Лабораторная работа по графике №1, 8 декабря

В этой работе вы будете учиться работать с графикой в python. Это потребуется нам для визуализации результатов, которые мы получим с помощью физического моделирования.

1 Matplolib

Для визуализации при использовании SciPy часто применяют библиотеку Matplotlib, являющуюся аналогом средств вывода графики MATLAB. Преимущества Matplotlib следующие в том что он использует Python, а значит мы можем задействовать любую из стандартных или других доступных библиотек, он распространяется по свободной лицензии, что так важно ученым и студентам, обладающим обычно небольшим бюджетом для своих изысканий. Также как и Python, поскольку он на нем основан, Matplotlib портируем на многие операционные системы.

2 Набор точек

Одним из больших преимуществ Matplotlib является та скорость, с которой мы можем построить график и привести первый пример:

```
import matplotlib.pyplot as plt
plt.plot([1, 3, 2, 4])
plt.show()
```

После выполнения последней строчки, вызывается окно Matplotlib, внутри которого мы видим рисунок, показанный справа, который мы можем сохранить в удобном формате с помощью крайней правой иконки в нижней панели, например, обычном графическом формате PNG. Что же мы построили?

В первой инструкции мы импортировали основной модуль библиотеки для построения графиков под именем plt, именно так наиболее часто сокращается это длинное имя. После того, как мы импортировали модуль, мы можем пользоваться его функциями, здесь мы использовали функцию plot(), которая собственно и строит график, а потом функция show() его нам показывает.

Аргумент, принимаемый функцией plot() это последовательность y-значений. Другой, который мы опустили, стоящий перед y — это последовательность x-значений. Поскольку его нет, генерируется для четырех указанных y, список из четырех x: [0, 1, 2, 3]. Отсюда и такой график.

3 Функции

Естественно использовать две координаты вместе и вместо списков массивы. Давайте построим график какой-нибудь сложной функции, например, $y(t) = t^2 e^{-t^2}$.

```
from numpy import *
import matplotlib.pyplot as plt

def f(t):
    return t**2*exp(-t**2)

t = linspace(0, 3, 51)
```

```
y = f(t)

plt.plot(t, y)
plt.show()
```

Как видно, код такой же ясный и простой, как полученный с помощью него график. 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()
```

4 Упражнения

Ниже приведены некоторые программы. Ваша задача — набрать эти программы, посмотреть на результат, после чего вы должны прокомментировать каждую строчку, ясно объясняя, что она делает. Если вы даете комментарий к некоторой функции, нужно описать, что она принимает в аргументы.

4.1 Украшения

```
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])
plt.xlabel('t')
plt.ylabel('y')
plt.title('My first normal plot')
plt.legend()

plt.show()
```

4.2 Несколько кривых

```
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)')

plt.xlabel('t')
plt.ylabel('y')
plt.title('Plotting two curves in the same plot')
plt.legend()

plt.show()
```

4.3 Еще упражнение

```
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^',
         t, y2, 'b—',
         t, y3, 'ro-')


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)'],
            loc='upper left')

plt.show()
```