In [490…
```
import numpy as np
import matplotlib.pyplot as plt
from typing import Callable
```

In [491…
```
from IPython.display import Image
Image(filename='screenshot.png')
```

Out[491…

10. Реализовать метод стрельбы для приближенного решения краевой задачи

$$y'' + y = x, \quad y(0) = 0,\ y(1) = 0$$

Соответствующую задачу Коши решить *методом Эйлера с пересчетом* с шагом $h = 0.01$, а параметр η вычислить *методом хорд*. Использовать точность решения $\varepsilon = 0.01$.

In [492…
```
%%latex
Постоновка краевой задачи:
\begin{equation*}
(y)'' + (y) = x \Rightarrow (y)'' = x - y
\end{equation*}
\begin{equation*}
y(0) = 0, y(1) = 0
\end{equation*}
\begin{equation*}
a=0 < x < b=1
\end{equation*}
\begin{equation*}
 A = 0, B = 0
\end{equation*}
```

Постоновка краевой задачи:

$$y'' + y = x \Rightarrow y'' = x - y$$

$$y(0) = 0, y(1) = 0$$

$$a = 0 < x < b = 1$$

$$A = 0, B = 0$$

In [533…
```
%%latex
Сделаем замену и приведем к задаче Коши:
\begin{equation*}
Z = (y)' \Rightarrow (z)' = x - y
\end{equation*}
\begin{equation*}
y(0) = 0, z(0) = (y)'(0) = \eta
\end{equation*}
```

Сделаем замену и приведем к задаче Коши:

$$Z = y' \Rightarrow z' = x - y$$

$$y(0) = 0, z(0) = y'(0) = \eta$$

In [494…
```
a = 0
b = 1
A = 0
B = 0

def f(x, u):
  return x - u[0]
```

In [495…
```
def wrap_func(f):
  return lambda x, u: np.append(u[1:], f(x, u))

def euler_recount(func: Callable, a: float, b: float, u: list, h: float):
  func = wrap_func(f)

  x = a
  u = np.array(u)
  res = [(x, u[0])]

  while x + h <= b:
    u = u + h / 2 * (func(x, u) + func(x + h, u + h * func(x, u)))
    x = x + h
    res.append((x, u[0]))

  return res
```

In [512…
```
"Схему вычисления значений задачи Коши"
eta = 4
np.round(euler_recount((lambda x, u: x - u[0]), a, b, [A, eta], 0.1), 5)
```

Out[512…
```
array([[0.     , 0.     ],
       [0.1    , 0.4    ],
       [0.2    , 0.797  ],
       [0.3    , 1.18802],
       [0.4    , 1.57015],
       [0.5    , 1.94055],
       [0.6    , 2.29652],
       [0.7    , 2.63549],
       [0.8    , 2.95506],
       [0.9    , 3.25303],
       [1.     , 3.52742]])
```

In [497…
```
def err(actual: float, res: np.array):
  return actual - res[-1][1]

err(B, euler_recount(f, a, b, [A, eta], 0.01))
```

Out[497…
```
-3.498105405171457
```

In [498…
```
def new_param(eta_k: float, eta_k_prev: float, err_k: float, err_k_prev: float):
    return eta_k - (err_k * (eta_k - eta_k_prev)) / (err_k - err_k_prev)
```

In [507…
```
def solve(func: Callable, a: float, b: float, A: float, B: float, eta0: float, eta1: float,
          h: float=0.1, eps: float=1e-5):

    result = [[eta0, err(B, euler_recount(func, a, b, [A, eta0], h))],
              [eta1, err(B, euler_recount(func, a, b, [A, eta1], h))]]

    _iter = 1

    while (np.abs(result[_iter][0] - result[_iter-1][0]) > eps):
        print(np.abs(result[_iter][0] - result[_iter-1][0]))
        current_eta = new_param(result[_iter][0], result[_iter - 1][0], result[_iter][1], result[_iter - 1][1
        result.append([current_eta, err(B, euler_recount(f, a, b, [A, current_eta], h))])
        _iter += 1

    return result

def solve_chord(func: Callable, a: float, b: float, A: float, B: float, eta0: float, eta1: float,
          h: float=0.1, eps: float=1e-5):
  eta = [eta_1, eta_2]
  hist = [recount(f, a, b, [A, eta_1], h), recount(f, a, b, [A, eta_2], h)]
  errors = [err(B, hist[0]), err(B, hist[1])]

  while abs(eta[-2] - eta[-1]) > eps:
    eta.append(eta[-1] - (eta[-1] - eta[-2]) * errors[-1] / (errors[-1] - errors[-2]))
    hist.append(recount(f, a, b, [A, eta[-1]], h))
    errors.append(err(B, hist[-1]))

  return hist[-1], hist
```

In [513…
```
%%latex
Выберем первоначальные значения стрельбы, пусть
\begin{equation*}
\eta_1 = -1.3 , \eta_2 = (B - A) / (b - a)
\end{equation*}
```

Выберем первоначальные значения стрельбы, пусть

$$\eta_1 = -1.3, \eta_2 = (B - A)/(b - a)$$

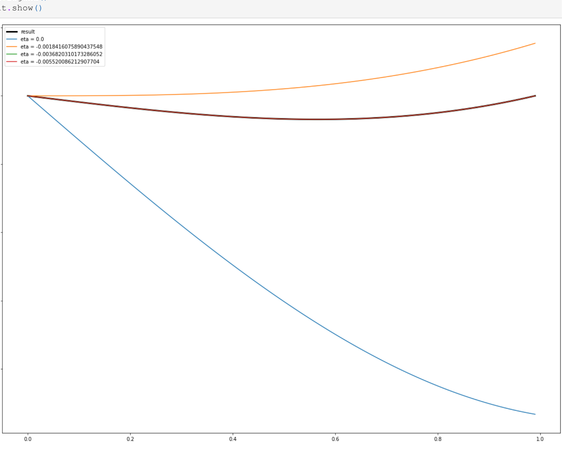
In [508…
```
eta_1 = -1.3
eta_2 = (B - A) / (b - a)

print('eta_1 =', eta_1, '\terr =', err(B, recount(f, a, b, [A, eta_1], 0.1)))
print('eta_2 =', eta_2, '\terr =', err(B, recount(f, a, b, [A, eta_2], 0.1)))
```

```
eta_1 = -1.3    err = 0.9376877082945141
eta_2 = 0.0     err = -0.1575270833502113
```

In [509…
```
res, hist = solve_chord(f, a, b, A, B, eta_1, eta_2, h=0.01, eps=0.01)
```

In [521…
```
arr = np.round(np.array(hist[-1]), 5)
```

In [523…
```
plt.rcParams["figure.figsize"] = (20, 15)

fig, ax = plt.subplots()

x = arr[:, 0]
y = arr[:, 1]
line = ax.plot(x, y, label='result', linewidth=3, color='black')

table = []

for item, eta_res in zip(hist, res):
    table.append(eta_res)
    item = np.round(np.array(item), 5)
    x = item[:, 0]
    y = item[:, 1]
    line = ax.plot(x, y, label=f'eta = {eta_res[1]}')

ax.legend()
plt.show()
```



In [ ]:
```
"""ТАБЛИЦА"""
```

In [532…
```
print(np.round(np.array(table), 10))
```

```
[[ 0.          0.        ]
 [ 0.01       -0.00184161]
 [ 0.02       -0.00368203]
 [ 0.03       -0.00552009]]
```

In [ ]: