# Курсовая работа по курсу "Математическое моделирование и вычислительный эксперимент"

**Выполнил: Садаков А. А.**

*Группа: М8О-406Б-19*

Значения $\rho_1, u_1, v_1, \varepsilon_1$ выражаются из соотношений Гюгонио.

$$\rho_1 v_1 - \rho_2 v_2 = D(\rho_1 - \rho_2),$$

$$\rho_1 v_1^2 + p_1 - \rho_2 v_2^2 - p_2 = D(\rho_1 v_1 - \rho_2 v_2),$$

$$v_1\left(\rho_1 \varepsilon_1 + \rho_1 \frac{v_1^2}{2} + p_1\right) - v_2\left(\rho_2 \varepsilon_2 + \rho_2 \frac{v_2^2}{2} + p_2\right) = D\left(\rho_1 \varepsilon_1 + \rho_1 \frac{v_1^2}{2} - \rho_2 \varepsilon_2 - \rho_2 \frac{v_2^2}{2}\right)$$

Скорость движения ударной волны равна $D = c_2 M, c_2 = \sqrt{\gamma(\gamma - 1)\varepsilon_2}$.

Если подставить $v_2 = 0$ в соотношения, то они значительно упрощаются:

$$\rho_1 v_1 = D(\rho_1 - \rho_2),$$

$$\rho_1 v_1^2 + p_1 - p_2 = D \rho_1 v_1,$$

$$v_1\left(\rho_1 \varepsilon_1 + \rho_1 \frac{v_1^2}{2} + p_1\right) = D\left(\rho_1 \varepsilon_1 + \rho_1 \frac{v_1^2}{2} - \rho_2 \varepsilon_2\right)$$

Выразим $\rho_1, v_1, \varepsilon_1$ из данных соотношений. Выразим $\rho_1$ через $v_1$:

$$\rho_1 v_1 = D(\rho_1 - \rho_2)$$

$$\rho_1 v_1 = D \rho_1 - D \rho_2$$

$$D \rho_2 = D \rho_1 - v_1 \rho_1$$

$$(D - v_1)\rho_1 = D \rho_2$$

$$\rho_1 = \frac{D \rho_2}{D - v_1}$$

Выразим $\varepsilon_1$ через $v_1$:

$$\rho_1 v_1^2 + p_1 - p_2 = D \rho_1 v_1$$

$$\rho_1 v_1^2 + (\gamma - 1)\rho_1 \varepsilon_1 - (\gamma - 1)\rho_2 \varepsilon_2 = D \rho_1 v_1$$

$$(\gamma - 1)\rho_1 \varepsilon_1 = D \rho_1 v_1 - \rho_1 v_1^2 + (\gamma - 1)\rho_2 \varepsilon_2$$

$$(\gamma-1)\rho_1\varepsilon_1=(D-v_1)\rho_1 v_1+(\gamma-1)\rho_2\varepsilon_2$$

Подставим выражение $\rho_1$:

$$\frac{(\gamma-1)D\rho_2}{D-v_1}\varepsilon_1=D\rho_2 v_1+(\gamma-1)\rho_2\varepsilon_2$$

$$\frac{\gamma-1}{D-v_1}\varepsilon_1=v_1+\frac{\gamma-1}{D}\varepsilon_2$$

Переобозначим $B=\frac{\gamma-1}{D}\varepsilon_2$. Это упростит вычисления в дальнейшем.

$$\frac{\gamma-1}{D-v_1}\varepsilon_1=v_1+B$$

$$\varepsilon_1=\frac{(D-v_1)(v_1+B)}{\gamma-1}$$

Найдём значение $v_1$ из третьего соотношения:

$$v_1\left(\rho_1\varepsilon_1+\rho_1\frac{v_1^2}{2}+p_1\right)=D\left(\rho_1\varepsilon_1+\rho_1\frac{v_1^2}{2}-\rho_2\varepsilon_2\right)$$

$$v_1\left(\rho_1\varepsilon_1+\rho_1\frac{v_1^2}{2}+(\gamma-1)\rho_1\varepsilon_1\right)=D\left(\rho_1\varepsilon_1+\rho_1\frac{v_1^2}{2}-\rho_2\varepsilon_2\right)$$

$$v_1\left(\rho_1\frac{v_1^2}{2}+\gamma\rho_1\varepsilon_1\right)=D\left(\rho_1\varepsilon_1+\rho_1\frac{v_1^2}{2}-\rho_2\varepsilon_2\right)$$

Подставим $\rho_1$ и $\varepsilon_1$:

$$\rho_1\varepsilon_1=\frac{D\rho_2}{D-v_1}\frac{(D-v_1)(v_1+B)}{\gamma-1}=\frac{D\rho_2(v_1+B)}{\gamma-1}$$

$$v_1\left(\frac{D\rho_2 v_1^2}{2(D-v_1)}+\frac{\gamma D\rho_2(v_1+B)}{\gamma-1}\right)=D\left(\frac{D\rho_2(v_1+B)}{\gamma-1}+\frac{D\rho_2 v_1^2}{2(D-v_1)}-\rho_2\varepsilon_2\right)$$

$$\frac{D\rho_2 v_1^3}{2(D-v_1)}+\frac{\gamma D v_1\rho_2(v_1+B)}{\gamma-1}=\frac{D^2\rho_2(v_1+B)}{\gamma-1}+\frac{D^2\rho_2 v_1^2}{2(D-v_1)}-D\rho_2\varepsilon_2$$

$$\frac{\gamma D\rho_2 v_1^2}{\gamma-1}+\frac{\gamma B D\rho_2 v_1}{\gamma-1}+D\rho_2\varepsilon_2=\frac{D^2\rho_2 v_1}{\gamma-1}+\frac{\frac{(\gamma-1)\varepsilon_2}{D}D^2\rho_2}{\gamma-1}+D\frac{D\rho_2 v_1^2}{2(D-v_1)}-v_1\frac{D\rho_2 v_1^2}{2(D-v_1)}$$

$$\frac{\gamma D \rho_2 v_1^2}{\gamma-1}+\frac{\gamma \frac{(\gamma-1)\varepsilon_2}{D} D \rho_2 v_1}{\gamma-1}-\frac{D^2 \rho_2 v_1}{\gamma-1}+D \rho_2 \varepsilon_2=D \rho_2 \varepsilon_2+\frac{D(D-v_1)\rho_2 v_1^2}{2(D-v_1)}$$

$$\frac{\gamma D \rho_2 v_1^2}{\gamma-1}+\gamma \rho_2 \varepsilon_2 v_1-\frac{D^2 \rho_2 v_1}{\gamma-1}=\frac{D \rho_2 v_1^2}{2}$$

$$\left(\left(\frac{\gamma D}{\gamma-1}-\frac{D}{2}\right) v_1+\gamma \varepsilon_2-\frac{D^2}{\gamma-1}\right) v_1=0$$

Если $v_1=0$, то $\rho_1=\frac{D \rho_2}{D-v_1}=\rho_2$. Для устойчивости ударной волны необходимо $\rho_1>\rho_2$, потому данное решение не подходит. Следовательно, $v_1 \neq 0$.

$$\left(\frac{\gamma D}{\gamma-1}-\frac{D}{2}\right) v_1+\gamma \varepsilon_2-\frac{D^2}{\gamma-1}=0$$

$$\left(\frac{\gamma D}{\gamma-1}-\frac{D}{2}\right) v_1=\frac{D^2}{\gamma-1}-\gamma \varepsilon_2$$

$$\left(\gamma D-\frac{D(\gamma-1)}{2}\right) v_1=D^2-\gamma(\gamma-1)\varepsilon_2$$

$$(2\gamma D-\gamma D+D) v_1=2 D^2-2\gamma(\gamma-1)\varepsilon_2$$

$$D(\gamma+1) v_1=2 D^2-2\gamma(\gamma-1)\varepsilon_2$$

$$v_1=\frac{2\left(D^2-\gamma(\gamma-1)\varepsilon_2\right)}{D(\gamma+1)}$$

Проверим условие $\rho_1>\rho_2$:

$$\frac{D \rho_2}{D-v_1}>\rho_2$$

$$\frac{D}{D-v_1}>1$$

Чтобы дробь была положительной, необходимо $v_1<D$.

$$D>D-v_1$$

$$0>-v_1$$

$$v_1>0$$

Итого, $0<v_1<D$. Проверим, что полученное $v_1$ удовлетворяет данным ограничениям.

$$\frac{2\left(D^2-\gamma(\gamma-1)\varepsilon_2\right)}{D(\gamma+1)}<D$$

$$2\left(D^2-\gamma(\gamma-1)\varepsilon_2\right)<D^2(\gamma+1)$$

$$2c_2^2M^2-2\gamma(\gamma-1)\varepsilon_2<c_2^2M^2(\gamma+1)$$

$$2\gamma(\gamma-1)\varepsilon_2M^2-2\gamma(\gamma-1)\varepsilon_2<\gamma(\gamma-1)(\gamma+1)\varepsilon_2M^2$$

$$2M^2-2<(\gamma+1)M^2$$

$$(\gamma+1)M^2-2M^2>-2$$

$$(\gamma-1)M^2>-2$$

Показатель адиабаты $\gamma>1$, потому выражение в левой части положительно. $v_1<D$ выполняется.

$$\frac{2\left(D^2-\gamma(\gamma-1)\varepsilon_2\right)}{D(\gamma+1)}>0$$

$$D^2-\gamma(\gamma-1)\varepsilon_2>0$$

$$c_2^2M^2-\gamma(\gamma-1)\varepsilon_2>0$$

$$\gamma(\gamma-1)\varepsilon_2M^2-\gamma(\gamma-1)\varepsilon_2>0$$

$$M^2-1>0$$

$$M^2>1$$

Число Маха $M>1$, потому неравенство выполняется. $v_1>0$ выполняется. Значит, найденное значение $v_1$ удовлетворяет условиям устойчивости. Зная значение $v_1$, можно вычислить значения $\rho_1$ и $\varepsilon_1$ по формулам, выписанным ранее.

Построим сетку на области интегрирования. Сетка выбирается так, чтобы число разбиений по оси Y было кратно десяти. Таким образом, ударная волна проходит по границе сетки. Значения гидродинамических величин в ячейках вычисляются следующим образом:

1) Если ордината центра ячейки больше 1.5, то задаём значения $\rho=\rho_1, u=0, v=v_1, \varepsilon=\varepsilon_1$.

2) Если ордината центра ячейки меньше 1.5, то вычисляем площадь $V_3$ под косинусом $\alpha\cos(n\pi x)+0.7$ внутри ячейки, $V_2=\Delta x\Delta y$. Так как $u_2=u_3=0$ и $v_2=v_3=0$, то задаём значения скоростей $u=0, v=0$. Найдём значения $\rho$ и $\varepsilon$. Из закона сохранения массы получаем:

$$\rho \Delta x \Delta y = \rho_2 V_2 + \rho_3 V_3$$

$$\rho = \frac{\rho_2 V_2 + \rho_3 V_3}{\Delta x \Delta y}$$

Из закона сохранения энергии получаем:

$$\rho \varepsilon \Delta x \Delta y = \rho_2 \varepsilon_2 V_2 + \rho_3 \varepsilon_3 V_3$$

$$\varepsilon = \frac{\rho_2 \varepsilon_2 V_2 + \rho_3 \varepsilon_3 V_3}{\rho \Delta x \Delta y}$$

Вычислим площадь под косинусом $a \cos(w x) + b$ внутри прямоугольника $x_1 \le x \le x_2$, $y_1 \le y \le y_2$. Построим абсциссы $p_0 < p_1 < \ldots < p_n$, где $p_0 = x_1, p_n = x_2$, а $p_1, p_2, \ldots, p_{n-1}$ - абсциссы точек пересечения косинуса с прямыми $y = y_1$ и $y = y_2$. Внутри каждого отрезка $[p_i, p_{i+1}]$ косинус не пересекает прямые $y = y_1$ и $y = y_2$, что позволяет вычислить площадь под косинусом внутри прямоугольника $p_i \le x \le p_{i+1}$, $y_1 \le y \le y_2$ следующим образом:

1) Вычисляем значение косинуса в середине отрезка $\frac{p_i + p_{i+1}}{2}$.

2) Если значение косинуса меньше $y_1$, значит косинус целиком проходит под прямоугольником, потому площадь равна 0.

3) Если значение косинуса больше $y_2$, значит косинус целиком проходит над прямоугольником, потому площадь равна площади прямоугольника $(p_{i+1} - p_i)(y_2 - y_1)$.

4) Если значение косинуса лежит в пределах от $y_1$ до $y_2$, то косинус проходит внутри прямоугольника, потому площадь считается как интеграл:

$$S = \int_{p_i}^{p_{i+1}} (a \cos(w x) + b - y_1) \, d x = \frac{a}{w} (\sin(w p_{i+1}) - \sin(w p_i)) + (p_{i+1} - p_i)(b - y_1).$$

Итоговая площадь считается как сумма площадей на каждом отрезке.

Численное интегрирование будем проводить по следующей схеме:

$$\frac{\partial \varphi_{i,j}}{\partial t} + \frac{F^x_{i+\frac{1}{2},j} - F^x_{i-\frac{1}{2},j}}{\Delta x} + \frac{F^y_{i,j+\frac{1}{2}} - F^y_{i,j-\frac{1}{2}}}{\Delta y} = 0.$$

В схеме первого порядка точности по пространству потоки на границах считаются следующим образом:

$$F^x_{i+\frac{1}{2},j} = \frac{F^x_{i,j} + F^x_{i+1,j}}{2} - c_{i+\frac{1}{2},j} \frac{\varphi_{i+1,j} - \varphi_{i,j}}{2},$$

$$c_{i+\frac{1}{2},j} = max\left(c_{i,j} + \left|u_{i,j}\right|, c_{i+1,j} + \left|u_{i+1,j}\right|\right).$$

В схеме второго порядка точности используются следующие величины:

$$\varphi^{-}_{i+\frac{1}{2},j} = \varphi_{i,j} + \alpha\left(R^{-}\right)\frac{\varphi_{i+1,j} - \varphi_{i,j}}{2},$$

$$\varphi^{+ ¿= \varphi_{i+1,j} - \alpha ¿¿}_{i+\frac{1}{2},j}$$

$$R^{-} = \frac{\varphi_{i,j} - \varphi_{i-1,j}}{\varphi_{i+1,j} - \varphi_{i,j}}, R^{+ ¿ = \frac{\varphi_{i+1,j} - \varphi_{i,j}}{\varphi_{i+2,j} - \varphi_{i+1,j}}, ¿}$$

$$\alpha\left(R\right) = \begin{cases} 0, R < 0, \\ R, 0 \le R \le 1, \\ 1, R > 1. \end{cases}$$

Вместо величин $i, j$ подставляем величины, вычисленные по $\varphi^{-}_{i+\frac{1}{2},j}$, а вместо $i+1, j$ - вычисленные по $\varphi^{+¿¿}_{i+\frac{1}{2},j}$. Потоки $F^{x}_{i-\frac{1}{2},j}, F^{y}_{i,j+\frac{1}{2}}, F^{y}_{i,j-\frac{1}{2}}$ считаются аналогично.

В схеме первого порядка точности по времени производная аппроксимируется как разность:

$$\frac{\varphi^{n+1}_{i,j} - \varphi^{n}_{i,j}}{\Delta t} + \frac{F^{x,n}_{i+\frac{1}{2},j} - F^{x,n}_{i-\frac{1}{2},j}}{\Delta x} + \frac{F^{y,n}_{i,j+\frac{1}{2}} - F^{y,n}_{i,j-\frac{1}{2}}}{\Delta y} = 0.$$

В схеме второго порядка $\varphi^{n+1}_{i,j}$ считается следующим образом:

$$\frac{\varphi^{n+0.5}_{i,j} - \varphi^{n}_{i,j}}{0.5\Delta t} + \frac{F^{x,n}_{i+\frac{1}{2},j} - F^{x,n}_{i-\frac{1}{2},j}}{\Delta x} + \frac{F^{y,n}_{i,j+\frac{1}{2}} - F^{y,n}_{i,j-\frac{1}{2}}}{\Delta y} = 0,$$

$$\frac{\varphi^{n+1}_{i,j} - \varphi^{n}_{i,j}}{\Delta t} + \frac{F^{x,n+0.5}_{i+\frac{1}{2},j} - F^{x,n+0.5}_{i-\frac{1}{2},j}}{\Delta x} + \frac{F^{y,n+0.5}_{i,j+\frac{1}{2}} - F^{y,n+0.5}_{i,j-\frac{1}{2}}}{\Delta y} = 0.$$

Подключим необходимые для вычислений библиотеки:

```
import numpy as np
import pylab
import math
from tqdm import tqdm

def to_phi(nrho, nu, nv, neps, k, i, j):
    rho, u, v, eps = nrho[k, i, j], nu[k, i, j], nv[k, i, j], neps[k, 
i, j]
```

```
    return np.array([rho, rho * u, rho * v, rho * (eps + 0.5 * (u * u
+ v * v))])

def from_phi(phi):
    rho = phi[0]
    u = phi[1] / rho
    v = phi[2] / rho
    eps = phi[3] / rho - 0.5 * (u * u + v * v)
    return rho, u, v, eps

def get_Fx(gamma, phi):
    rho, u, v, eps = from_phi(phi)
    P = (gamma - 1) * rho * eps
    E = rho * (eps + 0.5 * (u * u + v * v))
    return np.array([ rho * u, rho * u * u + P, rho * u * v, (E + P) *
u ])

def get_Fy(gamma, phi):
    rho, u, v, eps = from_phi(phi)
    P = (gamma - 1) * rho * eps
    E = rho * (eps + 0.5 * (u * u + v * v))
    return np.array([ rho * v, rho * u * v, rho * v * v + P, (E + P) *
v ])

def get_F_half(phi_m1, phi_p1, phi_m, phi_p, gamma, vert):
    rho_m1, u_m1, v_m1, eps_m1 = from_phi(phi_m1)
    rho_p1, u_p1, v_p1, eps_p1 = from_phi(phi_p1)
    rho_m, u_m, v_m, eps_m = from_phi(phi_m)
    rho_p, u_p, v_p, eps_p = from_phi(phi_p)

    s_m1 = v_m1 if vert else u_m1
    s_p1 = v_p1 if vert else u_p1
    s_m = v_m if vert else u_m
    s_p = v_p if vert else u_p

    # скорости звука
    c_m1 = math.sqrt(gamma * (gamma - 1) * eps_m1)
    c_p1 = math.sqrt(gamma * (gamma - 1) * eps_p1)
    c_m = math.sqrt(gamma * (gamma - 1) * eps_m)
    c_p = math.sqrt(gamma * (gamma - 1) * eps_p)

    c_mh = max(c_m1 + abs(s_m1), c_m + abs(s_m))
    c_ph = max(c_p1 + abs(s_p1), c_p + abs(s_p))

    # потоки
    get_F = get_Fy if vert else get_Fx

    F_m1 = get_F(gamma, phi_m1)
    F_p1 = get_F(gamma, phi_p1)
    F_m = get_F(gamma, phi_m)
```

```python
    F_p = get_F(gamma, phi_p)

    F_mh = 0.5 * (F_m1 + F_m - c_mh * (phi_m1 - phi_m))
    F_ph = 0.5 * (F_p + F_p1 - c_ph * (phi_p - phi_p1))
    return F_mh, F_ph

#второй порядок точности
def border2(phi_m, phi, phi_p, phi_2p):
    aRm = alpha_all(phi - phi_m, phi_p - phi)
    aRp = alpha_all(phi_p - phi, phi_2p - phi_p)

    phi_hm = phi + 0.5 * aRm * (phi_p - phi)
    phi_hp = phi_p - 0.5 * aRp * (phi_2p - phi_p)
    return phi_hm, phi_hp

def alpha_all(R1, R2):
    R = np.zeros_like(R1)
    for i in range(R.shape[0]):
        if R2[i] != 0:
            R[i] = R1[i] / R2[i]
            if R[i] < 0: R[i] = 0
            elif R[i] > 1: R[i] = 1
    return R

def solve(orderXY, orderT,n, Nx, Ny, Nt, T, M, gamma, alpha, rho_0,
eps_0, u_0, v_0):
    rho = np.zeros(shape=(Nt + orderT - 1, Nx + 4, Ny + 4))
    u = np.zeros(shape=(Nt + orderT - 1, Nx + 4, Ny + 4))
    v = np.zeros(shape=(Nt + orderT - 1, Nx + 4, Ny + 4))
    eps = np.zeros(shape=(Nt + orderT - 1, Nx + 4, Ny + 4))

    dx, dy, dt = 1.0 / Nx, 1.0 / Ny, T / Nt

    # начальные условия
    rho1 = rho_0[0]
    rho2 = rho_0[1]
    rho3 = rho_0[2]
    rho4 = rho_0[3]

    eps1 = eps_0[0]
    eps2 = eps_0[1]
    eps3 = eps_0[2]
    eps4 = eps_0[3]

    u1 = u_0[0]
    u2 = u_0[1]
    u3 = u_0[2]
    u4 = u_0[3]

    v1 = v_0[0]
```

```
v2 = v_0[1]
v3 = v_0[2]
v4 = v_0[3]

for i in range(2, Nx + 2):
    for j in range(2, Ny + 2):
        x, y = (i - 2 + 0.5) * dx, (j - 2 + 0.5) * dy

        if(i <= (Nx + 2)/2):
            if(j <= (Ny + 2)/2):
                #область 1
                rho[0,i,j] = rho1
                eps[0,i,j] = eps1
                u[0,i,j] = u1
                v[0,i,j] = v1
            else:
                #область 2
                rho[0,i,j] = rho2
                eps[0,i,j] = eps2
                u[0,i,j] = u2
                v[0,i,j] = v2
        else:
            if(j <= (Ny + 2)/2):
                #область 3
                rho[0,i,j] = rho3
                eps[0,i,j] = eps3
                u[0,i,j] = u3
                v[0,i,j] = v3
            else:
                #область 4
                rho[0,i,j] = rho4
                eps[0,i,j] = eps4
                u[0,i,j] = u4
                v[0,i,j] = v4


# вычисление следующих моментов времени
k0, k = -1, orderT - 2
for repeat in tqdm(range(0, orderT * (Nt - 1))):

    if orderT == 1:
        k0 += 1
        k += 1
    elif k0 == k:
        k = -1
    else:
        k0 += 1
        k = k0

    # краевые условия
```

```
        for j in range(2, Ny + 2):
            for i in range(2):
                # слева
                rho[k, i, j] = rho[k, 3 - i, j]
                u[k, i, j] = -u[k, 3 - i, j]
                v[k, i, j] = v[k, 3 - i, j]
                eps[k, i, j] = eps[k, 3 - i, j]
                # справа
                rho[k, Nx + 2 + i, j] = rho[k, Nx + 1 - i, j]
                u[k, Nx + 2 + i, j] = -u[k, Nx + 1 - i, j]
                v[k, Nx + 2 + i, j] = v[k, Nx + 1 - i, j]
                eps[k, Nx + 2 + i, j] = eps[k, Nx + 1 - i, j]

        for i in range(2, Nx + 2):
            for j in range(2):
                # снизу
                rho[k, i, j] = rho[k, i, 3 - j]
                u[k, i, j] = u[k, i, 3 - j]
                v[k, i, j] = -v[k, i, 3 - j]
                eps[k, i, j] = eps[k, i, 3 - j]
                # сверху
                rho[k, i, Ny + 2 + j] = rho[k, i, Ny + 1 - j]
                u[k, i, Ny + 2 + j] = u[k, i, Ny + 1 - j]
                v[k, i, Ny + 2 + j] = -v[k, i, Ny + 1 - j]
                eps[k, i, Ny + 2 + j] = eps[k, i, Ny + 1 - j]

        # следующие значения
        for i in range(2, Nx + 2):
            for j in range(2, Ny + 2):
                # текущее значение phi
                phi1 = to_phi(rho, u, v, eps, k0, i, j)

                # вычисляем потоки на границах
                phi = to_phi(rho, u, v, eps, k, i, j)
                phi_mx = to_phi(rho, u, v, eps, k, i - 1, j)
                phi_px = to_phi(rho, u, v, eps, k, i + 1, j)
                phi_my = to_phi(rho, u, v, eps, k, i, j - 1)
                phi_py = to_phi(rho, u, v, eps, k, i, j + 1)

                if orderXY == 2:
                    phi_2mx = to_phi(rho, u, v, eps, k, i - 2, j)
                    phi_2px = to_phi(rho, u, v, eps, k, i + 2, j)
                    phi_2my = to_phi(rho, u, v, eps, k, i, j - 2)
                    phi_2py = to_phi(rho, u, v, eps, k, i, j + 2)

                    phi_hmx_m, phi_hmx_p = border2(phi_2mx, phi_mx,
phi, phi_px)
                    phi_hpx_m, phi_hpx_p = border2(phi_mx, phi,
phi_px, phi_2px)
                    phi_hmy_m, phi_hmy_p = border2(phi_2my, phi_my,
```

```
phi, phi_py)
                    phi_hpy_m, phi_hpy_p = border2(phi_my, phi,
phi_py, phi_2py)

                    F_mhx, F_phx = get_F_half(phi_hmx_p, phi_hpx_m,
phi_hmx_m, phi_hpx_p, gamma, False)
                    F_mhy, F_phy = get_F_half(phi_hmy_p, phi_hpy_m,
phi_hmy_m, phi_hpy_p, gamma, True)
                else:
                    F_mhx, F_phx = get_F_half(phi, phi, phi_mx,
phi_px, gamma, False)
                    F_mhy, F_phy = get_F_half(phi, phi, phi_my,
phi_py, gamma, True)

                # вычисляем новое значение phi
                mult = 0.5 if k == k0 and orderT == 2 else 1
                phi2 = phi1 - mult * dt / dx * (F_phx - F_mhx) - mult
* dt / dy * (F_phy - F_mhy)

                # выражаем параметры
                kw = -1 if k == k0 and orderT == 2 else k0 + 1
                rho[kw, i, j], u[kw, i, j], v[kw, i, j], eps[kw, i, j]
= from_phi(phi2)

    # убираем границы и вспомогательный слой
    return rho[:Nt, 2:Nx+2, 2:Ny+2], u[:Nt, 2:Nx+2, 2:Ny+2], v[:Nt,
2:Nx+2, 2:Ny+2], eps[:Nt, 2:Nx+2, 2:Ny+2]

# визуализация графиков
def visualise(time_index):
    print('t =', time_index * T / Nt)

    pylab.figure()
    fig, splots = pylab.subplots(1, 4)
    fig.set_figheight(5)
    fig.set_figwidth(20)

    for i, arr in enumerate([np.log(rho), u, v, np.log(eps)]):
        splots[i].imshow(np.flip(arr[time_index].T, axis=0),
cmap='RdBu')

    pylab.show()

# Вариант 19
alpha = 0.05
n = 4
M = 4

Nx, Ny = 40, 40
Nt = 30
```

```
T = 0.4

rho0 = [1,1,1,1]
eps0 = [1,1,1,10]
u0 = [0,0,0,0]
v0 = [0,0,0,0]

t1, t2, t3 = 1, 5, 20
```

Запустим схему первого порядка точности по времени и пространству:

```
gamma = 25/24
rho, u, v, eps = solve(orderXY = 1, orderT = 1, Nx = Nx, Ny = Ny, Nt =
Nt, T = T, M = M, gamma = gamma,
                       alpha = alpha, n = n,
                       rho_0 = rho0,
                       eps_0 = eps0,
                       u_0 = u0,
                       v_0 = v0)
```

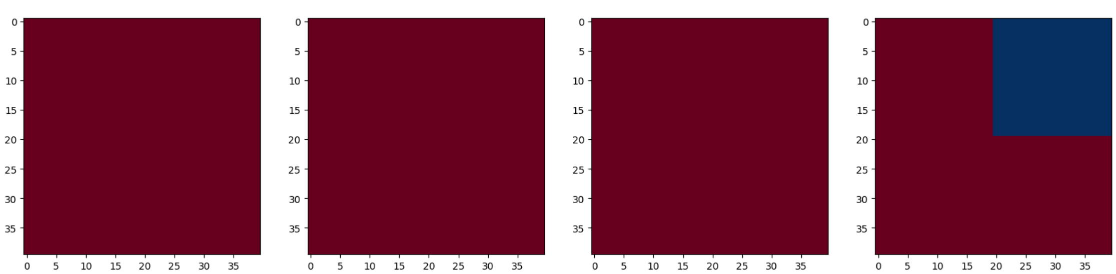
```
100%|██████████| 29/29 [00:02<00:00, 11.35it/s]
```

Построим графики для начального момента времени:

```
visualise(0)
visualise(t1)
visualise(t2)
visualise(t3)
```

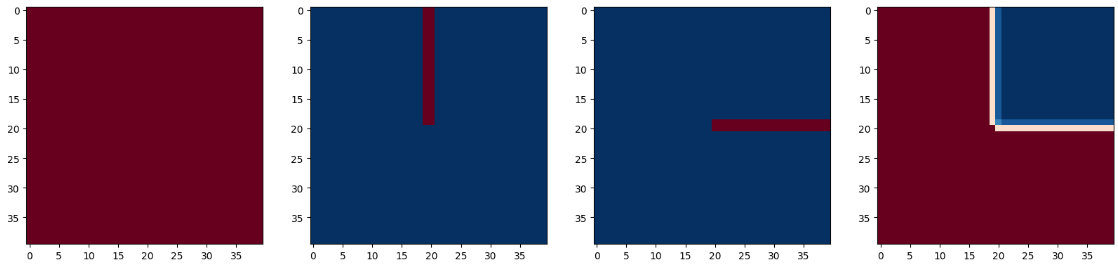
```
t = 0.0
```

```
<Figure size 640x480 with 0 Axes>
```



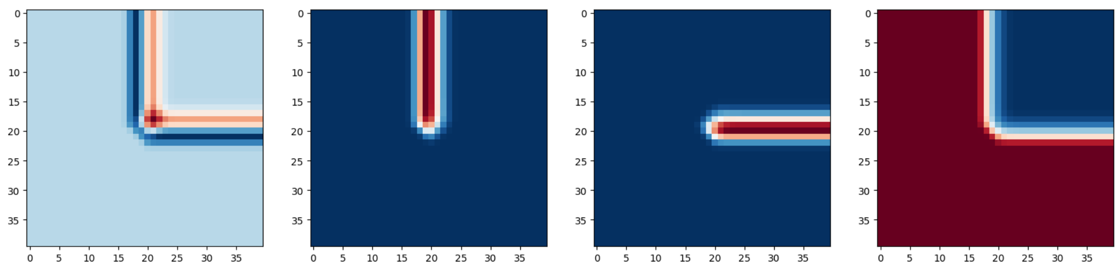

```
t = 0.013333333333333334
```

```
<Figure size 640x480 with 0 Axes>
```



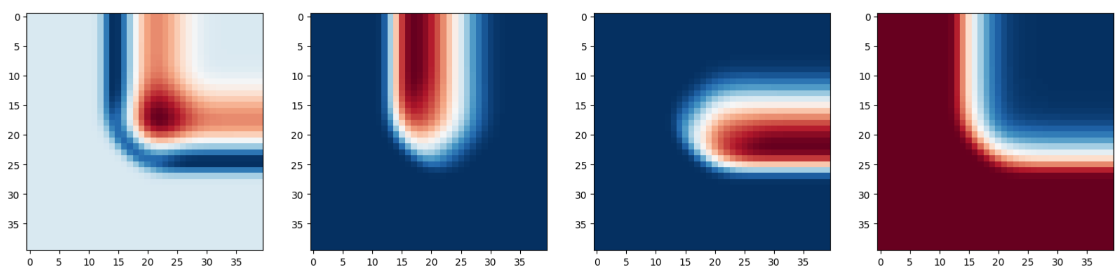
t = 0.06666666666666667

<Figure size 640x480 with 0 Axes>



t = 0.26666666666666666

<Figure size 640x480 with 0 Axes>



второй порядок по пространству, первый по времени

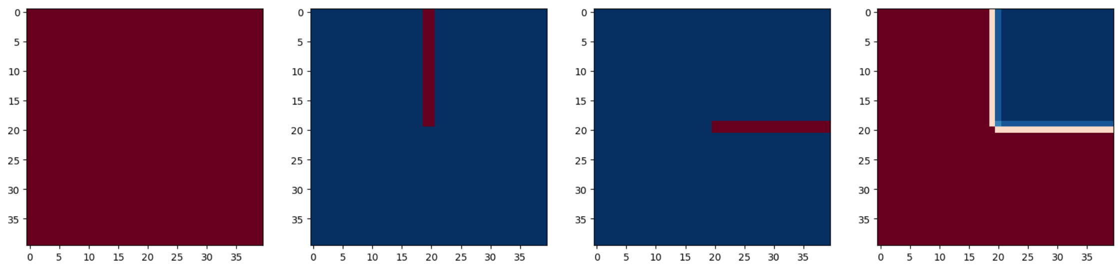
```
gamma = 26/25
rho, u, v, eps = solve(orderXY = 2, orderT = 1, Nx = Nx, Ny = Ny, Nt =
Nt, T = T, M = M, gamma = gamma,
                       alpha = alpha, n = n,
                       rho_0 = rho0,
                       eps_0 = eps0,
                       u_0 = u0,
                       v_0 = v0)
```

100%|██████████| 29/29 [00:06<00:00,  4.76it/s]

```
visualise(t1)
visualise(t2)
visualise(t3)
```

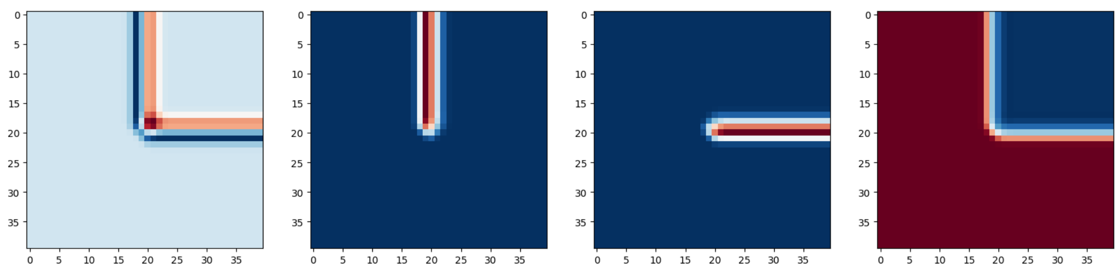
t = 0.013333333333333334

<Figure size 640x480 with 0 Axes>



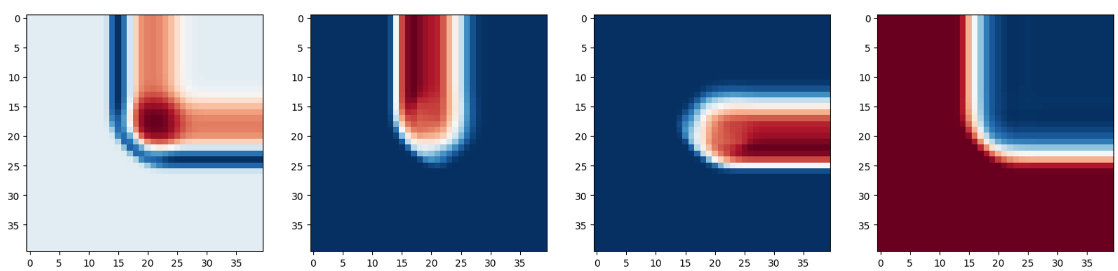
t = 0.06666666666666667

<Figure size 640x480 with 0 Axes>



t = 0.26666666666666666

<Figure size 640x480 with 0 Axes>



Второй порядок по пространству, второй по времени.

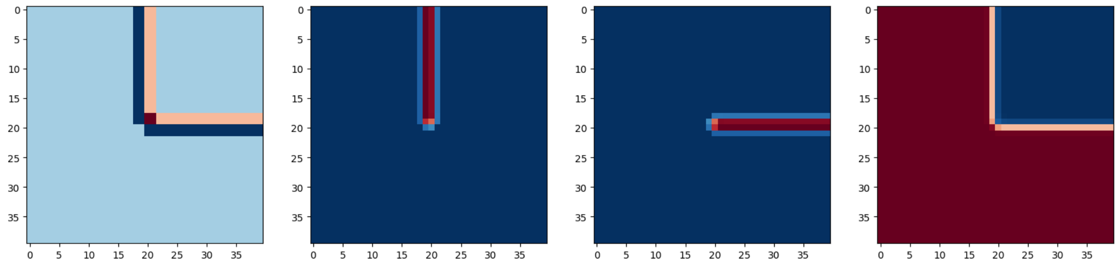
```
gamma = 28/27
rho, u, v, eps = solve(orderXY = 2, orderT = 2, Nx = Nx, Ny = Ny, Nt = 
Nt, T = T, M = M, gamma = gamma,
                       alpha = alpha, n = n,
                       rho_0 = rho0,
                       eps_0 = eps0,
                       u_0 = u0,
                       v_0 = v0)
```

100%|██████████| 58/58 [00:12<00:00,  4.72it/s]

```
visualise(t1)
visualise(t2)
visualise(t3)
```

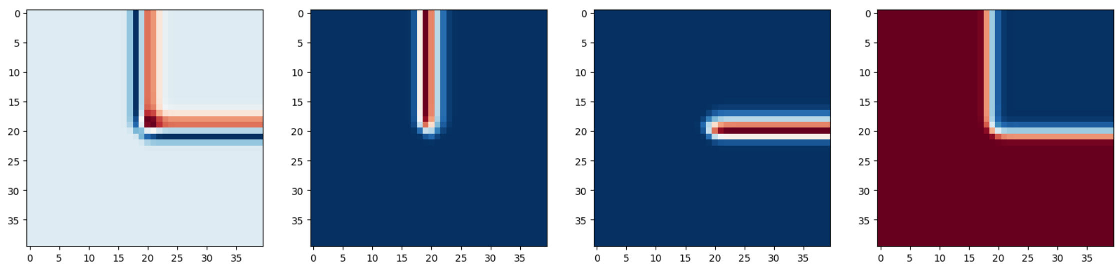
t = 0.013333333333333334

<Figure size 640x480 with 0 Axes>



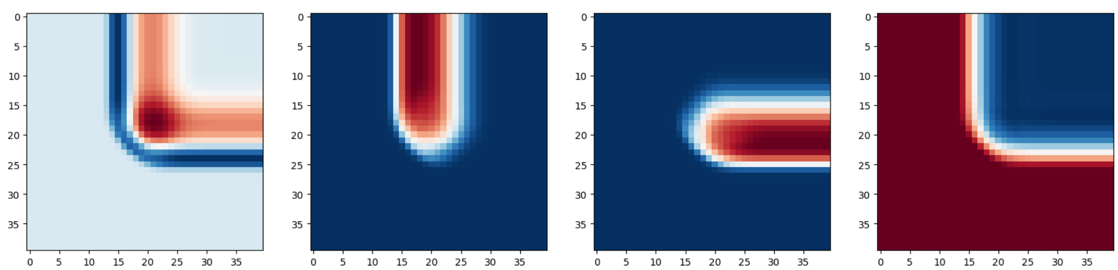
t = 0.06666666666666667

<Figure size 640x480 with 0 Axes>



t = 0.26666666666666666

<Figure size 640x480 with 0 Axes>



```
# измельчаем, пока не выполнится
# mean(|rho_1 - rh0_2| + mean(|u_1 - u_2| + mean(|v_1 - v_2| + mean(|
eps_1 - eps_2|) < lim_val
def subdiv(lim_val):
    nx, ny, nt, tmax = 20, 20, 10, T / 2
    print('Size:', nx, 'x', ny, 'x', nt)
    r1, u1, v1, e1 = rho, u, v, eps = solve(orderXY = 1, orderT = 1,
Nx = nx, Ny = ny, Nt = nt, T = tmax, M = M, gamma = gamma,
                              alpha = alpha, n = n,
                              rho_0 = [10,1,1,1],
                              eps_0 = [10,1,1,10],
                              u_0 = [0,0,0,0],
                              v_0 = [0,0,0,0])
    while True:
```

```
        nx, ny, nt = 2 * nx, 2 * ny, 2 * nt
        print('Size:', nx, 'x', ny, 'x', nt)
        r2, u2, v2, e2 =rho, u, v, eps = solve(orderXY = 1, orderT =
1, Nx = nx, Ny = ny, Nt = nt, T = tmax, M = M, gamma = gamma,
                          alpha = alpha, n = n,
                          rho_0 = [10,1,1,1],
                          eps_0 = [10,1,1,10],
                          u_0 = [0,0,0,0],
                          v_0 = [0,0,0,0])
        diffR = np.abs(r1 - r2[::2,::2,::2]).mean()
        diffU = np.abs(u1 - u2[::2,::2,::2]).mean()
        diffV = np.abs(v1 - v2[::2,::2,::2]).mean()
        diffE = np.abs(e1 - e2[::2,::2,::2]).mean()
        diff = diffR + diffU + diffV + diffE

        print('d rho:', diffR)
        print('d u:', diffU)
        print('d v:', diffV)
        print('d eps:', diffE)
        print('Sum d:', diff)

        if diff < lim_val:
            print('Done!')
            return r1, u1, v1, e1, r2, u2, v2, e2

        r1, u1, v1, e1 = r2, u2, v2, e2
```

Сетки с другими размерами (80x80)

```
r1, u1, v1, e1, r2, u2, v2, e2 = subdiv(0.3)
```

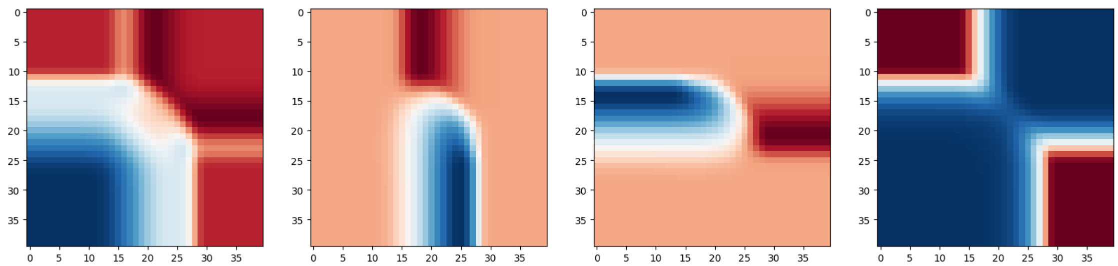
```
Size: 20 x 20 x 10

100%|██████████| 9/9 [00:00<00:00, 39.99it/s]

Size: 40 x 40 x 20

100%|██████████| 19/19 [00:01<00:00, 12.03it/s]

d rho: 0.1262550377282932
d u: 0.012767485832933011
d v: 0.01276748583293301
d eps: 0.17233149586488675
Sum d: 0.32412150525904593
Size: 80 x 80 x 40

100%|██████████| 39/39 [00:12<00:00,  3.14it/s]

d rho: 0.10056030987687573
d u: 0.009995071871254235
d v: 0.009995071871254233
d eps: 0.12468496431949977
```

```
Sum d: 0.24523541793888395
Done!
```

```python
rho, u, v, eps = r1, u1, v1, e1
visualise(rho.shape[0] - 1)
```

```
t = 0.25333333333333335
```

```
<Figure size 640x480 with 0 Axes>
```



```python
rho, u, v, eps = r2, u2, v2, e2
visualise(rho.shape[0] - 1)
```

```
t = 0.52
```

```
<Figure size 640x480 with 0 Axes>
```