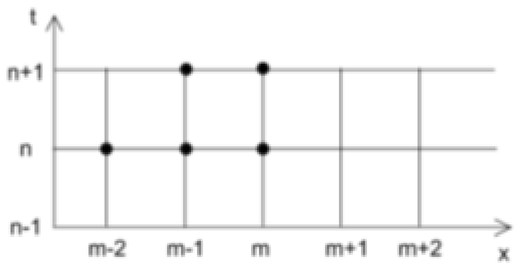
# Нелинейные вычислительные процессы

# Лабораторная работа 1

# Шабалина Алиса

$1)$Теоретическая часть

Необходимо проанализировать схему для решения уравнения переноса $u_t + \lambda u_x = 0, \lambda = 1$ на следующем сеточном шаблоне с $\sigma = 1.25$:




Шаблон № 8

Общий вид схемы: $u_m^{n+1} = \sum\limits_{\mu,\nu} \alpha_\mu^\nu(\tau, h) u_{m+\mu}^{n+\nu}$

Для моего шаблона:$u_n^{m+1} = \alpha_{-2}^0 u_{m-2}^n + \alpha_{-1}^0 u_{m-1}^n + \alpha_0^0 u_m^n + \alpha_0^1 u_m^{n+1}$

Запишем ошибку аппроксимации:

$\delta_r = L_{\tau,h}[u]^{\tau,h} - F_{\tau,h} \sim \alpha^0_{-2} u(x_{m-2}, t^n) + \alpha^0_{-1} u(x_{m-1}, t^n) + \alpha^0_0 u(x_m, t^n) + \alpha^1_{-1} u(x_m, t^{n+1}) - u(x_m, t^{n+1})$

Раскладываем в ряд Тейлора относительно точки $(x_m, t^n)$:

$$\delta_r \sim \alpha^0_{-2}(u - 2hu_x + \frac{4h^2}{2}u_{xx} - \frac{8h^3}{6}u_{xxx} + O(h^4)) + \alpha^0_{-1}(u - hu_x + \frac{h^2}{2}u_{xx} - \frac{h^3}{6}u_{xxx} + O(h^4)) + \alpha^0_0 u$$
$$+ \alpha^1_{-1}(u - hu_x + \tau u_t + \frac{h^2}{2}u_{xx} - \tau h u_{xt} + \frac{\tau^2}{2}u_{tt} - \frac{h^3}{6}u_{xxx} + \frac{h^2\tau}{2}u_{xxt} - \frac{h\tau^2}{2}u_{xtt} + \frac{\tau^3}{6}u_{ttt} + O(\tau^4, h^4))$$
$$- (u + \tau u_t + \frac{\tau^2}{2}u_{tt} + \frac{\tau^3}{6}u_{ttt} + O(\tau^4))$$

К производным только по пространству перейлем с помощью следующих соотношений:

$u_{tt} = -\lambda u_{tx} = \lambda^2 u_{xx}, u_{ttt} = -\lambda u_{ttx} = \lambda^2 u_{txx} = -\lambda^3 u_{xxx};$

$\sigma = \frac{\lambda \tau}{h} = \frac{\tau}{h}$:

$$\delta_r \sim u(\alpha^0_{-2} + \alpha^0_{-1} + \alpha^0_0 + \alpha^1_{-1} - 1) + u_x h(-2\alpha^0_{-2} - \alpha^0_{-1} - \alpha^1_{-1}(1+\sigma) + \sigma) + u_{xx}\frac{h^2}{2}(4\alpha^0_{-2} + \alpha^0_{-1} + \alpha^1_{-1}(1 + 2\sigma + \sigma^2) - \sigma^2)$$
$$+ u_{xxx}\frac{h^3}{6}(-8\alpha^0_{-2} - \alpha^0_{-1} - \alpha^1_{-1}(1 + 3\sigma + 3\sigma^2 + \sigma^3) + \sigma^3) + O(\tau^4, h^4)$$

При фиксированных $\sigma$ и $\lambda$ $\tau \sim h$, то $O(\tau^4, h^4) = O(h^4)$

Из соображений размерности ошибку надо разделить на $\tau$

$$\delta_0 = \alpha^0_{-2} + \alpha^0_{-1} + \alpha^0_0 + \alpha^1_{-1} - 1$$
$$\delta_1 = -2\alpha^0_{-2} - \alpha^0_{-1} - \alpha^1_{-1}(1+\sigma) + \sigma$$
$$\delta_2 = 4\alpha^0_{-2} + \alpha^0_{-1} + \alpha^1_{-1}(1+2\sigma+\sigma^2) - \sigma^2$$
$$\delta_3 = -8\alpha^0_{-2} - \alpha^0_{-1} - \alpha^1_{-1}(1+3\sigma+3\sigma^2+\sigma^3) + \sigma^3$$

Полная ошибка:

$$\delta_r = \frac{\delta_0}{\tau}u + \frac{h\delta_1}{\tau}u_x + \frac{h^2\delta_2}{2\tau}u_{xx} + \frac{h^3\delta_3}{6\tau}u_{xxx} + O(h^3)$$

Для аппроксимации уравнения надо $\delta_0 = 0$

Для аппроксимации первого порядка надо $\delta_0 = 0; \delta_1 = 0$

Для аппроксимации второго порядка надо $\delta_0 = 0; \delta_1 = 0; \delta_2 = 0$

Для аппроксимации третьего порядка надо $\delta_0 = 0; \delta_1 = 0; \delta_2 = 0; \delta_3 = 0$

(1т)

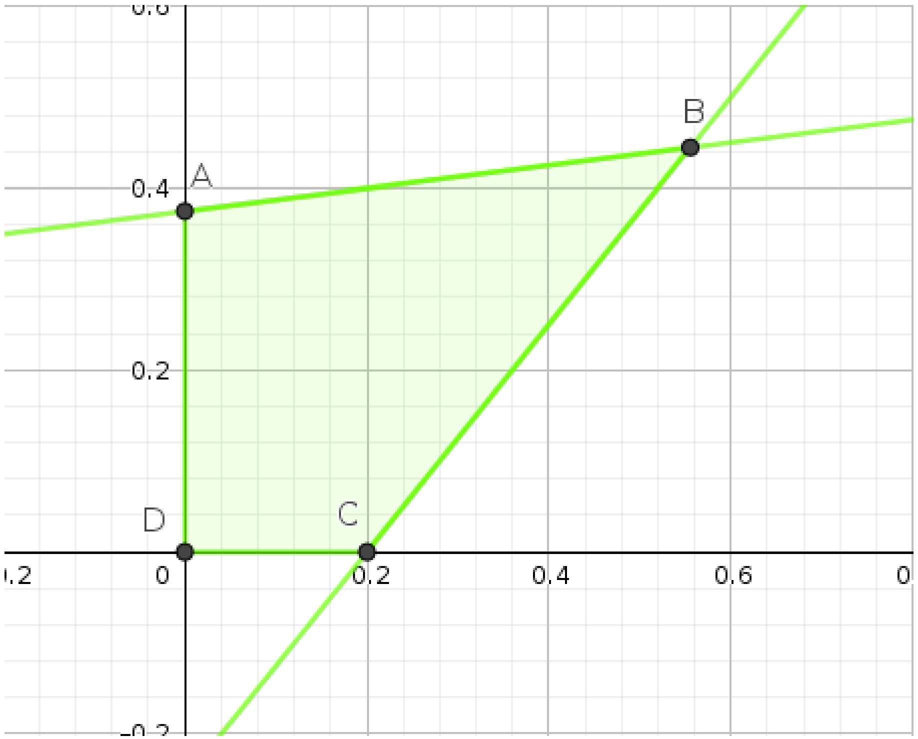
Первый порядок с двумя свободными параметрами $\alpha^0_0$ и $\alpha^1_{-1}$:

$$\begin{cases} \alpha^0_{-2} = \alpha^0_0 - \frac{5}{4}\alpha^1_{-1} + \frac{1}{4} \\ \alpha^0_{-1} = -2\alpha^0_0 + \frac{1}{4}\alpha^1_{-1} + \frac{3}{4} \end{cases}$$

Множество монотонных по Фридрихсу схем: $\begin{cases} \alpha^0_{-2} = \alpha^0_0 - \frac{5}{4}\alpha^1_{-1} + \frac{1}{4} \geq 0 \\ \alpha^0_{-1} = -2\alpha^0_0 + \frac{1}{4}\alpha^1_{-1} + \frac{3}{4} \geq 0 \\ \alpha^0_0 \geq 0 \\ \alpha^1_{-1} \geq 0 \end{cases}$

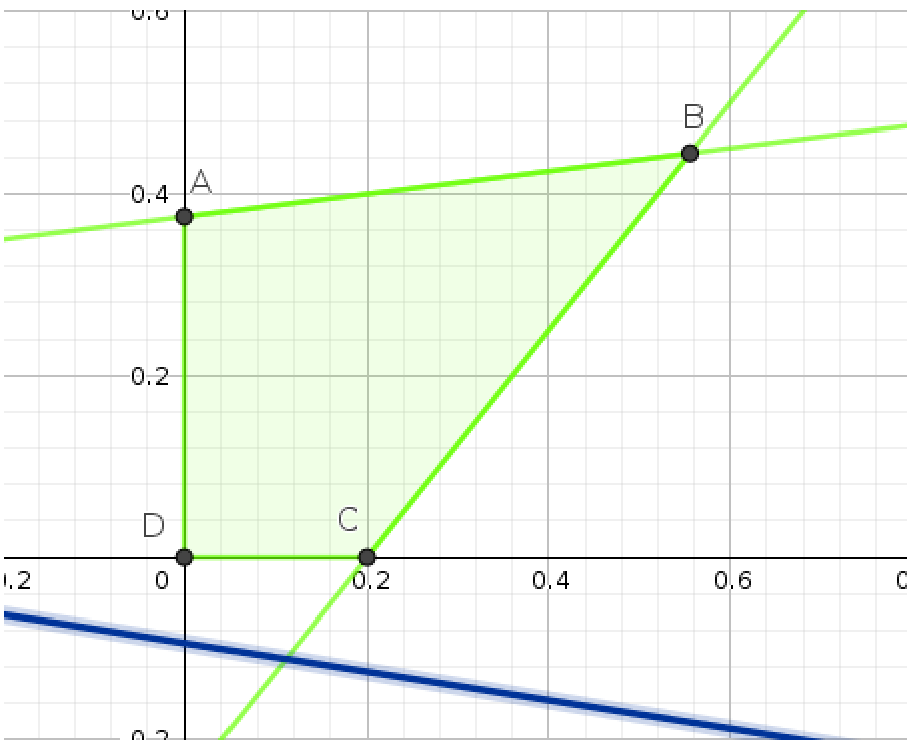
$$\begin{cases} \alpha_0^0 \geq \frac{5}{4}\alpha_{-1}^1 - \frac{1}{4} \\ \alpha_0^0 \leq \frac{1}{8}\alpha_{-1}^1 + \frac{3}{8} \\ \alpha_0^0 \geq 0 \\ \alpha_{-1}^1 \geq 0 \end{cases}$$

Будем строить в осях $(\alpha_{-1}^1; \alpha_0^0)$




(2т)

Аппроксимация второго порядка дает нам уравнение: $\alpha_0^0 = -\frac{5}{32}\alpha_{-1}^1 - \frac{3}{32}$

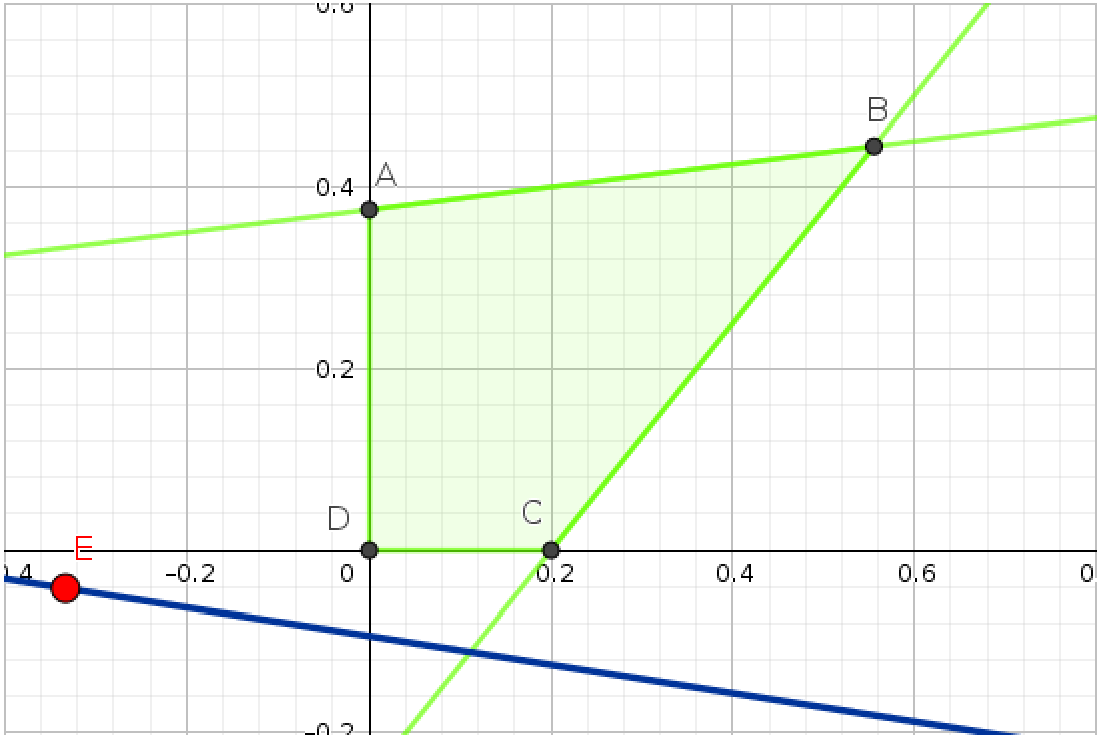



(3т)

Схема третьего порядка аппроксимации это есть точка $\left(\frac{5}{8}; \frac{3}{4}; -\frac{1}{24}; -\frac{1}{3}\right)$

Аналитический вид: $u_m^{n+1} = \frac{5}{8}u_{m-2}^n + \frac{3}{4}u_{m-1}^n - \frac{1}{24}u_m^n - \frac{1}{3}u_{m-1}^{n+1}$

На рисунке в наших осях точка $E = \left(-\frac{1}{3}; -\frac{1}{24}\right)$




(4т)

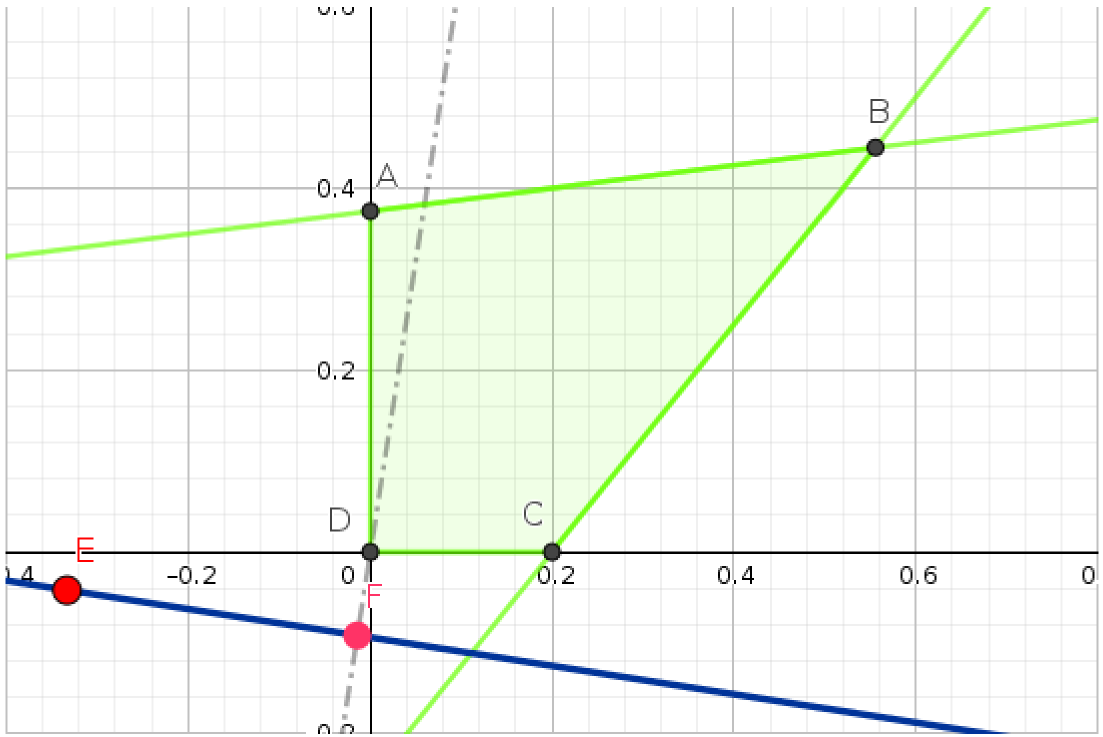
Аналитический вид схемы с "минимальной вязкостью"

$$u_m^{n+1} = \alpha_{-2}^0 u_{m-2}^n + \alpha_{-1}^0 u_{m-1}^n + 0 * u_m^n + 0 * u_m^{n+1} \rightarrow$$
$$\rightarrow u_m^{n+1} = \frac{1}{4} u_{m-2}^n + \frac{3}{4} u_{m-1}^n$$

(5т)

Схема 2-го порядка наиболее близкая ко множеству положительных по Фридрихсу схем отвечают точке $F = (-\frac{15}{1049}; -\frac{96}{1049})$, которая является основанием перпендикуляра из точки D

Аналтический вид: $u_m^{n+1} = \frac{185}{1049}u_{m-2}^n + \frac{975}{1049}u_{m-1}^n - \frac{96}{1049}u_m^n - \frac{15}{1049}u_{m-1}^{n+1}$




1)Практическая часть

Решим краевую задачу:

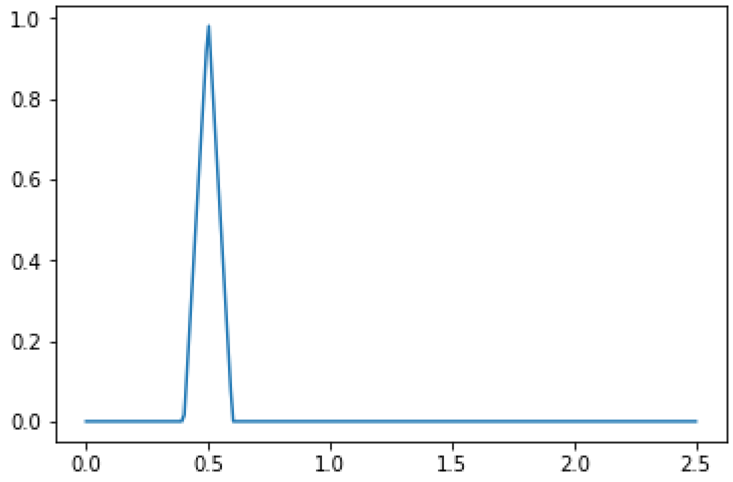
$$\begin{cases} u_t + \lambda u_x = 0,\ \lambda = 1\ (t > 0, 0 < x \leq X, X = 2), \\ u(0, x) = \phi(x)\ (0 \leq x \leq X), \\ u(t, 0) = 0\ (0 < t \leq 100\tau), \end{cases}$$

где функция $\phi(x)$ определяется способом (в) "треугольник":

$$\begin{cases} 10x - 4\ if\ 0.4 \leq x \leq 0.5, \\ -10x + 6\ if\ 0.5 \leq x \leq 0.6, \\ 0\ else. \end{cases}$$

на сетке с числом узлов 201 ($h = 0.01$) для заданного сеточного шаблона(см. начало документа) и указанного значения Куранта($\sigma = 1.25$), шаг по времени определим по числу Куранта: $\sigma = \frac{\lambda\tau}{h} \to \tau = \frac{\sigma h}{\lambda} = \frac{1.25*0.01}{1} = 0.0125$

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt

area_steps = 250
area_step = 2.5 / area_steps
time_step = 1.25 * area_step
time_steps = 100

def phi(x):
    variable = 0.0
    if ((x >= 0.4) and (x <= 0.5)):
        variable = 10 * x - 4
    if ((x >= 0.5) and (x <= 0.6)):
        variable = -10 * x + 6
    return variable

x_0 = np.linspace(0, 2.5, area_steps)
y_0 = np.zeros(area_steps)
for i in range(0,area_steps):
    y_1[i] = phi(x_1[i])
plt.plot(x_1, y_1)

plt.show()
```



In [2]:
```
u = np.zeros(area_steps)

for i in range(0, area_steps):
    u[i] = phi(i * area_step)

print(u)
```

```
[0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  0.9 0.8 0.7
 0.6 0.5 0.4 0.3 0.2 0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
```

In [3]:
```
def computation(alpha1, alpha2, alpha3, alpha4, u, step):
    prev = np.zeros(area_steps)
    curr = np.zeros(area_steps)
    curr = u.copy()
    new = np.zeros(area_steps)
    curr[0] = 0.0
    prev[0] = 0.0
    new[0] = 0.0
    for i in range(1, area_steps):
        prev[i] = phi(i * area_step)
        curr[i] = phi(i * area_step - time_step)

    for j in range(1, step):
        new[0] = 0.0
        new[1] = 0.0
        for i in range(2, area_steps):
            new[i] = alpha1*curr[i-2] + alpha2*curr[i-1] + alpha3*curr[i] + alpha4*new[i-1]
        prev = curr.copy()
        curr = new.copy()

    return curr
```

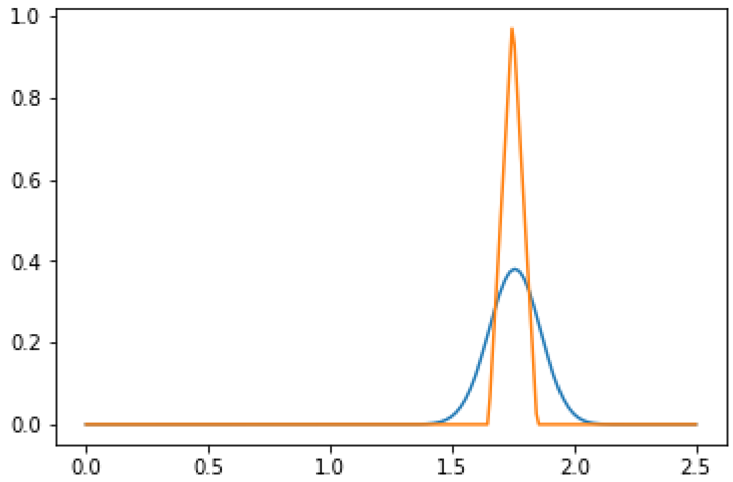
In [4]:
```
def draw_computation(alpha1, alpha2, alpha3, alpha4, u, step):
    x_1 = np.linspace(0, 2.5, area_steps)
    y_1 = computation(alpha1, alpha2, alpha3, alpha4, u, step)
    plt.plot(x_1, y_1)
    x_2 = np.linspace(0, 2.5, area_steps)
    y_2 = np.zeros(area_steps)
    for i in range(0, area_steps):
        y_2[i] = phi(x_1[i] - step * time_step)
    plt.plot(x_2, y_2)

    plt.show()
```

Построим графики в точках A, B, C, D, E и F

(1п)

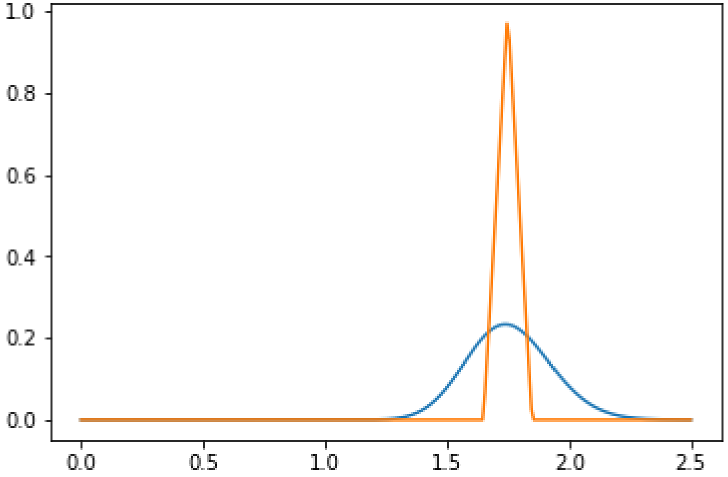
$A(0; \frac{3}{8}) \to u_m^{n+1} = \frac{5}{8} u_{m-2}^n + \frac{3}{8} u_m^n$

In [5]:
```
draw_computation(5.0/8, 0, 3.0/8, 0, u, time_steps)
```



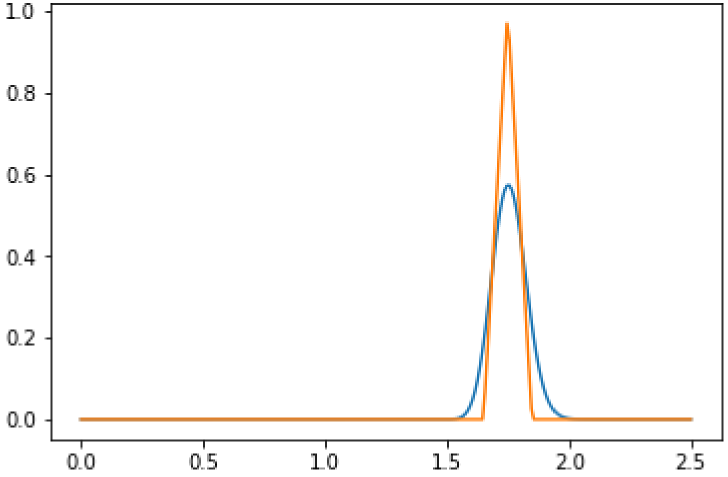

$B(\frac{5}{9}; \frac{4}{9}) \to u_m^{n+1} = \frac{4}{9} u_m^n + \frac{5}{9} u_{m-1}^{n+1}$

In [6]:
```
draw_computation(0.0, 0.0, 4.0/9, 5.0/9, u, time_steps)
```



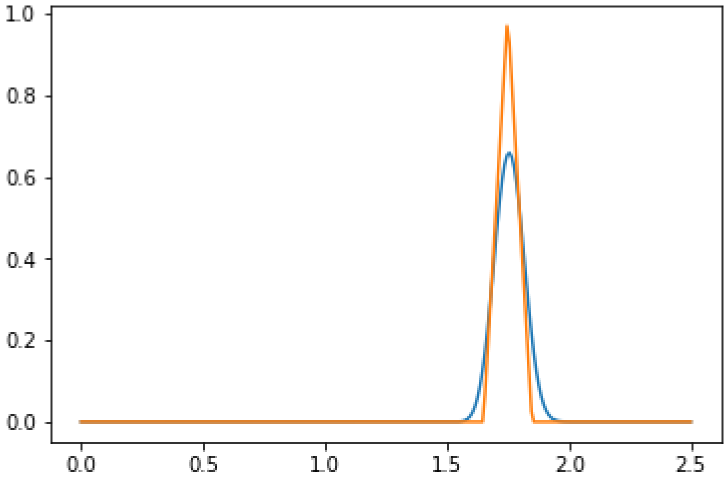

$C(\frac{1}{5};0) \rightarrow u_m^{n+1} = \frac{4}{5}u_{m-1}^n + \frac{1}{5}u_{m-1}^{n+1}$

In [7]:
```
draw_computation(0.0, 4.0/5, 0, 1.0/5, u, time_steps)
```



$D(0;0) \rightarrow u_m^{n+1} = \frac{1}{4}u_{m-2}^n + \frac{3}{4}u_{m-1}^n$

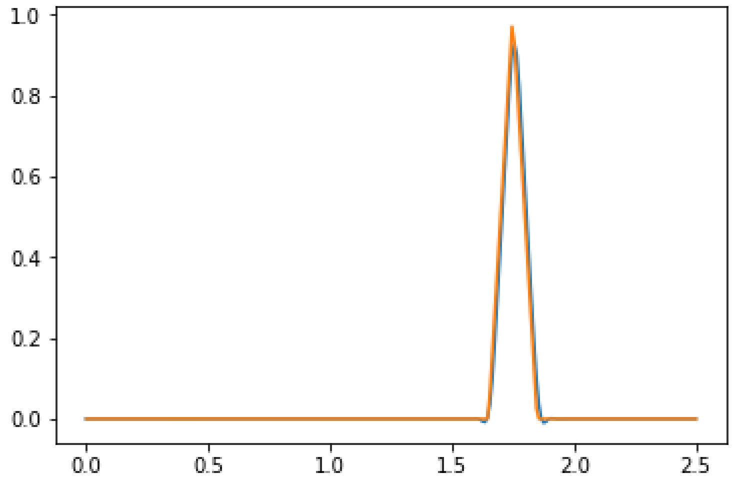
In [8]:
```
draw_computation(1.0/4, 3.0/4, 0, 0, u, time_steps)
```




(4п)

$E(-\frac{1}{3};-\frac{1}{24}) \rightarrow u_m^{n+1} = \frac{5}{8}u_{m-2}^n + \frac{3}{4}u_{m-1}^n - \frac{1}{24}u_m^n - \frac{1}{3}u_{m-1}^{n+1}$

In [9]:
```
draw_computation(5.0/8, 3.0/4, -1.0/24, -1.0/3, u, time_steps)
```

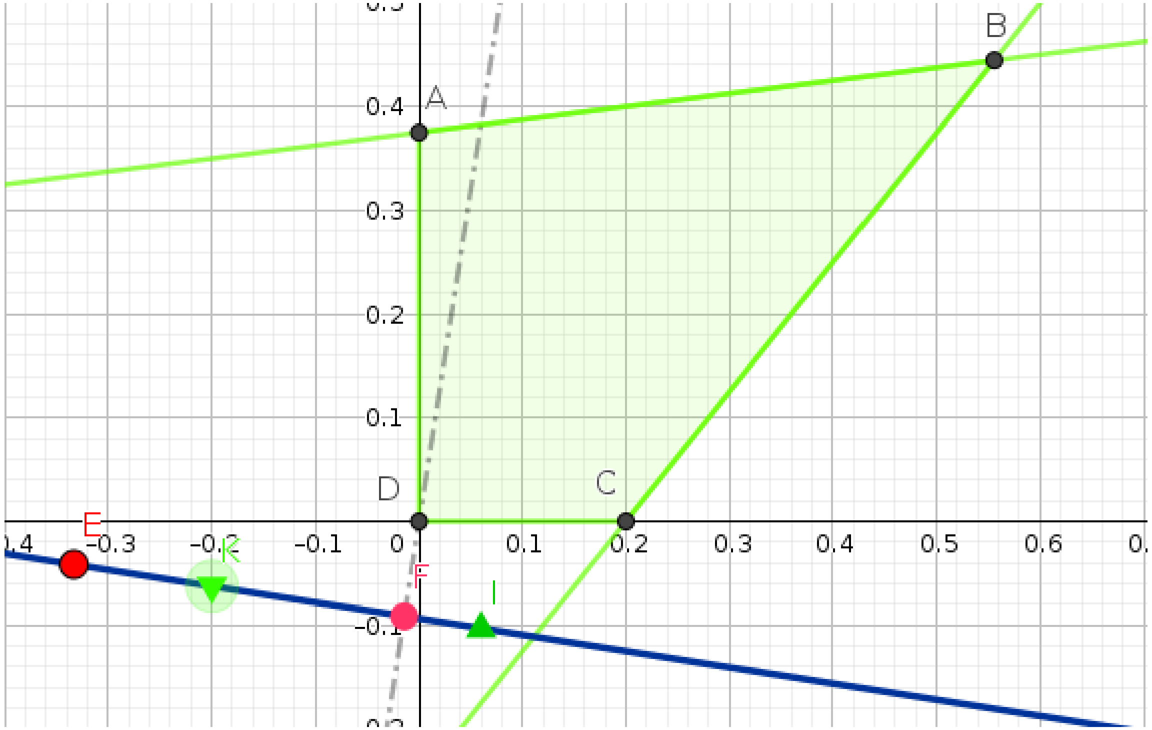



(3п)

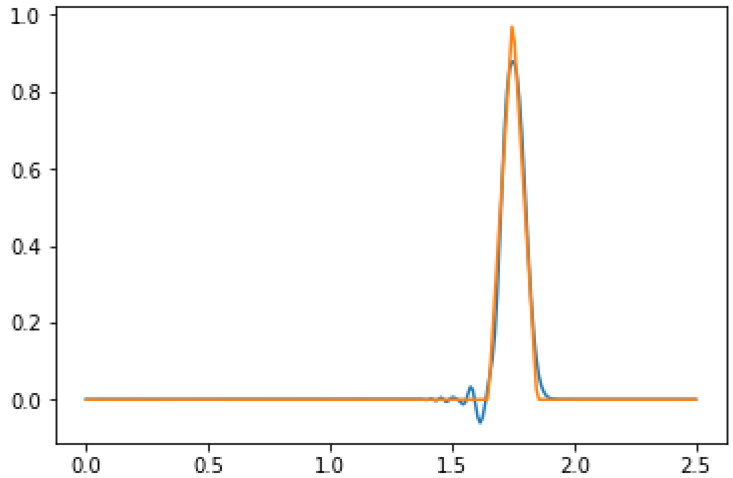
$$I(\frac{3}{50}; -\frac{33}{320}) \rightarrow u_m^{n+1} = \frac{23}{320}u_{m-2}^n + \frac{777}{800}u_{m-1}^n - \frac{33}{320}u_m^n + \frac{3}{50}u_{m-1}^{n+1}$$

и

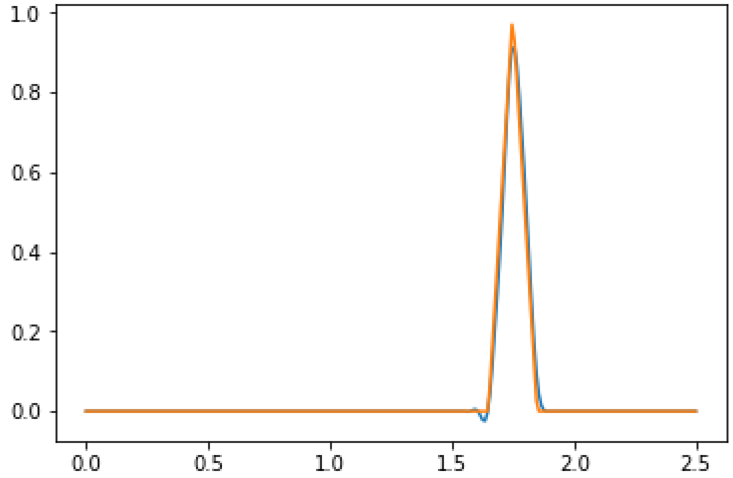
$$K(-\frac{1}{5}; -\frac{1}{16}) \rightarrow u_m^{n+1} = \frac{7}{16}u_{m-2}^n + \frac{33}{40}u_{m-1}^n - \frac{1}{16}u_m^n - \frac{1}{5}u_{m-1}^{n+1}$$


B
A
0.4
0.3
0.2
0.1
D
C
E
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
K
F
I
-0.1

In [23]:
```
draw_computation(23.0/320, 777.0/800, -33.0/320, 3.0/50, u, time_steps)
```



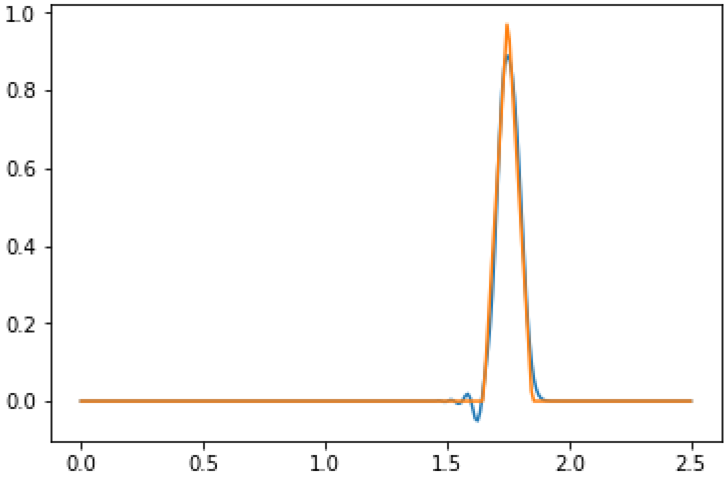

In [11]:
```
draw_computation(7.0/16, 33.0/40, -1.0/16, -1.0/5, u, time_steps)
```




(2п)

$F = (-\frac{15}{1049}; -\frac{96}{1049}) \to u_m^{n+1} = \frac{185}{1049}u_{m-2}^n + \frac{975}{1049}u_{m-1}^n - \frac{96}{1049}u_m^n - \frac{15}{1049}u_{m-1}^{n+1}$

In [12]:
```
draw_computation(185.0/1049, 975.0/1049, -96.0/1049, -15.0/1049, u, time_steps)
```



In [19]:
```
def computation_1(alpha1, alpha2, alpha3, alpha4, beta1, beta2, beta3, beta4, u, step):
    prev = np.zeros(area_steps)
    curr = np.zeros(area_steps)
    curr = u.copy()
    new = np.zeros(area_steps)
    curr[0] = 0.0
    prev[0] = 0.0
    new[0] = 0.0
    for i in range(1, area_steps):
        prev[i] = phi(i * area_step)
        curr[i] = phi(i * area_step - time_step)

    for j in range(1, step):
        new[0] = 0.0
        new[1] = 0.0
        for i in range(2, area_steps):
            test = alpha1*curr[i-2] + alpha2*curr[i-1] + alpha3*curr[i] + alpha4*new[i-1]
            if ((min(curr[i-2],curr[i-1]) <= test) and (max(curr[i-2],curr[i-1]) >= test)):
                new[i] = test
            else:
                new[i] = beta1*curr[i-2] + beta2*curr[i-1] + beta3*curr[i] + beta4*new[i-1]
        prev = curr.copy()
        curr = new.copy()
    return curr
```

In [14]:
```
def draw_computation_1(alpha1, alpha2, alpha3, alpha4, beta1, beta2, beta3, beta4, u, step):
    x_1 = np.linspace(0, 2.5, area_steps)
    y_1 = computation_1(alpha1, alpha2, alpha3, alpha4, beta1, beta2, beta3, beta4, u, step)
    plt.plot(x_1, y_1)
    x_2 = np.linspace(0, 2.5, area_steps)
    y_2 = np.zeros(area_steps)
    for i in range(0, area_steps):
        y_2[i] = phi(x_1[i] - step * time_step)
    plt.plot(x_2, y_2)

    plt.show()
```

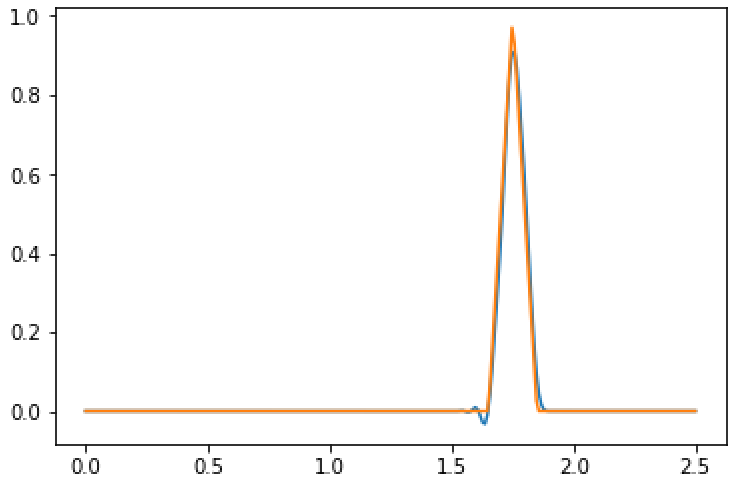
(5п)

$$I(\frac{3}{50}; -\frac{33}{320}) \rightarrow u_m^{n+1} = \frac{23}{320}u_{m-2}^n + \frac{777}{800}u_{m-1}^n - \frac{33}{320}u_m^n + \frac{3}{50}u_{m-1}^{n+1}$$

и

$$K(-\frac{1}{5}; -\frac{1}{16}) \rightarrow u_m^{n+1} = \frac{7}{16}u_{m-2}^n + \frac{33}{40}u_{m-1}^n - \frac{1}{16}u_m^n - \frac{1}{5}u_{m-1}^{n+1}$$

In [24]:
```
draw_computation_1(7.0/16, 33.0/40, -1.0/16, -1.0/5, 23.0/320, 777.0/800, -33.0/320, 3.0/50, u, time_step
s)
```



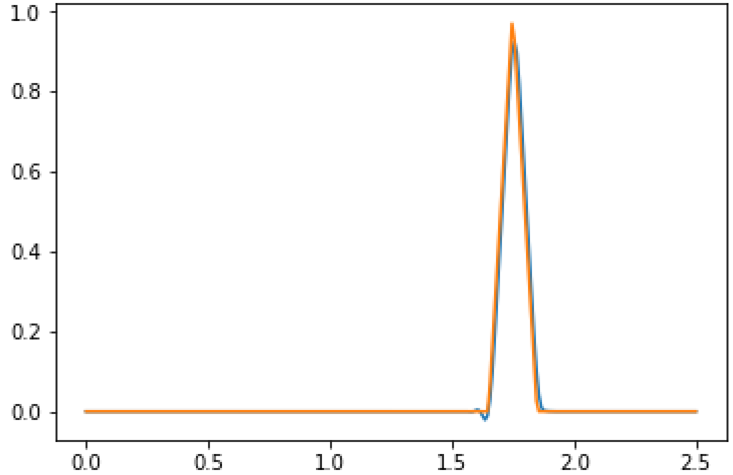

(6п)

$$E(-\frac{1}{3}; -\frac{1}{24}) \rightarrow u_m^{n+1} = \frac{5}{8}u_{m-2}^n + \frac{3}{4}u_{m-1}^n - \frac{1}{24}u_m^n - \frac{1}{3}u_{m-1}^{n+1}$$

и

$$I(\frac{3}{50}; -\frac{33}{320}) \rightarrow u_m^{n+1} = \frac{23}{320}u_{m-2}^n + \frac{777}{800}u_{m-1}^n - \frac{33}{320}u_m^n + \frac{3}{50}u_{m-1}^{n+1}$$

In [17]:
```
draw_computation_1(5.0/8, 3.0/4, -1.0/24, -1.0/3, 23.0/320, 777.0/800, -33.0/320, 3.0/50, u, time_steps)
```



In [21]:

```
def computation_2(alpha1, alpha2, alpha3, alpha4, beta1, beta2, beta3, beta4, gamma1, gamma2, gamma3, ga
mma4, u, step):
    prev = np.zeros(area_steps)
    curr = np.zeros(area_steps)
    curr = u.copy()
    new = np.zeros(area_steps)
    curr[0] = 0.0
    prev[0] = 0.0
    new[0] = 0.0
    for i in range(1, area_steps):
        prev[i] = phi(i * area_step)
        curr[i] = phi(i * area_step - time_step)

    for j in range(1, step):
        new[0] = 0.0
        new[1] = 0.0
        for i in range(2, area_steps):
            test_1 = alpha1 * curr[i-2] + alpha2 * curr[i-1] + alpha3 * curr[i] + alpha4 * new[i-1]
            if ((min(curr[i-2], curr[i-1]) <= test_1) and (max(curr[i-2], curr[i-1]) >= test_1)):
                new[i] = test_1
            else:
                test_2 = beta1 * curr[i-2] + beta2 * curr[i-1] + beta3 * curr[i] + beta4 * new[i-1]
                if ((min(curr[i-2], curr[i-1]) <= test_2) and (max(curr[i-2], curr[i-1]) >= test_2)):
                    new[i] = test_2
                else:
                    new[i] = beta1*curr[i-2] + beta2*curr[i-1] + beta3*curr[i] + beta4*new[i-1]
        prev = curr.copy()
        curr = new.copy()
    return curr
```

In [29]:
```
def draw_computation_2(alpha1, alpha2, alpha3, alpha4, beta1, beta2, beta3, beta4, gamma1, gamma2, gamma3
, gamma4, u, step):
    x_1 = np.linspace(0, 2.5, area_steps)
    y_1 = computation_2(alpha1, alpha2, alpha3, alpha4, beta1, beta2, beta3, beta4, gamma1, gamma2, gamma
3, gamma4, u, step)
    plt.plot(x_1, y_1)
    x_2 = np.linspace(0, 2.5, area_steps)
    y_2 = np.zeros(area_steps)
    for i in range(0, area_steps):
        y_2[i] = phi(x_1[i] - step * time_step)
    plt.plot(x_2, y_2)

    plt.show()
```

(7п)

$$K(-\frac{1}{5}; -\frac{1}{16}) \rightarrow u_m^{n+1} = \frac{7}{16}u_{m-2}^n + \frac{33}{40}u_{m-1}^n - \frac{1}{16}u_m^n - \frac{1}{5}u_{m-1}^{n+1}$$

и

$$I(\frac{3}{50}; -\frac{33}{320}) \rightarrow u_m^{n+1} = \frac{23}{320}u_{m-2}^n + \frac{777}{800}u_{m-1}^n - \frac{33}{320}u_m^n + \frac{3}{50}u_{m-1}^{n+1}$$

и

$$E(-\frac{1}{3}; -\frac{1}{24}) \rightarrow u_m^{n+1} = \frac{5}{8}u_{m-2}^n + \frac{3}{4}u_{m-1}^n - \frac{1}{24}u_m^n - \frac{1}{3}u_{m-1}^{n+1}$$

In [30]:
```
draw_computation_2(7.0/16, 33.0/40, -1.0/16, -1.0/5, 23.0/320, 777.0/800, -33.0/320, 3.0/50, 5.0/8, 3.0/4
, -1.0/24, -1.0/3, u, time_steps)
```