**Лабораторная работа №7**

**Тема:** Построение графиков математических функций в Python

**Цель:** приобрести навыки работы с библиотекой Matplotlib для визуализации данных

**теоретическая часть**

**Matplotlib** - библиотека на языке программирования Python для визуализации данных двумерной 2D графикой (3D графика также поддерживается).Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях. Изображения, генерируемые в различных форматах, могут быть использованы в интерактивной графике, научных публикациях, графическом интерфейсе пользователя, веб-приложениях, где требуется строить диаграммы (англ. Plotting).

Библиотека Matplotlib построена на принципах ООП, но имеет процедурный интерфейс pylab, который предоставляет аналоги команд MATLAB.

Пакет поддерживает многие виды графиков и диаграмм:

* Графики (line plot)
* Диаграммы рассеяния (scatter plot)
* Столбчатые диаграммы (bar chart) и гистограммы (histogram)
* Секторные диаграммы (pie chart)
* Диаграммы «Ствол-листья» (stem plot)
* Контурные графики (contour plot)
* Поля градиентов (quiver)
* Спектральные диаграммы (spectrogram)

Набор поддерживаемых форматов изображений, векторных и растровых, можно получить из словаря FigureCanvasBase.filetypes. Типичные поддерживаемые форматы: EPS, EMF, JPEG, PDF, PNG, PostScript, RGBA, SVG, SVGZ, TIFF.

Рассмотрим построение графиков на примерах.

**набор точек**

>>> **import** **matplotlib.pyplot** **as** **plt**

>>> plt. plot ([1, 3, 2, 4])

[<Matplotlib. lines. Line2D object at 0x01A00430>]

>>> plt. show ()

Рисунок 1

Функция plot () строит график, а функция show () его показывает. Аргумент, что принимается функцией plot () - это последовательность y-значений.Другой, который мы опустили, что стоит перед y - это последовательность x-значений. Поскольку его нет, график генерируется для четырех указанных y, список из четырех x: [0, 1, 2, 3].

**функция**

**from** **numpy** **import** \* # Для использования функцийexpта linspace

**import** **matplotlib.pyplot** **as** **plt**

**def** f (t):

**return** t \*\* 2 \* exp (- t \*\* 2)

t = linspace (0, 3, 51)   *# 51 точка между 0 и 3*

y = f (t)

plt. plot (t, y)

plt. show ()

Рисунок 2

Если функция больше нигде не используется, то можно получить еще более компактный код, задав ее сразу же после определения массива t:

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace (0, 3, 51)

y = t \*\* 2 \* exp (- t \*\* 2)

plt. plot (t, y)

plt. show ()

**Настройка вида графиков**

Кроме того, чтобы просто построить кривую, было бы хорошо ее назвать, обозначить оси, вывести легенду (это особенно пригодится, если строить несколько графиков). Кроме того, иногда нужно изменить вид самой кривой, границы ее построения.

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace (0, 3, 51)

y = t \*\* 2 \* exp (- t \*\* 2)

plt. plot (t, y, "g--", label = 't ^ 2 \* exp (-t ^ 2)')

plt. axis ([0, 3, - 0.05, 0.5])   *# Задания [xmin, xmax, ymin, ymax]*

plt. xlabel ( 't')     *# Обозначения оси абсцисс*

plt. ylabel ( 'y')     *# Обозначения е оси ординат*

plt. title ( 'My first normal plot')   *# Название графика*

plt. legend ()        *# Вставка легенды (текста в label)*

plt. show ()

Рисунок 3

Кроме указанных нововведений, обозначенных в комментариях, в аргументах функции plot () мы видим два новых. Последний задает текст легенды графика. Срочный аргумент g-- ответственен за то, что изменился вид кривой. По сравнению с предыдущим примером, график позеленел (green) и вырисовывается - штриховой линией. По умолчанию этот аргумент b-, что означает синий (blue) сплошную линию. Ниже приведена таблица, которая позволяет выбрать нужный аргумент.

|  |  |
| --- | --- |
| b, blue | синий цвет |
| c, cyan | голубой цвет |
| g, green | зеленый цвет |
| k, black | черный цвет |
| m, magenta | пурпурный цвет |
| r, red | красный цвет |
| w, white | белый цвет |
| y, yellow | желтый цвет |
| - | сплошная линия |
| - | штриховая линия |
| -. | штрих-пунктирная линия |
| : | пунктирная линия |

**Несколько кривых на одном графике**

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace (0, 3, 51)

y1 = t \*\* 2 \* exp (- t \*\* 2)

y2 = t \*\* 4 \* exp (- t \*\* 2)

plt. plot (t, y1, label = 't ^ 2 \* exp (-t ^ 2)')

plt. plot (t, y2, label = 't ^ 4 \* exp (-t ^ 2)')

*# Декоративная часть*

plt. xlabel ( 't')

plt. ylabel ( 'y')

plt. title ( 'Plotting two curves in the same plot')

plt. legend ()

plt. show ()

Рисунок 4

**Несколько кривых на одном графике - 2**

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace (0, 3, 51)

y1 = t \*\* 2 \* exp (- t \*\* 2)

y2 = t \*\* 4 \* exp (- t \*\* 2)

y3 = t \*\* 6 \* exp (- t \*\* 2)

plt. plot (t, y1, "g ^ ',     *# Маркеры из зеленых треугольников*

         t, y2, "b--",    *# Синяя штриховая*

         t, y3, "ro- ')    *# Красные круглые маркеры*

*# Соединены сплошной линией*

plt. xlabel ( 't')

plt. ylabel ( 'y')

plt. title ( 'Plotting with markers')

plt. legend ([ 't ^ 2 \* exp (-t ^ 2)',

            "t ^ 4 \* exp (-t ^ 2) ',

            "t ^ 6 \* exp (-t ^ 2) '],     *# Список легенды*

            loc = 'upper left')     *# Положения легенды*

plt. show ()

Рисунок 5

Мы изменили положение легенды, иначе бы она закрывала красный график. По умолчанию легенда располагается в правом верхнем углу, но можно ее и перенести за счет аргумента loc. Этому аргументу можно присваивать и численное значение, но обычно легче воспринимается строку. В таблице ниже приводятся возможные варианты.

|  |  |  |
| --- | --- | --- |
| Место | String | Code |
| лучший вариант | best | 0 |
| вверху справа | upper right | 1 |
| вверху слева | upper left | 2 |
| внизу справа | lower left | 3 |
| внизу слева | lower right | 4 |
| дело | right | 5 |
| посередине слева | center left | 6 |
| посередине дело | center right | 7 |
| посередине внизу | lower center | 8 |
| посередине вверху | upper center | 9 |
| посередине | center | 10 |

**маркеры**

Здесь также показано как можно объединять сразу три графика в одной инструкции. Кроме того, видно, что можно не только использовать маркеры (y1) или линии (y2), но и объединять их вместе (y3). Наиболее часто в научных исследованиях и журналах приводят графики, отличаются друг от друга именно маркерами, так и в matplotlib для их обозначения существует множество способов:

. точечный маркер;

, Точки, размером с пиксель;

o цепи;

˅ треугольники носом вниз;

Треугольники носом кверху;

> Треугольники смотрят вправо;

<Треугольники смотрят влево;

s квадраты;

p пьятикутнкы;

\* Звездочки;

h шестиугольники;

H возвращены шестиугольники;

+ Плюсы;

x крестики;

D ромбы;

d узкие ромбы;

| вертикальные засечки.

**Дополнительные аргументы plot ()**

Итак, в один аргумент мы можем поставить сразу три параметра: первым указываем цвет, вторым - стиль линии, третьим - тип маркера. Однако уже такая нотация может у человека незнакомого с ней, вызвать удивление. Кроме того, она не позволяет разделять параметры линии и маркера, поэтому существует вариант с использованием keywords - также это позволяет щедрая функция plot ():

|  |  |
| --- | --- |
| **Keyword argument** | **что меняет** |
| color или c | цвет линии |
| linestyle | стиль линии, используются обозначения, показанные выше |
| linewidth | толщина линии в виде float-числа |
| marker | вид маркера |
| markeredgecolor | цвет края (edge) маркера |
| markeredgewidth | толщина края маркера |
| markerfacecolor | цвет самого маркера |
| markersize | размер маркера |

Можно также вносить изменения в отметки на осях координат. Делается это с помощью функций xticks () и yticks (), в которые передаются один или два списка значений: или просто список упомянутых значений, или их же, но сначала те места, на которые они встают:

x = [5, 3, 7, 2, 4, 1]

plt. xticks (range (len (x)), [ 'A', "b", "c", "d", "e", "f '])

plt. yticks (range (1, 8, 2))

Для нанесения сетки существует команда:

plt. grid (True)

Для того, чтобы одну или несколько осей выставить в логарифмическом масштабе применяются команды plt.semilogx () и plt.semilogy ().

**сохранение файла**

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace (0, 3, 51)

y = t \*\* 2 \* exp (- t \*\* 2)

plt. plot (t, y)

plt. savefig ( 'name\_of\_plot.png ", dpi = 200)

Файл сохраняется в той же директории с именем и расширением, указанным в первом аргументе. Второй необязательный аргумент позволяет «на лету» менять разрешение картинки, сохраняется в файл.

Бывает так, что смотреть на картинки в уже настроенной программе не нужно и нужно их именно хранить на будущее, чтобы пересмотреть и сравнить их все вместе. Тогда нам не нужно запускать окно просмотра результатов. Для этого в бывших инструкций посылаем главном модулю сообщение:

**import** **matplotlib**

matplotlib. use ( 'Agg')

**гистограммы**

Для построения гистограмм (диаграмм в виде набора колонок) в Matplotlib используются функция bar и barh, которые строят вертикальные или горизонтальные гистограммы соответственно.Эти функции, как и другие функции рисования, импортируются из модуля pylab.Функции bar и barh имеют множество необязательных параметров с дополнительными настройками, мы рассмотрим только наиболее часто используемые возможности для настройки внешнего вида гистограмм.

Функции bar и barh имеют два обязательных параметра:

• Список координат местоположения столбиков по оси X для bar или по оси Y для barh.

• Значение, задают высоту (длину) столбиков.

Длины этих двух списков должны быть равны.

Простейший пример может выглядеть так:

**import** pylab

xdata = [0, 1, 2, 4, 5, 8]

ydata = [0.1, 0.2, 0.4, 0.8, 0.6, 0.1]

pylab. bar (xdata, ydata)

pylab. show ()

Рисунок 6

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

y = np. random. randn (1000)

plt. hist (y, 25)

plt. show ()

Рисунок 7

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

data1 = 10 \* np. random. rand (5)

data2 = 10 \* np. random. rand (5)

data3 = 10 \* np. random. rand (5)

locs = np. arange (1, len (data1) + 1)

width = 0.27

plt. bar (locs, data1, width = width)

plt. bar (locs + width, data2, width = width, color = 'red')

plt. bar (locs + 2 \* width, data3, width = width, color = 'green')

plt. xticks (locs + width \* 1.5, locs)

plt. show ()

Рисунок 8

**Текст, примечания**

Кроме текста в названиях, подписях к осям, легенды, можно непосредственно вставлять его в график с помощью простой функции text (x, y, text), где x и y координаты, а text текстовую строку. Эта функция вставляет текст в соответствии с координатами данных. Существует возможность вводить и в координатах графика, в которых по (0, 0) принимается нижний левый угол, а за (1, 1) правый верхний. Это делается с помощью функции figtext (x, y, text).

Текстовые функции, конечно вставляют текст в график, но часто бывает нужно именно указать, выделить какой-то экстремум, необычную точку. Это легко сделать с помощью примечаний - функции annotate ( 'annotation ", xy = (x1, y1), xytext = (x2, y2)). Здесь вместо annotation мы пишем текст примечания, вместо (x1, y1) координаты интересной точки, вместо (x2, y2) координаты, где мы хотим вставить текст.

**Задание 1:** изобразить 2d график функции соответственно своему варианту и сохранить в .png файл.

**варианты:**

|  |  |
| --- | --- |
| **№** | **функция** |
| 1. | Y (x) = x \* sin (5 \* x), x = [- 2 ... 5] |
| 2. | Y (x) = 1 / x \* sin (5 \* x), x = [- 5 ... 5] |
| 3. | Y (x) = 2 ^ x \* sin (10x), x = [- 3 ... 3] |
| 4. | Y (x) = x ^ (1/2) \* sin (10 \* x), x = [0 ... 5] |
| 5. | Y (x) = 15 \* sin (10 \* x) \* cos (3 \* x), x = [- 3 ... 3] |
| 6. | Y (x) = 5 \* sin (10 \* x) \* sin (3 \* x), x = [0 ... 4] |
| 7. | Y (x) = sin (10 \* x) \* sin (3 \* x) / (x ^ 2), x = [0 ... 4] |
| 8. | Y (x) = 5 \* sin (10 \* x) \* sin (3 \* x) / (x ^ (1/2)), x = [1 ... 7] |
| 9. | Y (x) = 5 \* cos (10 \* x) \* sin (3 \* x) / (x ^ (1/2)), x = [0 ... 5] |
| 10. | Y (x) = - 5 \* cos (10 \* x) \* sin (3 \* x) / (x ^ (1/2)), x = [0 ... 10] |
| 11. | Y (x) = - 5 \* cos (10 \* x) \* sin (3 \* x) / (x ^ x), x = [0 ... 5] |
| 12. | Y (x) = 5 \* sin (10 \* x) \* sin (3 \* x) / (x ^ x), x = [0 ... 8] |
| 13. | Y (x) = x ^ sin (10 \* x), x = [1 ... 10] |
| 14. | Y (x) = - x ^ cos (5 \* x), x = [0 ... 10] |
| 15. | Y (x) = x ^ cos (x ^ 2), x = [0 ... 10] |
| 16. | Y (x) = cos (x ^ 2) / x, x = [0 ... 5] |
| 17. | Y (x) = 10 \* cos (x ^ 2) / x ^ 2, x = [0 ... 4] |
| 18. | Y (x) = (1 / x) \* cos (x ^ 2 + 1 / x), x = [1 ... 10] |
| 19. | Y (x) = sin (x) \* (1 / x) \* cos (x ^ 2 + 1 / x), x = [- 2 ... 2] |
| 20. | Y (x) = 5 \* sin (x) \* cos (x ^ 2 + 1 / x) ^ 2, x = [1 ... 10] |
| 21. | Y (x) = 5 \* sin (1 / x) \* cos (x ^ 2 + 1 / x) ^ 2, x = [1 ... 4] |
| 22. | Y (x) = 5 \* sin (1 / x) \* cos (x ^ 2) ^ 3, x = [- 4 ... 4] |
| 23. | Y (x) = (x ^ 3) \* cos (x ^ 2), x = [- 2 ... 2] |
| 24. | Y (x) = (x ^ 3) + cos (15 \* x), x = [- 2 ... 2] |
| 25. | Y (x) = (3 ^ x) + cos (15 \* x), x = [- 1 ... 2] |

**Задача 2:** Изобразить гистограмму частоты появления букв в определенном тексте и сохранить в .png файл.

**Задача 3:** Изобразить гистограмму частоты появления в определенном тексте обычных, вопросительных и восклицательных предложений, а также предложений, завершаются многоточием и сохранить в .png файл.