Matplotlib

Пакет Matplotlib является основным для визуализации расчетных данных.

Для рисования графика используем модуль pyplot из пакета.

```
import matplotlib.pyplot as plt
x = [0, 6, 11, 17, 24, 31, 38, 40]
y = [0, 2, 13, 10, 16, 12, 20, 17]
plt.plot(x, y)
plt.show()
```

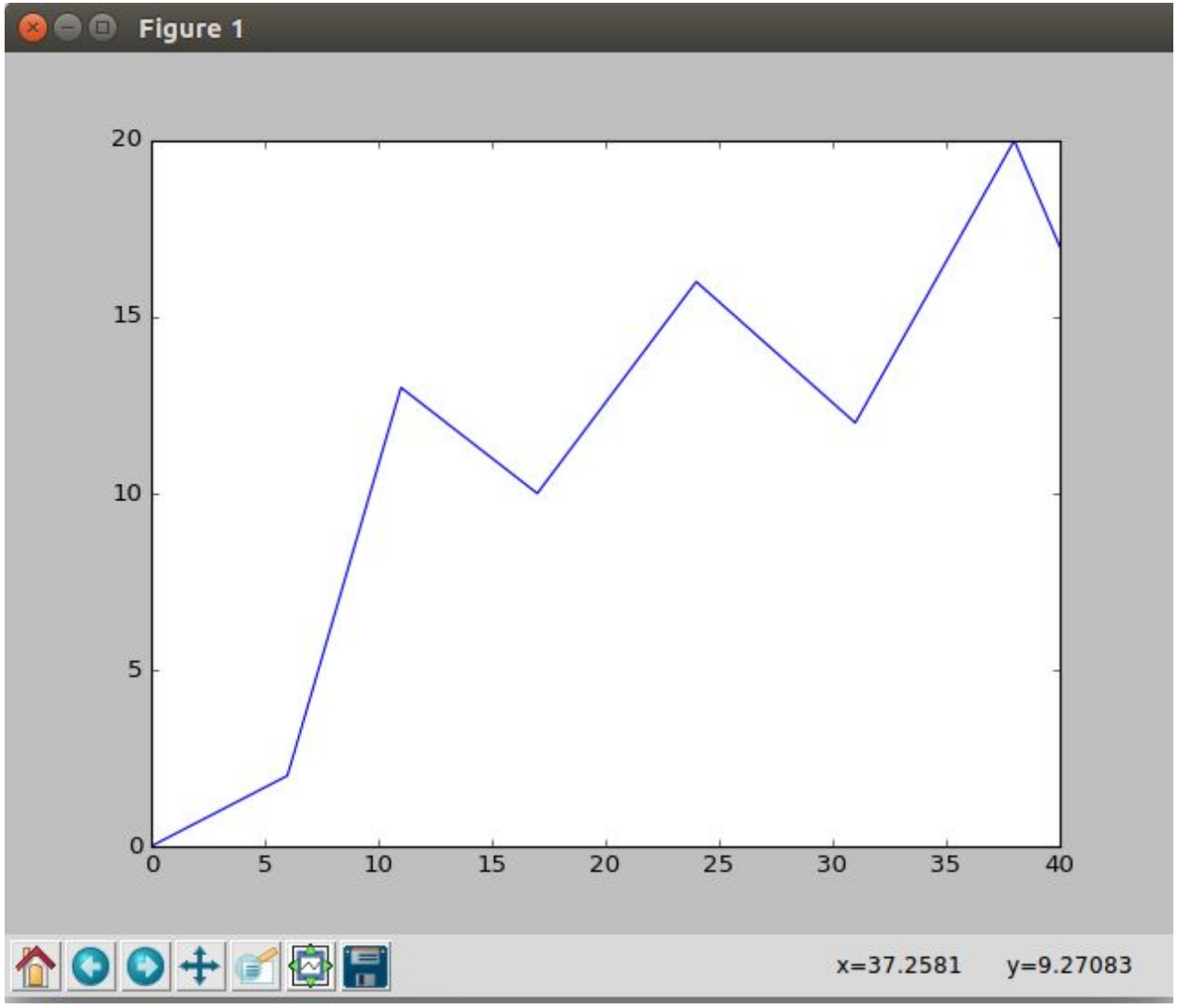
Figure 1
20
15
10
5
0
0
5
10
15
20
25
30
35
40
x=37.2581
y=9.27083

Функция plot

Можно задавать цвета линий, используя необязательный строковый параметр style. При этом можно писать цвет одной буквой.

Таблица цветов

b, blue - синий

k, black - черный

w, white - белый

c, cyan - голубой

r, red - красный

y, yellow - жёлтый

g, green – зелёный

m, magenta - малиновый

Печать графика красным цветом

plt.plot(x, y, ‘red’) plt.plot(x, y, ‘r’)

Символы отрисовки графика:

. точечный маркер

, точки размером с пиксель

o круги

< треугольник вершиной влево

> треугольник вершиной вправо

s квадраты

p пятиконечная звезда

*

h шестиугольники

+ плюсы

D ромбы

d узкие ромбы

| вертикальные линии

plt.plot(x, y, ‘r’) - появятся точки

plt.plot(x, y, ‘k’) - также появятся точки, но их почти не видно

plt.plot(x, y, ‘kh’) - как большие точки

plt.plot(x, y, ‘kp’) - с трудом, но можно рассмотреть звездочки

plt.plot(x, y, ‘ko’) - как большая точка

Тип и толщина линии

символы тип линии

- сплошная линия

-- штриховая линия

-. штрих – пунктирная

: пунктирная линия

plt.plot(x, y, ‘-.’)

Толщину линии нужно задавать аргументом linewidth(lw), а маркера markersize(ms). По умолчанию толщина равна 1.

plt(x, y, ’-.’, linewidth=3)

Заголовок и название осей:

title(), можно указать параметр fontsize, horisontalaligment, verticaligment.

xlabel(),

ylabel().

fontsize: 'large‘, ‘medium’, ‘small’.

horisontalaligment(ha) :‘center’, ‘left’, ‘right’.

verticaligment(va): ‘top’, ‘baseline’, ‘bottom’.

Для надписей на осях используются функции xlabel и ylabel со строковыми параметрами. Функцию axis() можно использовать для указания диапазонов по осям x и y.

axis([0, 10, 0, 20])

Оси можно удалить командой axis(‘off’) и выравнить диапазоны по осям командой axis(‘equal’). При выравнивании диапазонов происходит не полное, а некоторое выравнивание. При этом axis(‘equal’) нужно задавать отдельной строкой. Предыдущие числа, заданные в axis([…]) не принимаются во внимание.

```
import matplotlib.pyplot as plt

x = [0, 6, 11, 17, 24, 31, 38, 40]

y = [0, 2, 13, 10, 16, 12, 20, 30]

#plt.axis([0,10,0,20])

#plt.axis('equal')

plt.title('Graphic points $x^2$') #, fontsize=14, ha='left', va='baseline')

plt.plot(x, y, '-', linewidth=3.5)

plt.xlabel('x')

plt.ylabel('y $f(x)$')

plt.grid(True)

plt.show()
```

Для отображения координатной сетки grid(True).

Функции xticks() и yticks() используются для дополнительного контроля шага координатной сетки.

```
x = [0, 6, 11, 17, 24, 31, 38, 40]
y = [0, 2, 13, 10, 16, 12, 20, 17]
plt.plot( x, y )
plt.xticks(range(0, len(x) + 33,5),[‘a’, ‘b’, ‘c’, ‘d’, ‘e’, ‘f’, ‘g’, ‘h’, ’w’])
plt.yticks(range(0, 23, 4))
plt.grid(True)
plt.show()
```

Очень часто для получения массивов x и y используется пакет numpy и функцию linspace(start, end, n).

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 1, 100)
y = np.sin(x)
plt.plot( x, y )
plt.show()
```

Функция legend()

При этом нужно задать аргумент label в виде строки в функции plot() для каждого графика. Запуск ‘легенды’ производится аргументом loc, принимающим следующие значения

Строка	Код	Строка	Код
‘best’	0	‘center left’	6
‘upper right’	1	‘center right’	7
‘upper left’	2	‘lower center’	8
‘lower left’	3	‘upper center’	9
‘lower right’	4	‘center’	10
‘right’	5		

```
#Пример с использованием нескольких графиков различного оформления
import numpy as np
import matplotlib.pyplot as plt
x1 = np.linspace(-10, 10., 100) #массив абцисс для графиков синуса, косинуса и экспоненты
x4 = np.linspace(-2, 2, 10) #массив абцисс для графика модуля
y1 = np.sin(x1)
y2 = np.cos(x1)
y3 = x1**2 * np.exp(-x1**2)
y4 = abs(x4)

plt.plot(x1, y1, '--', label='sin(x)') #штриховая линия
plt.plot(x1, y2, label='cos(x)')
plt.plot(x1, y3, '-.', label='x**2*exp(-x**2)') #штрих-пунктир
#plt.plot(x1, y3, '-.', label='$x^2 exp(-x^2)$')
```

```
plt.plot(x4, y4, ':', linewidth=5, label='|x|') #пунктирная, толщиной 5
plt.title('4 graphs') #
plt.xlabel('X') #
plt.ylabel('Y') #
plt.legend(loc='upper left') #
plt.grid() #
plt.show()
```

Работа с несколькими независимыми графиками

Для создания новых фигур используется функция figure(). Можно подготовить несколько фигур, каждая из которых может включать несколько графиков.

Нарисуем 2 фигуры: в первой отобразим график функции sin(x), а во второй x*x. Значения (диапазоны) по x будут разными.

```
import numpy as np

import matplotlib.pyplot as plt

x1 = np.linspace(0, 6, 100)

plt.figure(1)

plt.plot(x1, np.sin(x1))

plt.grid(True)

plt.title('$sin(x)$')

x2 = np.linspace(-6, 10, 100)
```

```
plt.figure(2)
plt.plot(x2, x2*x2)
plt.grid(True)
plt.title('$x^2$')
plt.show()
```

Нарисуем 4 графика на одной фигуре: в первой отобразим график функции sin (x), во второй x*x, в третий x*x и t*t по разному, в четвертой -x.

Разделяем фигуру на несколько (у нас 4) прямоугольных областей равного размера, расположенных подобно элементам матрицы. Используем функцию **subplot**, которая и осуществляет такой раздел. Первый аргумент задаёт число строк (у нас будет 2), второй – число столбцов (у нас будет 2), третий – номер графика.

```
import numpy as np

import matplotlib.pyplot as plt

x1 = np.linspace(0, 6, 100)

plt.subplot(221)

plt.plot(x1, np.sin(x1))

plt.axis('equal')

plt.grid(True)

plt.title('$sin(x)$')

x2 = np.linspace(-6, 10, 100)

plt.subplot(222)

plt.plot(x2, x2*x2, 'g')
```

```
plt.grid(True)
plt.title('$x^2$')
x3 = np.linspace(-10, 10, 100)
t = np.arange(-10, 11,1)
plt.subplot(223)
plt.plot(x3, x3*x3, t, t*t, 'ro')
plt.title('$x^2$')
x4 = np.linspace(-10, 10, 100)
plt.subplot(224)
plt.plot(x4, -x4, 'y')
plt.subplot(224).spines['left'].set_position('center')
plt.subplot(224).spines['bottom'].set_position('center')
plt.title('$x$')
plt.show()
```

Гистограмма

Гистограмма используется для изображения зависимости частоты попадания элементов в соответствующий интервал группировки

```
Import matplotlib.pyplot as plt

import numpy as np

x = np.random.normal(0, 3, 1000) # np.random.randn(1000)

plt.hist(x,25) #  по умолчанию 10

plt.show()
```

Столбцовые диаграммы

Для визуализации применяем функцию **bar**(). Для этой функции нужно задать последовательность x, которая определяет левый край столбца и y, задающую высоту.

```
import matplotlib.pyplot as plt

import numpy as np

h1=10*np.random.rand(6) # Генерация данных

h2=10*np.random.rand(6)

h3=10*np.random.rand(6)

pos = np.arange(1, len(h1)+1) # Массив с координатами левых точек первого столбца

wide = 0.3 # Ширина столбцов

plt.bar(pos, h1, width=wide) # создание диаграммы заданной толщины и цветов

plt.bar(pos+wide, h2, width=wide, color='red')

plt.bar(pos+2*wide, h3, width=wide, color='green')

plt.xticks(pos + wide*1.5, pos) # Изменение местоположения засечек на оси x

plt.show()
```

Круговые диаграммы

pie(), т. к. они похожи на разрезаемый пирог.

Первым параметром является последовательность внесенных значений. После следуют необязательные аргументы:

explode – часто кусок ‘ пирога’ выдвигают из центра. Эта последовательность имеет тот же размер, что и первый аргумент.

colors - задает цвета. По умолчанию для matplotlib это blue, green, red, cyan, magenta, yellow.

labels – это имена, у нас названия языков программирования.

labeldistance – определяет радиальные расстояния, на котором эти имена выводятся

auopct – задаёт, как форматируются численные значения

pctdistance – каком расстоянии от центра располагаются числовые значения

shadow – тень: boolean

plt.axes(0.0, 0.0, 1.0, 1.0) - одинаковые размеры по осям.

```
import matplotlib.pyplot as plt
x = [6, 12, 20, 7, 5, 5]
languages = ['Matlab', 'Java', 'Python', 'C', 'C++', 'Other']
plt.figure(figsize=(10,10))
explode = [0, 0, 0, 0.1, 0, 0]
plt.pie(x, labels = languages, explode=explode, autopct='%1.1f%%', shadow=False)
plt.title('Circle diagram')
plt.show()
```