# Лабораторная работа 1. Теория погрешности

Подключим ноебходимые библиотеки: matplotlib.pyplot – для построения графиков, numpy – для работы с массивами.

```
import numpy as np
import matplotlib.pyplot as plt
```

## Задача 3. Экономизация степенного ряда

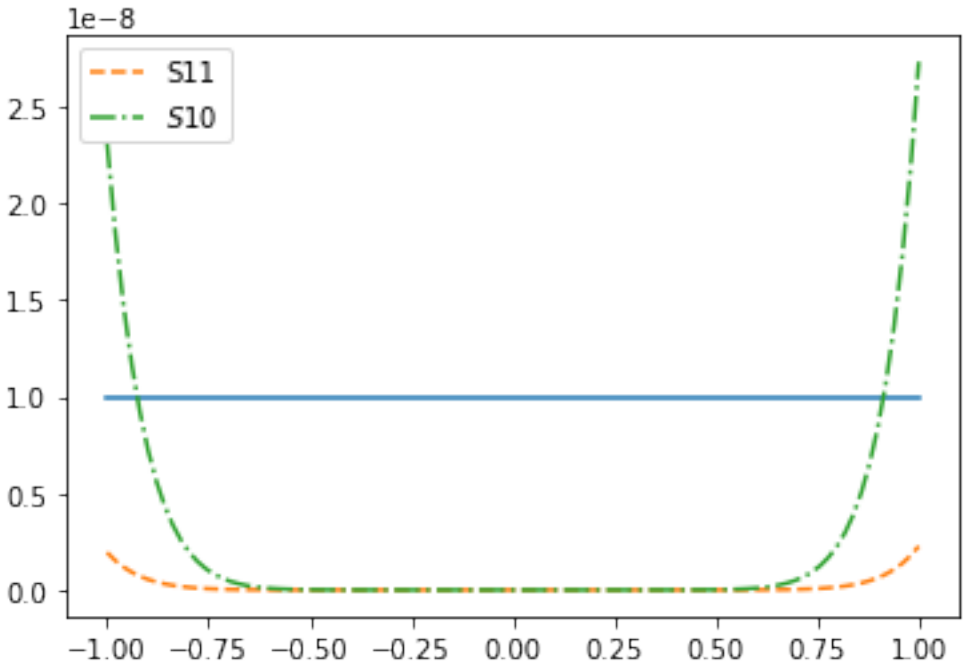
Определим функцию, которая по коэффициентам вычисляет частичную сумму ряда

```
#Реализуйте функцию с помощью цикла while.
#Обращаться к элементам массива можно традиционно: coef[i].
#coef.size (это не функция!) хранит размер массива.
def S(t,coef):

```

Вычислим коэффициенты разложения функции $f(x) = e^x$ с точностью до $\varepsilon = 10^{-8}$. Экспериментальным путем с помощью графика находим, что нужно раскладывать экспоненту вплоть до $x^{11}$.

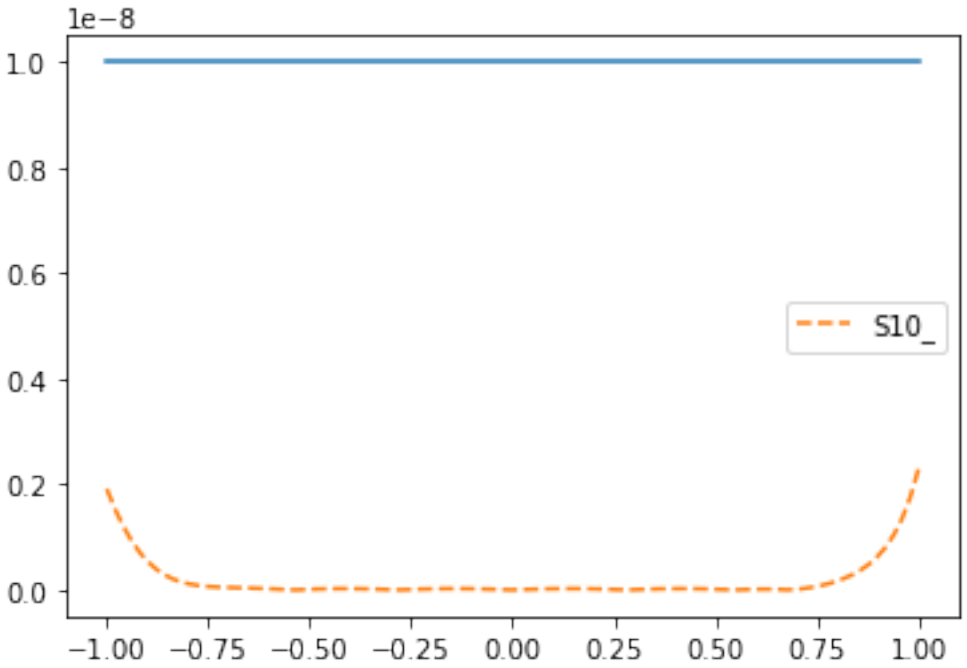
```
#Создаем массив питру состоящий из 12 элементов и заполненный нулями.
coef = np.zeros(12)
#Заполняем его оптимальным способом нужными нам значениями.
#В данном примере он заполняется значениями для экспоненты
coef[0] = 1
i = 1
while i < coef.size:
    coef[i] = coef[i-1]/i
    i += 1

#Для построения графика нам необходим массив значений по оси абсцисс
#Команда linspace создает массив чисел от -1 до 1, в количестве 100 штук.
#endpoint=true значит, что число 1 должно входить в этот набор.
x = np.linspace(-1,1,100,endpoint=True)

#Создадим массив значений функции, в данном случае константы, равной точности␣
 ↪нашей задачи.
#Для создания массива используется уже знакомая вам команда создания массива,␣
 ↪заполненого нулями.
#Далее над ним совершается арифметическая операция прибавления 10**(-8), которая␣
 ↪совершается поэлементно.
#Таким образом получаем массив из 100 элементов, заполненный значением точности␣
 ↪нашей задачи.
err = np.zeros(100)+10**(-8)

#Команда, строящая (но не отображающая) график.
plt.plot(x,err)
```

```
#Следующие две команды строят графики погрешности, при этом команды получают два␣
 ↪дополнительных аргумента.
#Аргумент ls определяет стиль линии, а label определяет имя кривой.
#Для отображения имен кривых, то есть создания "легенды", нужно вызвать команду␣
 ↪legend().
#Обратите внимание, что здесь в качестве второго параметра стоит комбинация из␣
 ↪арифметических операций
#и вычисления абсолютного значения массивов. Все они являются поэлементными и␣
 ↪возвращают массив того же размера.
plt.plot(x,np.abs(np.exp(x)-S(x,coef)),ls="--",label="S11")

#Построим на том же графике вторую кривую.
#Обратите внимание, что функция S вторым параметром получает coef[0:11]. Такая␣
 ↪конструкция называется срезом.
#По сути это подмассив исходного массива с нулевого по десятый (включая) элемент.
 ↪ Важно, что это не копия этих данных,
#а ссылка на эти данные.
plt.plot(x,np.abs(np.exp(x)-S(x,coef[0:11])),ls="-.",label="$S10$")

#Создаем легенду
plt.legend()

#Команда показывает график. После её вызова нарисованные данные сбрасываются.
#Чтобы сохранить график как изображение в нужном вам качестве можно вместо show␣
 ↪вызвать
#plt.savefig("name.png", dpi = 500), где dpi задает качество.
plt.show()
```

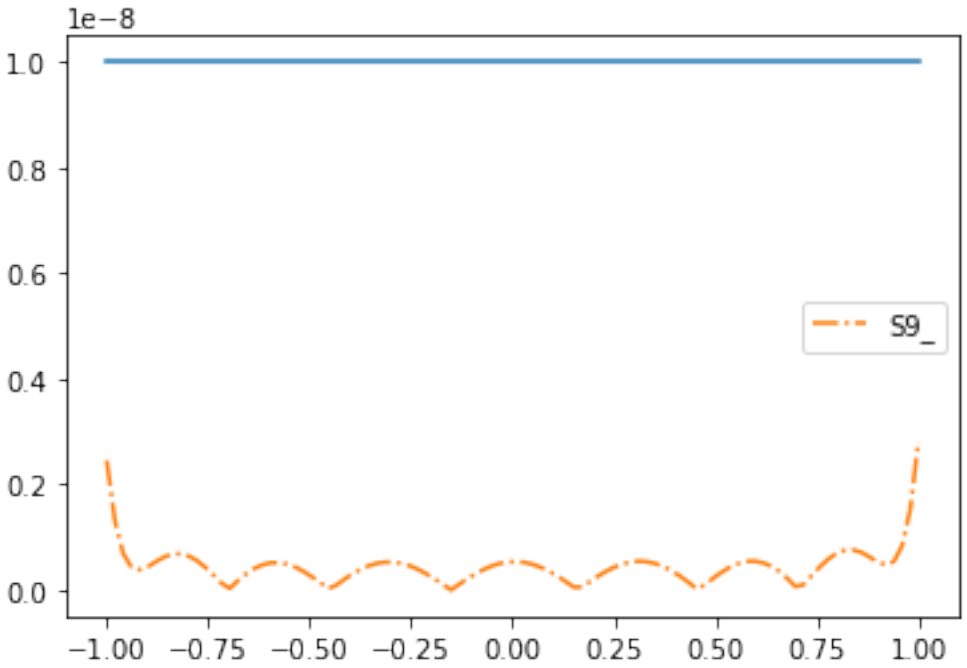



Произведем экономизацию ряда до тех пор, пока сохраняется точность вычисления

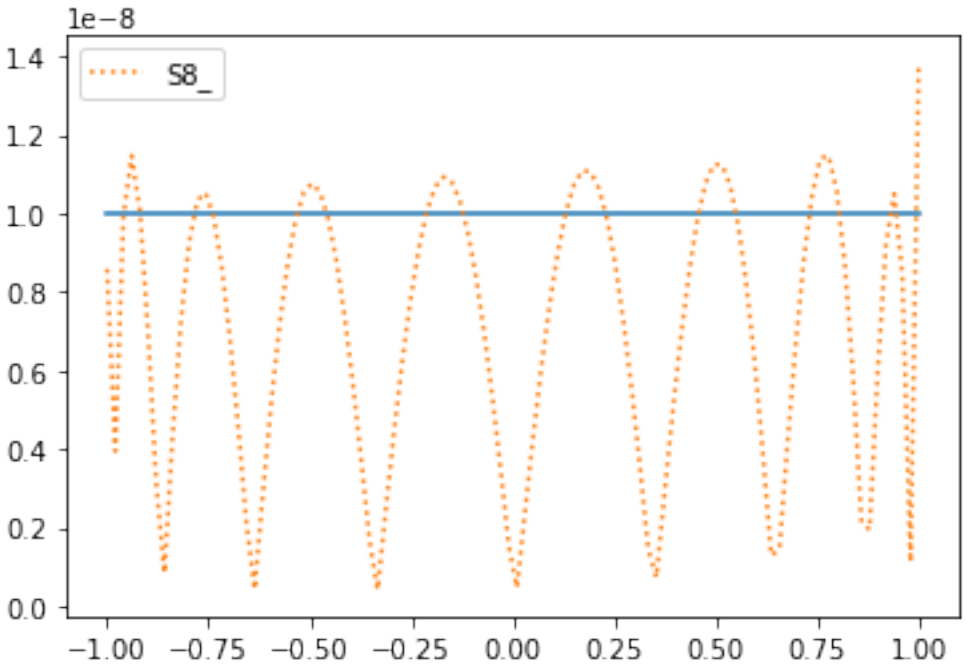
```
#Копируем в новую переменную подмассив с нулевого по десятый элемент.
coef10 = coef[0:11].copy()

#Пользуясь формулами экономизации обновляем значения коэффициентов ряда.
#Вам необходимо проделать это самостоятельно.

#Для проверки, что сохраняется точность вычислений, строим график
plt.plot(x,err)
plt.plot(x,np.abs(np.exp(x)-S(x,coef10)),ls="--",label="S10_")
plt.legend()
plt.show()
```




```
#Копируем в новую переменную подмассив с нулевого по девятый элемент.
coef9 = coef10[0:10].copy()

#Пользуясь формулами экономизации обновляем значения коэффициентов ряда.
#Вам необходимо проделать это самостоятельно.

#Для проверки, что сохраняется точность вычислений, строим график
plt.plot(x,err)
plt.plot(x,np.abs(np.exp(x)-S(x,coef9)),ls="-.",label="S9_")
plt.legend()
plt.show()
```




```
#Копируем в новую переменную подмассив с нулевого по восьмой элемент.
coef8 = coef9[0:9].copy()

#Пользуясь формулами экономизации обновляем значения коэффициентов ряда.
#Вам необходимо проделать это самостоятельно.

#Для проверки, что сохраняется точность вычислений, строим график
plt.plot(x,err)
plt.plot(x,np.abs(np.exp(x)-S(x,coef8)),ls=":",label="S8_")
plt.legend()
plt.show()
```




Делаем вывод, что уменьшить количество слагаемых можно вплоть до $x^9$.