In [ ]:

```
%matplotlib inline
import matplotlib
from matplotlib import pyplot as plt
matplotlib.rcParams.update({'font.size': 14})
# plt.rcParams['figure.figsize'] = (12,6)
```

Задание 1.

Найдите выражение для *ошибки метода* при вычислении производной функции с помощью конечной разности вида:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

(*разложите значения функции по формуле Тейлора относительно точки* x)

Задание 2.

- Оцените ошибку округления для формулы из предыдущей задачи.
- Получите выражение для оптимального шага h, при котором оценка суммарной ошибки минимальна.
- Получите выражение для минимальной оценки ошибки.

Задание 3. (*используйте заготовку кода в следующем окне*)

Напишите программу, которая вычисляет приближенное значение производной заданной функции по формуле из предыдущей задачи для n шагов вида

$$h = h_0, h_0/2, h_0/2^2 \dots$$

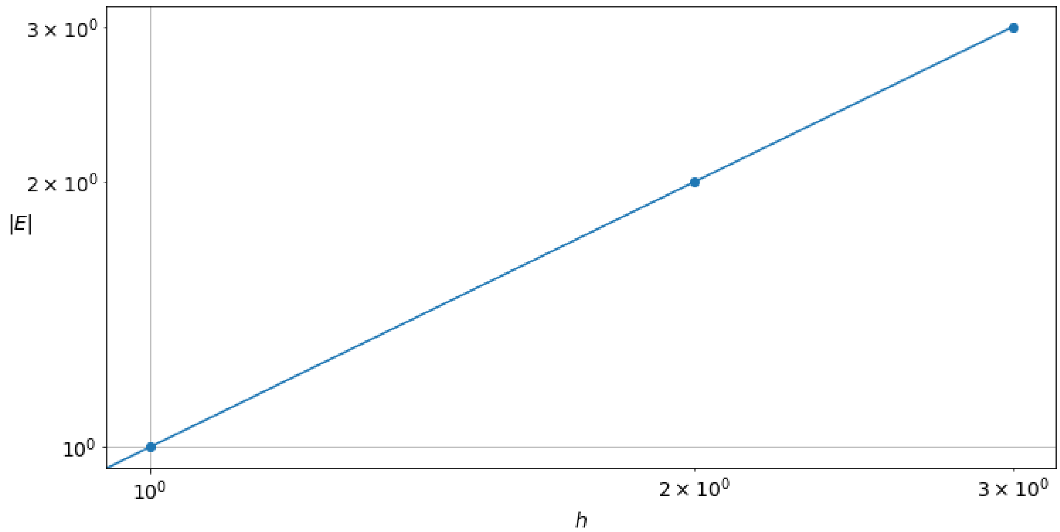
вычисляет ошибку в значении производной для каждого шага и строит график зависимости ошибки от шага в логарифмической шкале.

Запустите программу для функции $\sin(x)$ и точки $x = 0.5$, и функции e^x и точки $x = 1$. Сравните полученный экспериментально оптимальный шаг с теоретической оценкой.

Запустите программу для чисел одинарной точности. Для этого нужно задать массив h[:] типа np.float32
Сравните полученный результат с теоретической оценкой.

In [ ]:

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

# Так можно быстро задать функцию
f = lambda x: np.sin(x)
df = lambda x: np.cos(x)

# число различных шагов
n = 10

h0 = 0.1
# Массив шагов
h = h0 * 2.**-np.arange(n)

'''
 Ваш код здесь
'''

# Так строится график в логарифмической шкале
plt.loglog(np.arange(4), np.arange(4), 'o-')
plt.xlabel('$h$')
plt.ylabel('$|E|$', rotation = 0)
plt.grid(True)
```



In [ ]:

```
# Постройте такой же график для одинарной точности
```

Задание 4. (*используйте заготовку кода в следующем окошке*)

Для суммирования большого количества чисел используется [Алгоритм Кэхэна (https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D1%8D%D1%85%D1%8D%D0%BD%D0](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D1%8D%D1%85%D1%8D%D0%BD%D0)
Этот алгоритм позволяет уменьшить влияние ошибок округления на результат

```
sum = 0
c = 0
for i = 1, N
    y = a[i] - c
    t = sum + y
    c = (t - sum) - y
    sum = t
end for
```

Напишите программу для вычисления суммы элементов массива a с помощью алгоритма Кэхэна и с помощью "наивного" суммирования. Выведите ошибку этих двух методов, и метода np.sum().

Объясните, почему алгоритм Кэхэна имеет маленькую ошибку.

In [ ]:

```
# Вычисление суммы
import numpy as np

# Количество элементов в массиве
n = 10**7

# Создаём три подмассива с разным "масштабом" значений
a = np.ones(n, dtype = np.float32)
a[0:n//4] =  1e-8
a[n//4 : n//2] = 1.
a[n//2 : ] = 1e+2

# Перемешиваем элементы
np.random.seed(1) # Чтобы результат воспроизводился
np.random.shuffle(a)

# Точное значение суммы
sum_ex = 1e-8 * n//4 + n//4 + 1e+2 * n//2

# функция из numpy
sum_np = np.sum(a)

'''
Добавьте вычисление суммы по алгоритму Кэхэна и с помощью простой суммы в цикле
и добавьте ошибку в вывод
Не забудьте все числа привести к формату np.float32
Попробуйте то же самое для двойной точности
'''
sum_naive = np.float32(0.)
#
sum_kah = np.float32(0.)

print('Error in numpy sum = {0:5.3e} '.format(np.abs(sum_ex - sum_np)/sum_ex))

print('Error in naive sum = {0:5.3e} '.format(np.abs(sum_ex - sum_naive)/sum_ex))

print('Error in Kahan sum = {0:5.3e} '.format(np.abs(sum_ex - sum_kah)/sum_ex))
```

```
Error in numpy sum = 3.184e-07
Error in naive sum = 1.000e+00
Error in Kahan sum = 1.000e+00
```

Задание 5

Пусть $\|x\|$ - векторная норма в $\mathbb{C}^n$, $A \in \mathbb{C}^n$ - невырожденная матрица. Докажите, что $\|x\|_* = \|Ax\|$ также является векторной нормой.

Задача 6

Проверьте, является ли выражение $\min(|x_1| + 5|x_2|, 5|x_1| + |x_2|)$ нормой вектора в $\mathbb{R}^2$?