Лабораторная работа № 8

# *Использование библиотек SymPy и SciPy*

## Указания к выполнению лабораторной работы

**1. SymPy –** это пакет для символьных вычислений на питоне, подобный математической системе Mathematica.

В SymPy есть разные функции, которые применяются в сфере символьных вычислений, математического анализа, алгебры, дискретной математики, квантовой физики и пр. Результат может представляться в разных форматах: LaTeX, MathML и так далее.

**2. SciPy –** это библиотека Python с открытым исходным кодом, предназначенная для решения научных и математических задач.

Она построена на базе NumPy и позволяет управлять данными, а также визуализировать их с помощью разных высокоуровневых команд.

Если вы уже импортировали SciPy, то NumPy отдельно импортировать не нужно, но не наоборот!

Таблица 1

**Пакеты в SciPy**

| Название | Описание |
|---|---|
| `constants` | Физические и математические константы |
| `integrate` | Решение интегральных и обычных дифференциальных уравнений |
| `interpolate` | Интерполяция и сглаживание |
| `linalg` | Линейная алгебра |
| `optimize` | Оптимизация и численное решение уравнений |
| `sparse` | Разреженные матрицы |
| `stats` | Статистические распределения и функции |

И другие

В лабораторной работе требуется написать собственную программу на языке программирования Python с использованием стандартных функций из библиотеки SciPy.

В качестве отчета по работе преподавателю предъявляются решения в электронном виде (файлы .py или .ipynb). При необходимости нужно ответить на дополнительные вопросы.

## Задание на лабораторную работу

**Задание 1.** *Символьное дифференцирование функции одной переменной*

1. Подключить символьную библиотеку и инициализировать отображение формул:
```
from sympy import *
init_printing()
```

2. Создать и присвоить переменным объекты класса Symbol:
```
x=Symbol('x')
```

3. Описать функцию своего варианта в виде отдельной функции на Python. Её заголовок может иметь следующий вид:
```
def f(x):
```

4. Вывести свою функцию и оценить её представление. Сравнить с записью на языке программирования.

5. Вызвать функцию дифференцирования, оценить результаты
```
diff(f(x),x)
```

| **Вариант** | **Функция** |
|---|---|
| 1 | $y = x\,tg\,x + lncos\,x + e^{5x}.$ |
| 2 | $y = \cos x \ln tg\,x - \ln tg\,(x/2).$ |
| 3 | $y = x \arccos(x/2) - \sqrt{4 - x^2}.$ |
| 4 | $y = x(\sin\ln x - \cos\ln x).$ |
| 5 | $y = x^2 + x \arcsin x + \sqrt{1 - x^2}.$ |
| 6 | $y = x\sqrt{4 - x^2} + 4 \arcsin(x/2).$ |
| 7 | $y = \sqrt{x} - (1 + x)\,arctg\sqrt{x}.$ |
| 8 | $y = \ln tg(x/2) - \frac{x}{\sin x}.$ |
| 9 | $y = e^x(\cos 2x + 2\sin 2x).$ |
| 10 | $y = x\,arctg\,x - \ln\sqrt{1 + x^2}.$ |

**Задание 2.** Для функции двух переменных своего варианта найти частные производные первого и второго порядков. Экспериментально убедиться в справедливости теоремы о равенстве смешанных производных.

| **Вариант** | **Функция** |
|---|---|
| 1 | $z = e^{x^2 + y^2}$ |
| 2 | $z = 2e^{\sin xy}$ |
| 3 | $z = \ln\left(x + \frac{y}{2x}\right)$ |
| 4 | $z = \ln\left(\frac{x}{y} + \frac{y}{x}\right)$ |
| 5 | $z = arctg\sqrt{x^y}$ |

| | |
|---|---|
| 6 | $z=\sqrt{1+\frac{y}{2x}}$ |
| 7 | $z=\arcsin\frac{x+y}{xy}$ |
| 8 | $z=e^{xy}(\sin x+\cos y)$ |
| 9 | $z=(2x+y)^{2x+y}$ |
| 10 | $z=\frac{xy}{x^2+y^2}$ |

**Задание 3.** *Поиск минимума функции одной переменной*

Для функции своего варианта найти экстремум с помощью стандартной функции `minimize` и градиентным методом `BFGS`.

1. Импортировать минимизацию из библиотеки SciPy:
```
from scipy.optimize import minimize
```

2. Описать функцию своего варианта в виде отдельной функции на Python.

3. Построить график и убедиться в унимодальности функции.
```
xArr = np.arange(-5., 5.) #диапазон изменения x
yArr = np.array([f(x) for x in xArr])
import matplotlib.pyplot as plt
plt.plot(xArr, yArr)
plt.grid(True)
plt.axis([-5, 5, -15, 5])
plt.show()
```

   **Примечание:** если у функции Вашего варианта на графике определяется максимум, а не минимум, в описании нужно поменять знак (умножить всё на –1).

4. Вызвать функцию минимизации и вывести результаты решения и число итераций:
```
    minFunc1Value1 = minimize(f,22) #в качестве второго параметра указывается начальная точка
    print("Minimized f(x) (standard method): ",
    minFunc1Value1.fun, " for x = ", minFunc1Value1.x)
    print("Number of iterations: ", minFunc1Value1.nit)
```

| **Вариант** | **Функция** |
|---|---|
| 1 | $y=2x^2+\frac{1}{x}$ |

| | |
|---|---|
| 2 | $y=\frac{\ln x}{\sqrt{x}}$ |
| 3 | $y=x+e^{-x}+1$ |
| 4 | $y=x-arctg\,x$ |
| 5 | $y=e^{x^2-4x+5}$ |
| 6 | $y=\sqrt{5-2x}+x$ |
| 7 | $y=e^{\frac{1}{x+2}}$ |
| 8 | $y=\ln\left(1-x^2\right)$ |
| 9 | $y=\frac{x^3}{3-x^2}$ |
| 10 | $y=x-\ln x$ |

5. Укажите в scipy.optimize.minimize в качестве метода BFGS (один из самых точных в большинстве случаев градиентных методов оптимизации), запустите из начального приближения. Градиент функции при этом указывать не нужно – он будет оценен численно.

```
minFunc1Value2 = minimize(f, 22, method = 'BFGS')
```

6. Варьируя начальную точку в обоих методах, сравните получаемые результаты. Какая задача оптимизации решается: локальная или глобальная?

**Задание 4.** *Поиск минимума функции многих переменных*

Будем минимизировать функцию $z = x^2 + y^2 – 4(x – y)$.

1. Описать функцию на языке Python. Обратите внимание, что в качестве аргумента **должен быть np.array!**
```
def target_func(x):
  return x[0]**2+x[1]**2 - 4*(x[0] - x[1])
```

2. Задать начальную точку (0; 0) тоже в виде np.array:
```
x0 = np.array([0, 0])
```

3. Вызвать функцию минимизации методом Нелдера-Мида и вывести результат:
```
res = minimize(target_func, x0, method='nelder-
mead')
print(res)
```

Обратите внимание, что выводится вся информация о процессе оптимизации. Исправьте вывод таким образом, чтобы выводилась точка минимума, соответствующее значение функции и количество итераций.

4. Найти минимум методами сопряженных градиентов (’CG’) и BFGS. Сравнить скорость сходимости методов. Какой самый быстрый?

5. Найти экстремум функции аналитически и оценить абсолютные погрешности всех использованных алгоритмов. Какой метод точнее?

**Задание 5.** *Метод наименьших квадратов*

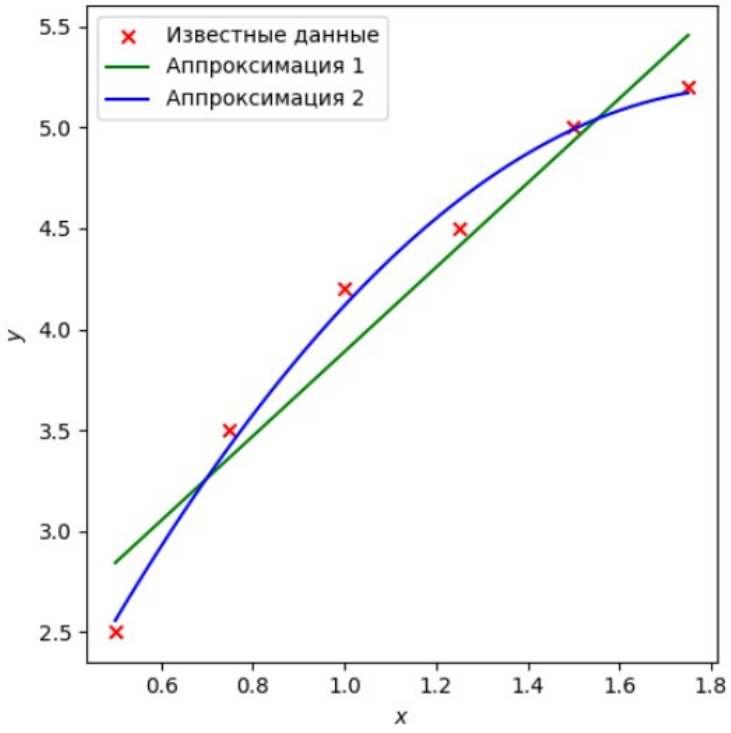
Дана таблица значений некоторой функции. Используя возможности библиотек NumPy и SciPy, определить вид функциональной зависимости.

1. Описать исходные данные в виде массивов np.array:
```
x = np.array([])
y = np.array([])
```

2. Импортировать из NumPy полиномиальную аппроксимацию:
```
from numpy.polynomial import Polynomial
```

3. Выполнить полиномиальную аппроксимацию функциями с степени 1 (линейная) и 2 (квадратичная):
```
approximation1 = Polynomial.fit(x, y, 1) #Пример для линейной аппроксимации
```

4. Изобразить на одном графике исходные данные и две полученные функции. Пример:




5. Вычислить величину суммы квадратов отклонений для полученных приближений.

6. Выполнить нелинейную аппроксимацию с помощью метода optimize.curve_fit для функций вида:

3. $y = ax^m$
4. $y = ae^{mx}$
5. $y = \frac{1}{ax+b}$
6. $y = a \ln x + b$
7. $y = a \frac{1}{x} + b$
8. $y = \frac{x}{ax+b}$

В качестве параметров в curve_fit передаётся приближающая функция, данные, под которые эту функцию необходимо «подогнать», и начальное приближение, используемое в качестве стартовой точки для подбора параметров, при которых достигается минимальное среднее квадратичное отклонение от заданного набора точек.
Пример для экспоненциальной зависимости:

```
from scipy import optimize

def exp_approx(x, a, m):
    return a * np.exp(m * x)

(a, b), _ = optimize.curve_fit(exp_approx, x, y, p0=[1, -1])
```

Обратите внимание, что не все данные могут быть аппроксимированы любой функцией!

7. Вычислить величину суммы квадратов отклонений для полученных приближений. Какая аппроксимация оказалась наиболее точной?

**Вариант 1**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.5 | 0.75 | 1.0 | 1.25 | 1.5 | 1.75 |
| *y* | 2.5 | 3.5 | 4.2 | 4.5 | 5.0 | 5.2 |

**Вариант 2**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.50 | 0.75 | 1.00 | 1.25 | 1.50 | 1.75 |
| *y* | 5.10 | 3.50 | 3.10 | 2.50 | 2.30 | 2.00 |

**Вариант 3**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.50 | 0.75 | 1.00 | 1.25 | 1.5 | 1.75 |
| *y* | 0.26 | 0.82 | 2.1 | 4.0 | 6.6 | 10.8 |

**Вариант 4**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.50 | 0.75 | 1.00 | 1.25 | 1.50 | 1.75 |
| *y* | 6.1 | 5.1 | 3.9 | 2.6 | 0.9 | -0.8 |

**Вариант 5**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.5 | 0.75 | 1.00 | 1.25 | 1.5 | 1.75 |
| *y* | -2.5 | -3.5 | -4.2 | -4.5 | -5.0 | -5.2 |

**Вариант 6**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.50 | 0.75 | 1.00 | 1.25 | 1.50 | 1.75 |
| *y* | -5.10 | -3.50 | -3.10 | -2.50 | -2.30 | -2.00 |

**Вариант 7**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.50 | 0.75 | 1.00 | 1.25 | 1.50 | 1.75 |
| *y* | -0.26 | -0.82 | -2.1 | -4.0 | -6.6 | -10.8 |

**Вариант 8**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.50 | 0.75 | 1.00 | 1.25 | 1.50 | 1.75 |
| *y* | -6.1 | -5.1 | -3.9 | -2.6 | -0.9 | 0.8 |

**Вариант 9**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.5 | 0.75 | 1.00 | 1.25 | 1.5 | 1.75 |
| *y* | 2.5 | 3.5 | 4.2 | 4.5 | 5.0 | 5.2 |

**Вариант 10**

| | | | | | | |
|---|---|---|---|---|---|---|
| *x* | 0.50 | 0.75 | 1.00 | 1.25 | 1.50 | 1.75 |
| *y* | -5.10 | -3.50 | -3.10 | -2.50 | -2.30 | -2.00 |

**Справочная информация**

https://fadeevlecturer.github.io/python_lectures/notebooks/scipy/interpolation_approx.html

https://habr.com/ru/articles/827018/