

Лабораторная работа №2

«Численное решение уравнения Хопфа»

Квазилинейное уравнение переноса

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = 0 \quad (1a)$$

или в дивергентной форме

$$\frac{\partial u}{\partial t} + \frac{\partial F(u)}{\partial x} = 0, \quad F(u) = u^2/2, \quad (1b)$$

называется *уравнением Хопфа* и относится к основным модельным уравнениям вычислительной гидроаэродинамики, на которых тестируются различные разностные схемы.

В случае разрывных начальных данных

$$u(x, 0) = \varphi(x) = \begin{cases} u_L, & x \leq 0 \\ u_R, & x \geq 0 \end{cases} \quad (2)$$

задача (1) – (2) называется *задачей Римана* и имеет точное решение

$$u_{ex} = \begin{cases} u_L, & x \leq Dt \\ u_R, & x \geq Dt \end{cases}$$

где $D = (u_L + u_R)/2$ – скорость распространения сильного разрыва (УВ).

Для численного решения (1) – (2) можно использовать одну из разностных схем:

КИР

$$\frac{v_i^{n+1} - v_i^n}{\tau} + \frac{1}{h} \begin{cases} F_{i+1}^n - F_i^n, & v_i^n < 0 \\ F_i^n - F_{i-1}^n, & v_i^n > 0 \end{cases} = 0, \quad (3)$$

Мак-Кормака 1

$$\frac{\tilde{v}_i - v_i^n}{\tau} + \frac{F_{i+1}^n - F_i^n}{h} = 0, \quad (4a)$$

$$\frac{v_i^{n+1} - (v_i^n + \tilde{v}_i)/2}{\tau} + \frac{\tilde{F}_i - \tilde{F}_{i-1}}{2h} = 0, \quad (4b)$$

Мак-Кормака 2

$$\frac{\tilde{v}_i - v_i^n}{\tau} + \frac{F_i^n - F_{i-1}^n}{h} = 0, \quad (5a)$$

$$\frac{v_i^{n+1} - (v_i^n + \tilde{v}_i)/2}{\tau} + \frac{\tilde{F}_{i+1} - \tilde{F}_i}{2h} = 0, \quad (5b)$$

Лакса-Вендроффа

$$\frac{v_i^{n+1} - v_i^n}{\tau} + \frac{F_{i+1}^n - F_{i-1}^n}{2h} - \frac{\tau}{4h^2} [(v_{i+1}^n + v_i^n)(F_{i+1}^n - F_i^n) - (v_i^n + v_{i-1}^n)(F_i^n - F_{i-1}^n)] = 0, \quad (6)$$

неявная 1

$$\frac{v_i^{n+1} - v_i^n}{\tau} + \frac{F_i^{n+1} - F_{i-1}^{n+1}}{h} = 0, \quad (7)$$

неявная 2

$$\frac{v_i^{n+1} - v_i^n}{\tau} + \frac{1}{2} \left[\frac{F_i^{n+1} - F_{i-1}^{n+1}}{h} + \frac{F_i^n - F_{i-1}^n}{h} \right] = 0, \quad (8)$$

TVD

$$\frac{v_i^{n+1}-v_i^n}{\tau} + \frac{1}{h} \begin{cases} F_{i+1}^n - F_i^n, & \alpha_i < 0 \\ F_i^n - F_{i-1}^n, & \alpha_i > 0 \end{cases} = 0, \quad (9)$$

где

$$\alpha_i = \begin{cases} v_i^n, & \Delta v \equiv v_{i+1}^n - v_i^n = 0 \\ \frac{F_{i+1}^n - F_i^n}{\Delta v}, & \Delta v \neq 0 \end{cases}$$

неявная 3

$$\frac{v_i^{n+1}-v_i^n}{\tau} + \frac{F_{i+1}^{n+1}-F_{i-1}^{n+1}}{2h} = 0, \quad (10)$$

Кранка-Николсона

$$\frac{v_i^{n+1}-v_i^n}{\tau} + \frac{1}{2} \left[\frac{F_{i+1}^n - F_{i-1}^n}{2h} + \frac{F_{i+1}^{n+1} - F_{i-1}^{n+1}}{2h} \right] = 0, \quad (11)$$

компактная

$$A \frac{v_i^{n+1}-v_i^n}{\tau} + \frac{1}{2} \left[\frac{F_{i+1}^n - F_{i-1}^n}{2h} + \frac{F_{i+1}^{n+1} - F_{i-1}^{n+1}}{2h} \right] = 0, \quad (12)$$

где

$$Aw_i = \frac{1}{6} (w_{i-1} + 4w_i + w_{i+1}).$$

ЗАДАНИЕ. Методом конечных разностей решить для (16) задачу Римана (2). Исследовать зависимость максимальной погрешности

$$\delta_{max} = \max_{1 \leq i \leq N, 0 \leq n \leq M} |v_i^n - u_{ex}(x_i, t_n)|$$

приближенного решения $\{v_i^n\}$ от шага сетки h (при фиксированном шаге по времени τ) и от шага по времени τ (при фиксированном шаге по пространству h).

Варианты задания приведены в таблице:

№ вар.	№ схемы	u_L	u_R	№ вар.	№ схемы	u_L	u_R
1	3	3/2	1/2	2	4	3/2	1/2
3	5	2	1	4	6	2	1
5	7	5/2	3/2	6	8	5/2	3/2
7	9	4/3	1/3	8	3	4/3	1/3
9	4	5/4	1/4	10	5	5/4	1/4
11	6	6/5	2/5	12	7	6/5	2/5
13	8	3/2	1/2	14	9	3/2	1/2
15	3	2	1	16	4	2	1
17	5	5/2	3/2	18	6	5/2	3/2
19	7	4/3	1/3	20	8	4/3	1/3
21	9	5/4	1/4	22	10	5/4	1/4
23	11	6/5	2/5	24	12	6/5	2/5

УКАЗАНИЯ. В качестве основы использовать программу «conv1» (см. ниже), в которой для решения задачи (1) – (2) используется схема Лакса

$$\frac{v_i^{n+1} - (v_{i+1}^n + v_{i-1}^n)/2}{\tau} + \frac{F_{i+1}^n - F_{i-1}^n}{2h} = 0. \quad (13)$$

Выразив из (13) решение на новом временном слое v_i^{n+1} , получим следующий алгоритм:

$$v_1^{n+1} = u_L, \quad i = 1$$

$$v_i^{n+1} = \frac{v_{i+1}^n + v_{i-1}^n}{2} - \frac{\tau}{2h} (F_{i+1}^n - F_{i-1}^n), \quad i = 2, \dots, N_x.$$

Для расчета потока $F_{N_x+1}^n$ на правой границе в узле $i = N_x + 1$ применяем экстраполяцию

$$v_{N_x+1}^n = 2v_{N_x}^n - v_{N_x-1}^n. \quad (14)$$

Этот прием применяем для всех схем 2-го порядка (4) - (6), (10) - (12).

Для регуляризации численного решения, полученного по схемам 2-го порядка (4) - (6), (10) - (12), используем явное сглаживание

$$v_i \rightarrow \hat{v}_i = (1 - 2\alpha)v_i + \alpha(v_{i-1} + v_{i+1}), \quad i = 2, \dots, N_x - 1, \quad \alpha \sim 10^{-2} \dots 10^{-1}.$$

Значение параметра α подбираем так, чтобы минимизировать одновременно численные «размазывание» и «рябь».

Текст программы «conv1»:

```
from math import sqrt
import numpy as np
import matplotlib.pyplot as plt

def Phi(x,t,uL,uR):
    Phi = uL
    if x > 0:
        Phi = uR
    return Phi

def Uex(x,t,uL,uR):
    D = (uL + uR) / 2.0
    if x <= D * t:
        Uex = uL
    else:
        Uex = uR
    return Uex

def Flux(u): return u * u / 2.0

def Xshock(x, v, Nx):
    dvmax = 0.0
    for i in range(2, Nx+1):
        dv = v[i] - v[i-1]
        if abs(dv) > dvmax: imax = i; dvmax = abs(dv)
    xsh = x[imax]
    return xsh

def SMOOTHING(v, Nx, alf):
    v1 = [v[i] for i in range(Nx+1)]
    for i in range(2, Nx): v[i] = (1-2*alf)*v1[i]+alf*(v1[i-1]+v1[i+1])

def LAX(uL, tauh, v, Nx):
    v1 = [v[i] for i in range(Nx+1)]
    v1.append(2*v1[Nx]-v1[Nx-1])
    for i in range(2,Nx+1):
        v[i] = 0.5*(v1[i+1]+v1[i-1])-0.5*tauh*(Flux(v1[i+1])-Flux(v1[i-1]))
    v[1] = uL

Nx = 101 # число узлов
x,u,v,v1 = ([0]*(Nx+1) for k in range(4))
xL, xR = -0.1, 0.9
h = (xR-xL)/(Nx-1)
tmax = 1.5
gam = 1.0 # число Куранта
print(' h=', h)
print(' gam=', gam)
uL = 1
uR = 0 # НУ
t = 0.
for i in range(1,Nx+1):
    x[i] = xL+(i-1)*h
    u[i] = Phi(x[i],t,uL,uR)
    v[i] = u[i]
tau = gam*h/max([abs(v[i]) for i in range(1,Nx+1)]) # нач. шаг по времени
Nt = int(tmax/tau+0.5) # число шагов по времени
nout = Nt//10 # частота вывода

n = 0
deltamax = 0
time = np.zeros(Nt+1)
V = np.zeros((Nt+1,Nx))

while n <= Nt:
    time[n] = t
    V[n,:] = np.array(v[1:])
    delta = sqrt(h*sum([(u[i]-v[i])**2 for i in range(Nx+1)]))
    if delta>deltamax: deltamax = delta
    if n%nout==0:
        xsh = Xshock(x, v, Nx)

print('%5d'%n,'%6.3f'%t,'%6.3f'%tau,'%10.3e'%delta,'%6.3f'%xsh)
plt.plot(x[1:],u[1:], 'k', x[1:],v[1:], 'b')
tstr = 't='+str(t)[0:4]
plt.title(tstr)
plt.axis([xL, xR, 0.0, 1.2])
```

```

plt.xlabel("x")
plt.ylabel("u, v")
plt.pause(0.0001)
tauh = tau/h
LAX(uL, tauh, v, Nx)
# LAXVEN(uL, tauh, v, Nx)
# KIR(uL, tauh, v, Nx)
# MACC1(uL, tauh, v, Nx)
# MACC2(uL, tauh, v, Nx)
# IMPL(uL, tauh, v, Nx)
# TVD(uL, tauh, v, Nx)
# SMOOTHING(v, Nx, 0.1)
t += tau
for i in range(Nx): u[i] = Uex(x[i],t,uL,uR)
tau = gam*h/max([abs(v[i]) for i in range(1,Nx+1)])
n += 1

print('delmax=%10.3e'%deltamax)
plt.show()

X,T = np.meshgrid(x[1:],time)

fig = plt.figure(dpi=200)
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(X, T, V, cmap = 'jet')

```

```

ax.set_xlabel("x", labelpad=20)
ax.set_ylabel("t", labelpad=20)
ax.set_zlabel("v(x,t)", labelpad=20)
fig.colorbar(surf, shrink=0.5, aspect=8)
plt.savefig('surface',dpi=fig.dpi,bbox_inches='tight',pad_inches=0.1)
plt.show()

def Isolines(X,Y,Z,title,fname):
    levels = np.linspace(np.min(Z),np.max(Z),10)
    fig = plt.figure(dpi=200)
    _, ax = plt.subplots()
    ax.set_aspect('equal')
    plt.contour(X,Y,Z,levels=levels,cmap='jet')
    plt.xlabel("x")
    plt.ylabel("t")
    plt.title(title)
    plt.colorbar()

plt.savefig(fname,dpi=fig.dpi,bbox_inches='tight',pad_inches=0.1)
plt.show()

Isolines(X,T,V,'Изолинии v(x,t)','contours')

```

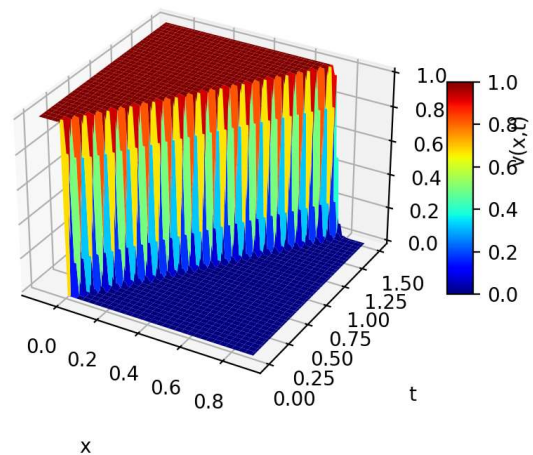
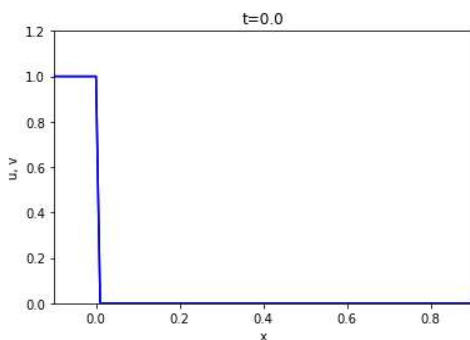
Назначение функций: «Phi» – расчет начальных данных $\varphi(x)$, «Uex» – точного решения $u_{ex}(x, t)$, «Flux» – потока $F(u)$, «Xshock» – координаты конечно-разностной УВ, «SMOOTHING» – сглаживание численного решения.

Результаты запуска:

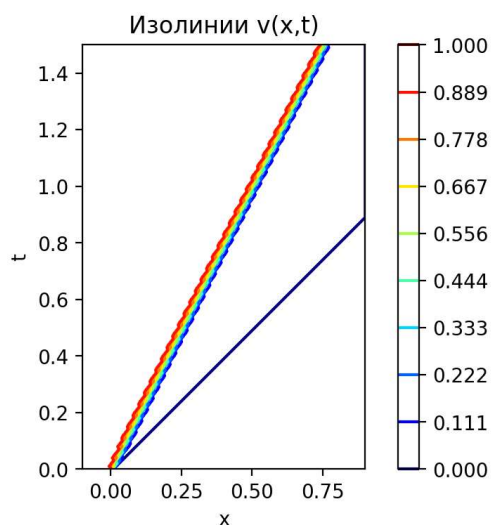
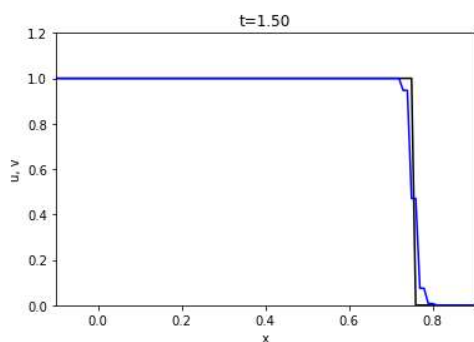
```

h= 0.01
gam= 1.0
0 0.000 0.010 0.000e+00 0.010
15 0.150 0.010 1.116e-01 0.080
30 0.300 0.010 1.232e-01 0.150
45 0.450 0.010 1.251e-01 0.240
60 0.600 0.010 1.094e-01 0.310
75 0.750 0.010 1.117e-01 0.380
90 0.900 0.010 1.232e-01 0.450
105 1.050 0.010 1.251e-01 0.540
120 1.200 0.010 1.094e-01 0.610
135 1.350 0.010 1.117e-01 0.680
150 1.500 0.010 1.232e-01 0.750
delmax= 1.321e-01

```



...



Содержание отчета:

- 1) Постановка задачи с записью уравнения, НУ и ГУ для вашего варианта.
- 2) Разностная схема и численный алгоритм на ее основе.
- 3) Тексты главной программы и дополнительных процедур (кроме графического модуля).
- 4) Текстовые и графические результаты одного запуска программы.
- 5) Таблицы и графики зависимостей $\delta_{max}(h)$ и $\delta_{max}(\tau)$. Их анализ и вывод о порядках точности схемы по времени и по пространству.
- 6) Определение скорости УВ из результатов численного решения, сравнение с точным значением.