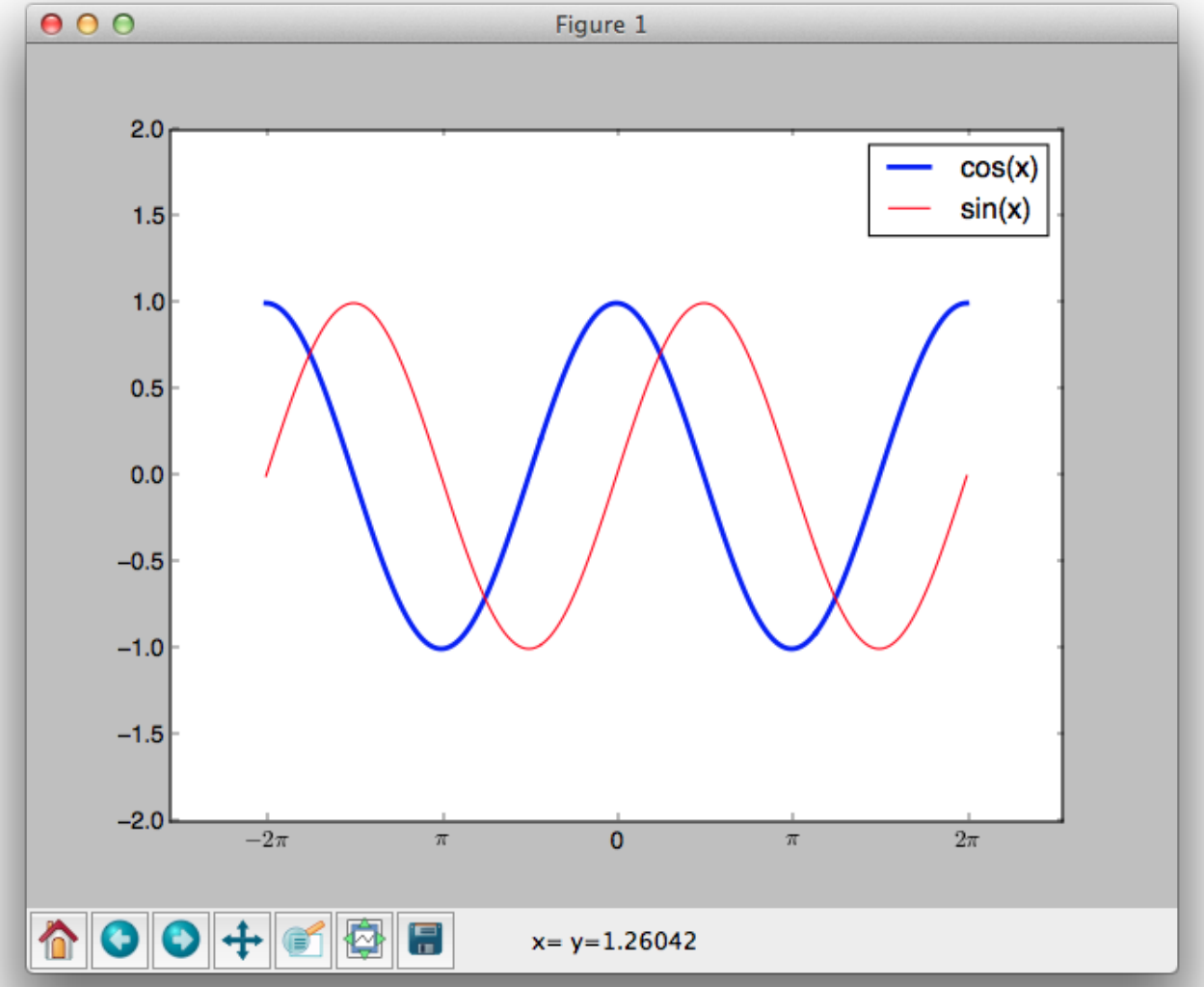
  Line thickness and colour, legend text, etc.

- Adjusting the drawing area and other elements

  Limits of axes, ticks and text (LaTeX support), legend, etc.

```
1  plt.plot(x, c, linewidth=2.5, label='cos(x)')
2  plt.plot(x, s, color='red', label='sin(x)')
3
4  plt.ylim(-2, 2)
5  plt.xticks(
6      [-2*np.pi, -np.pi, 0, np.pi, 2*np.pi],
7      [r'$-2\pi$', r'$\pi$', '0', r'$\pi$', r'$2\pi$']
8  )
9  plt.legend(loc='upper right')
```

Drawing Configuration (2)



Axis Insertion (1)

- Axes are placed inside figures

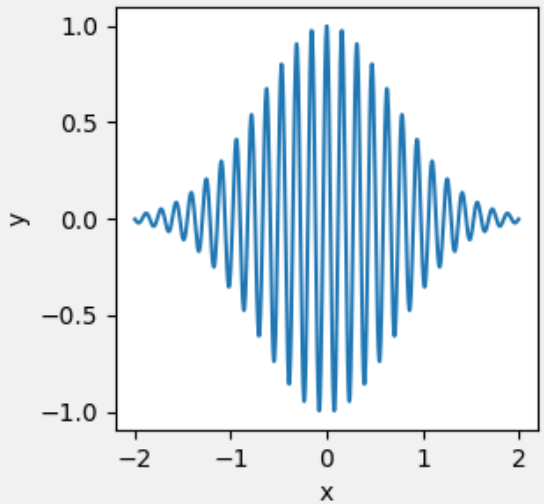
  Size and positioning of the axis in the figure

- Possibility to save the figure in a file

  Choice of the name, depth per inch and background colour

```
1  x = np.linspace(-2, 2, 1000)
2
3  fig = plt.figure(figsize=(8, 4))
4  left, bottom, width, height = 0.1, 0.2, 0.3, 0.6
5  axe = fig.add_axes((left, bottom, width, height))
6
7  axe.plot(x, np.cos(40 * x) * np.exp(-x**2))
8  axe.set_xlabel('x')
9  axe.set_ylabel('y')
10
11 fig.savefig('graph.png', dpi=100, facecolor='#f1f1f1')
```

Axis Insertion (2)



Axis Grid (1)

- Automatic positioning of axes in a figure

  Rectangular grid of axes with m lines and n columns, for example

```
1  x = np.linspace(-10, 10, 200)
2
3  fig, axes = plt.subplots(nrows=1, ncols=2)
4
5  axes[0].plot(x, np.exp(x))
6  axes[1].plot(x, np.sin(x))
7
8  plt.show()
```

Axis Grid (2)

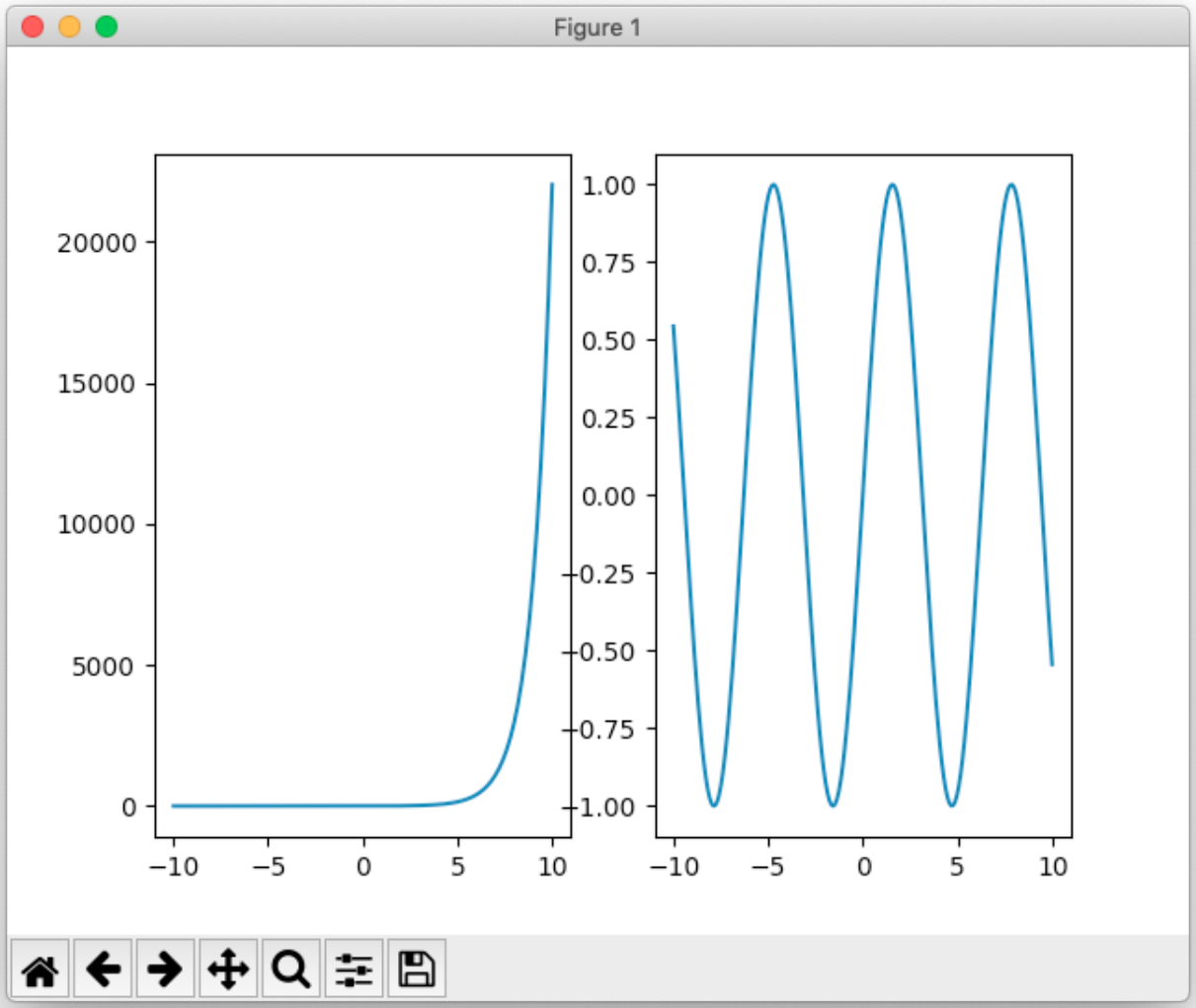


Figure 1
20000
15000
10000
5000
0
−10
−5
0
5
10
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−10
−5
0
5
10

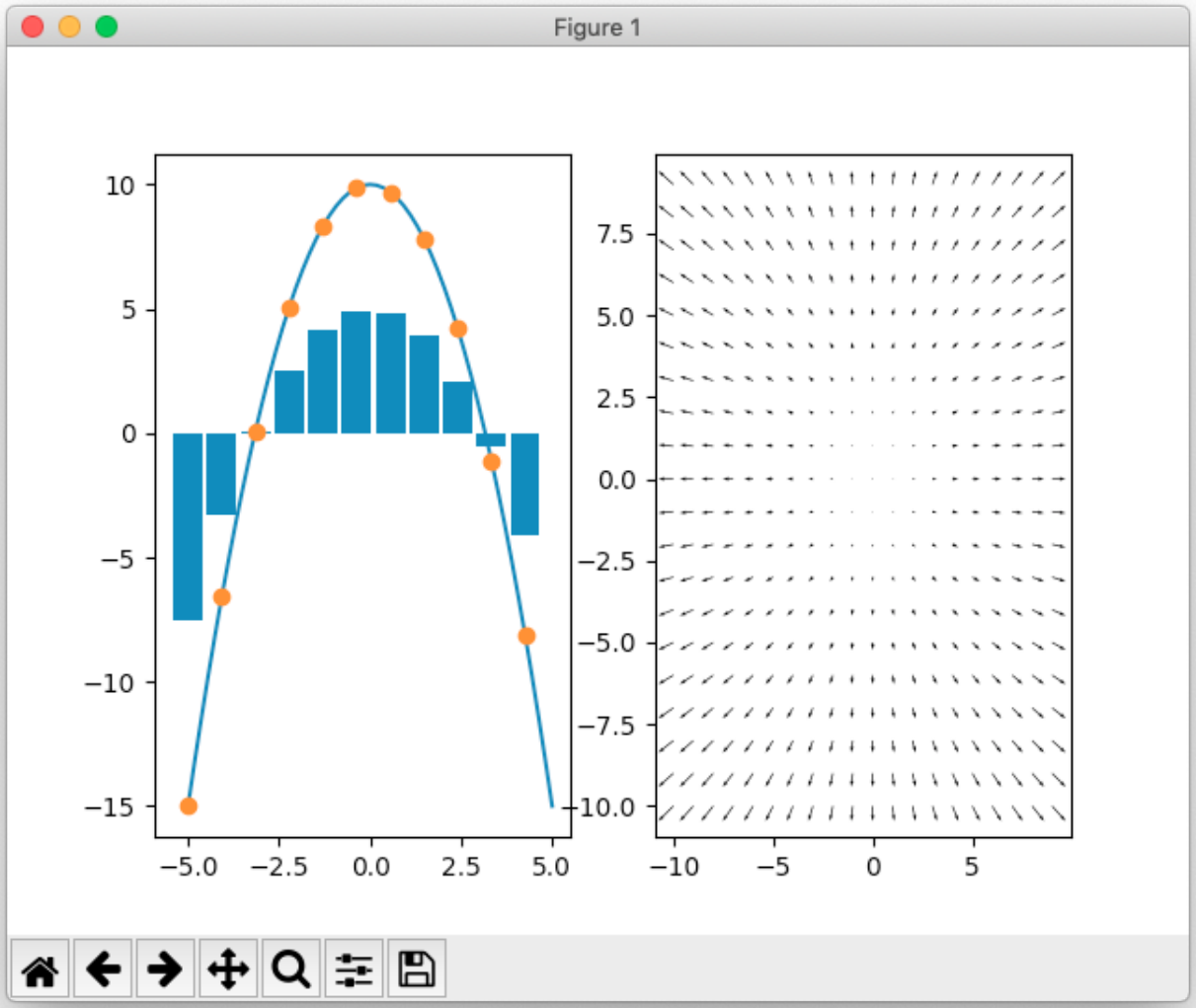
Plot Types (1)

- Several plot types are possible, choice via the axis

  Plot, step, bar, hist, errorbar, scatter, fill_between et quiver

```
1  x = np.linspace(-5, 5, 55)
2  y = -x**2 + 10
3  X = np.arange(-10, 10, 1)
4  U, V = np.meshgrid(X, X)
5
6  fig, axes = plt.subplots(nrows=1, ncols=2)
7
8  axes[0].plot(x, y)
9  axes[0].plot(x[::5], y[::5], 'o')
10 axes[0].bar(x[::5], 0.5 * y[::5])
11 axes[1].quiver(X, X, U, V)
12
13 plt.show()
```

Plot Types (2)




Numerical Analysis

`scipy` Library

- Base package of the `scipy` stack among others

  Algorithms and utility functions built on `numpy`

- Many features are available in the `scipy` library
  - Numerical integration
  - Optimisation
  - Statistical distributions
  - ...
- Open source code available on GitHub

  https://github.com/scipy/scipy

Polynomial

- Polynomial represented using the `poly1d` function

  Polynomial manipulation operations and methods

```
1  from numpy import poly1d
2
3  p = poly1d([1, 2, -1])
4  print(p)
5  print(2 * p)
6  print(p ** 2)
7  print(p.deriv())
```

```
   2
1 x + 2 x - 1
   2
2 x + 4 x - 2
   4      3      2
1 x + 4 x + 2 x - 4 x + 1

2 x + 2
```

Function Vectorisation

- Transform a scalar function into a vector function

  Completely transparent transformation

```
1  import numpy as np
2
3  def add(a, b):
4      return a + b
5
6  vec_add = np.vectorize(add)
7
8  x = [1, 2, 3]
9  y = [7, 8, 9]
10 print(vec_add(x, y))
```

```
[ 8 10 12]
```

Numerical Integration

- Numerical integration with the `scipy.integrate` function

  Specification of the function to integrate and integration bounds

- Several integration methods are available
  - The `quad` function computes a definite integral
  - Other functions are available `romberg`, `trapz`, `simps`...

```
1 import scipy.integrate as integrate
2
3 r = integrate.quad(lambda x: -x + 1, 0, 1)
4 print(r)
```

```
(0.5, 5.551115123125783e-15)
```

Optimisation

- Function optimisation with the `scipy.optimize` function

  Specification of the function to optimise and the method

```
1  from scipy.optimize import minimize
2
3  def obj(x):
4      return x ** 2 - x + 1
5
6  r = minimize(obj, 0, method='nelder-mead', options={'disp': True})
7  print(r.x)
```

```
Optimization terminated successfully.
         Current function value: 0.750000
         Iterations: 23
         Function evaluations: 46
[ 0.5]
```

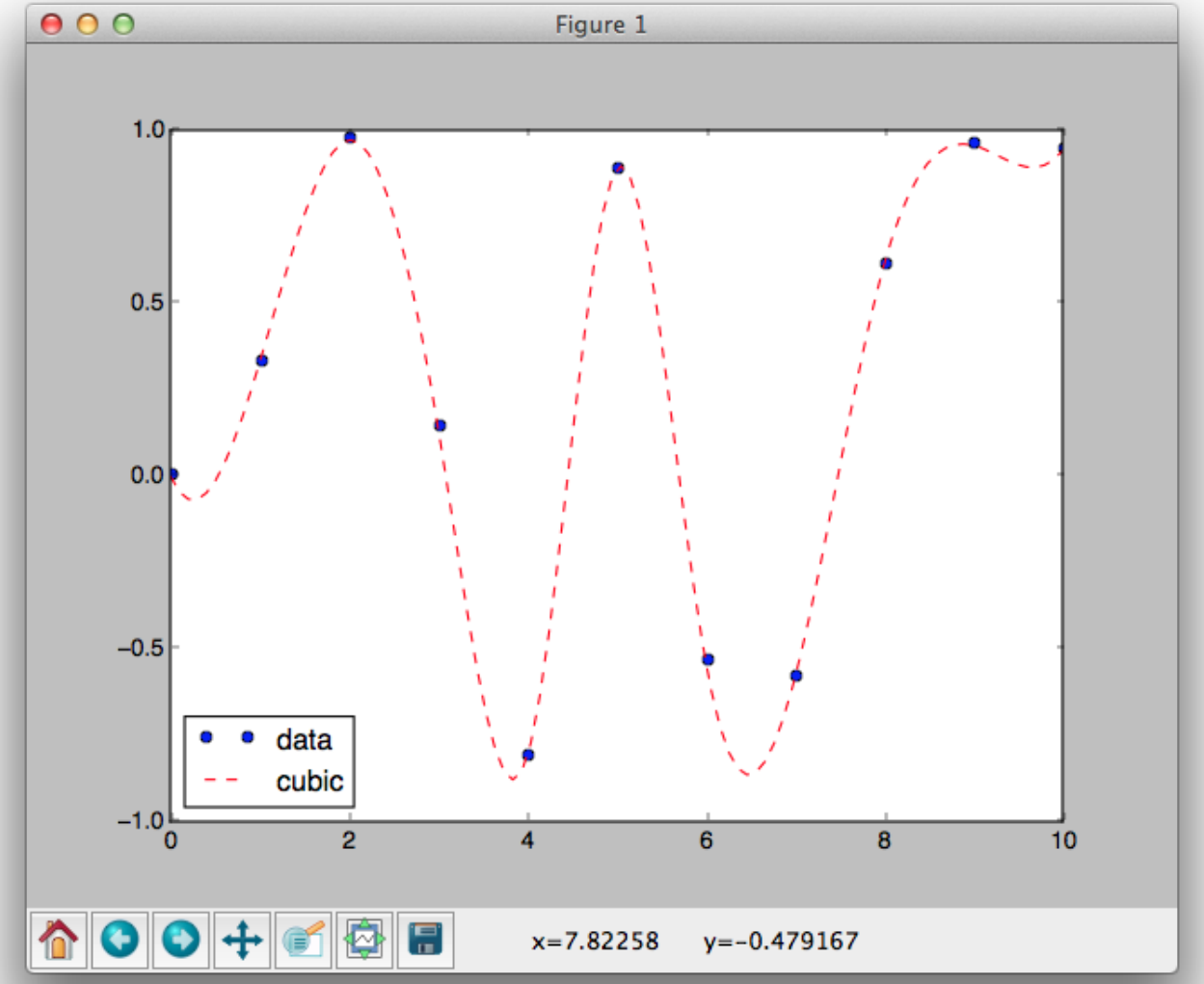
Interpolation (1)

- Interpolation of points with the `scipy.interpolate` function

  Several types of interpolation functions are available

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  from scipy.interpolate import interp1d
4
5  x1 = np.linspace(0, 10, 11, endpoint=True)
6  y1 = np.sin(x1 ** 2 / 3)
7
8  f = interp1d(x1, y1, kind='cubic')
9  x2 = np.linspace(0, 10, 100, endpoint=True)
10
11 plt.plot(x1, y1, 'o', color='blue', label='data')
12 plt.plot(x2, f(x2), '--', color='red', label='cubic')
13 plt.legend(loc='best')
14 plt.show()
```

Interpolation (2)



References

- Badreesh Shetty (2018). *Data Visualization using Matplotlib*, November 12, 2018.
  https://towardsdatascience.com/data-visualization-using-matplotlib-16f1aae5ce70
- Ehi Aigiomawu (2018). *Introduction to Matplotlib — Data Visualization in Python*, May 22, 2018.
  https://heartbeat.fritz.ai/introduction-to-matplotlib-data-visualization-in-python-d9143287ae39
- Sailaja Karra (2020). *Interpolation using Scipy*, February 11, 2020.
  https://medium.com/@sailaja.karra/interpolation-using-scipy-707dba3e8169

Credits

- Ray Bouknight, February 23, 2012, https://www.flickr.com/photos/raybouk/8201310617.
- 8 Kome, September 29, 2019, https://www.flickr.com/photos/kome8/48814920596.