Рахматуллаев - УрЧаП 3.9

2 июня 2022 г.

1 Неявный метод

Решаем задачу 3.9 из сборника Дьяченко.

$$\begin{cases} \frac{\partial u}{\partial t} = u^2 \left(\frac{\partial^2 u}{\partial x^2} + \alpha \right) \\ \alpha \in \{2.0, 0.0\} \\ x = 0 : \frac{\partial u}{\partial x} = 0 \\ x = 1 : \frac{\partial u}{\partial x} + u^2 = 0 \\ t = 0 : u = x^2 \left(1 - x^2\right) + 0.1 \end{cases} \tag{1}$$

Запишем разностную схему:

$$\frac{u_m^{n+1} - u_m^n}{\tau} = (u_m^n)^2 \left(\frac{u_{m+1}^{n+1} - 2u_m^{n+1} + u_{m-1}^{n+1}}{h^2} + \alpha \right)$$

Перепишем в виде, удобным для метода прогонки:

$$-\frac{\tau\sigma}{h^2} u_{m+1}^{n+1} + \left(1 + \frac{2\tau\sigma}{h^2}\right) u_m^{n+1} - \frac{\tau\sigma}{h^2} u_{m+1}^{n+1} = u_m^n + \alpha\tau (u_m^n)^2,$$

где $\sigma(n, m) = (u_m^n)^2$

$$a = -\frac{\tau\sigma}{h^2},$$

$$b = 1 + \frac{2\tau\sigma}{h^2}$$

$$c = -\frac{\tau\sigma}{h^2}$$

$$\epsilon = u_m^n + \alpha\tau (u_m^n)^2$$

И самое главное:

$$u_m^{n+1} = \alpha_m u_{m+1}^{n+1} + \beta_m = -\frac{a_m^n}{b_m^n + c_m^n \alpha_{m-1}} u_{m+1}^{n+1} + \frac{\epsilon_m^n - c_m^n \beta_{m-1}^n}{b_m^n + c_m^n \alpha_{m-1}}$$

Из левого краевого, $u_1^{n+1} = u_0^{n+1}$, т.е.:

$$\alpha_0 = 1, \quad \beta_0 = 0$$

Из правого краевого, записывая:

$$\frac{u_M^{n+1} - u_{M-1}^{n+1}}{h} + (u_M^n)^2 = 0$$

получаем

$$u_M^{n+1} - u_{M-1}^{n+1} = h \cdot (u_M^n)^2$$

$$u_M^{n+1} = \frac{h(u_M^n)^2 + \beta_{M-1}}{1 - \alpha_{M-1}}$$

1.1 Анализ устойчивости

$\Delta_m = \delta_m^n, \quad \delta_m = \delta_{m+1}^n$ Применим метод замороженных коэффициентов. Из решения $v_m^n = u_m^n + \delta_m^n$, $v_m^{n+1} = u_m^{n+1} + \delta_m$ вычтем решение u_m^n.

Обозначая

$$U = \frac{u_{m+1}^{n+1} - 2u_m^{n+1} + u_{m-1}^{n+1}}{h^2}$$

$$D = \frac{\delta_{m+1}^{n+1} - 2\delta_m^{n+1} + \delta_{m-1}^{n+1}}{h^2}$$

Получим:

$$\frac{\delta_m^{n+1} - \delta_m^n}{\tau} = (U + D + \alpha)(u_m^n + \delta_m^n)^2 - (U + \alpha)(u_m^n)^2 = \ = 2u_m^n \delta_m^n (U + \alpha) + D(u_m^n)^2 + O(\delta^2) \quad (2)$$

Замораживая оставшиеся коэффициенты $u_{m+1}^{n+1}, u_m^{n+1}, u_{m-1}^{n+1}$ в произвольной фиксированной точке u, получаем, что $U = 0$, и итоговая схема имеет вид(отбрасывая члены второго порядка малости по δ):

$$\frac{\delta_m^{n+1} - \delta_m^n}{\tau} - a^2 \frac{\delta_{m+1}^{n+1} - 2\delta_m^{n+1} + \delta_{m-1}^{n+1}}{h^2} = 0 \quad (3)$$

Анализируем далее спектральную устойчивость полученной схемы. Подставляя $\delta_m^n = \lambda^n e^{im\phi}$, и сразу разделив на $\lambda^n e^{im\phi}$:

$$\frac{\lambda - 1}{\tau} = a \frac{e^{i\phi} + e^{-i\phi} - 2}{h^2} = \frac{2a(\cos\phi - 1)}{h^2} \quad (4)$$

Откуда

$$\lambda = 1 - \frac{2a\tau}{h^2}(1 - \cos\phi) \quad \Rightarrow$$

$$|\lambda|^2 = \left(1 - \frac{2a\tau}{h^2}(1 - \cos\phi)\right)^2 \leq 1$$

При $\frac{2a\tau}{h^2} \le 1$ последнее равенство выполняется для любого ϕ. Таким образом, при указанных условиях на τиh схема спектрально устойчива.

2 Явный метод

Схема:

$$\frac{u_{n+1}^m - u_n^m}{\tau} = (u_n^m)^2 \left(\frac{u_n^{m+1} - 2u_n^m + u_n^{m-1}}{h^2} + \alpha \right)$$

$$u_{n+1}^m = u_n^m + \tau (u_n^m)^2 \left(\frac{u_n^{m+1} - 2u_n^m + u_n^{m-1}}{h^2} + \alpha \right)$$

Правый конец:

$$\frac{u_{n+1}^M - u_{n+1}^{m-1}}{h} + (u_n^M)^2 = 0 \quad \Rightarrow \quad u_{n+1}^M = u_{n+1}^{m-1} - h(u_n^M)^2 = 0$$

Левый конец:

$$u_0^{n+1} = u_{n+1}^1$$

3 Код

3.0.1 Параметры запуска:

[1]:
```
N = 10000
M = 100
h = 1/M
dt = 1/N
U0 = []
alpha=0
U = []
pth = f"{N}_{M}_{alpha}"
```

3.0.2 методы

[2]:
```
import numpy as np
from matplotlib import pyplot as plt
from tqdm.notebook import tqdm
from mpl_toolkits.mplot3d import Axes3D
from copy import copy
from celluloid import Camera
import os
```

[3]:
```
def coeffs(n, dt = dt, h=h, U = U, alpha=alpha):
    S = np.array(U[n])**2
    A = -dt*S/h**2
    B = 1 + 2*dt*S/h**2
    C = -dt*S/h**2
    F = np.array(U[n]) + alpha*S*dt
```

```
    return A, B, C, F

def TDMA(a,b,c,f):
    a, b, c, f = tuple(map(lambda k_list: list(map(float, k_list)), (a, b, c,␣
 ↪f)))

    alpha = [-a[0] / b[0]]
    beta = [f[0] / b[0]]
    n = len(f)
    x = [0]*n

    for i in range(1, n):
        alpha.append(-a[i]/(c[i]*alpha[i-1] + b[i]))
        beta.append((f[i] - c[i]*beta[i-1])/(c[i]*alpha[i-1] + b[i]))

    x[n-1] = beta[n - 1]

    for i in range(n-1, 0, -1):
        x[i - 1] = alpha[i - 1]*x[i] + beta[i - 1]

    return x

def gif_(U,fold, method, alpha=alpha, N=N, M=M):
    fig, ax = plt.subplots()
    ax.set_ylim(0, 0.5)
    camera = Camera(fig)

    for i in tqdm(range(0, len(U), int(np.ceil(len(U)/100)))):
        ax.plot(U[i], color = 'b')
        camera.snap()

    animation = camera.animate()
    if not os.path.isdir(pth):
        os.mkdir(pth)

    with open(f"{pth}/params.txt", 'w') as X__:
        X__.write(f'N = {N}, M = {M}, alpha = {alpha}'+'\n')

    animation.save(f"{pth}/celluloid_{method}plisit{alpha}.gif", writer =␣
 ↪'imagemagick')

    with open(f"{pth}/{method}plicit{alpha}.txt", 'w') as X__:
        for LINE in U:
            X__.write(str(LINE)[1:-1]+'\n')
```

3.1 Неявный метод

[4]:
```
#зполняем слой при t=0
for m in range(0, M+1):#np.arange(0, 1+h, h):
    x = m*h
    U0.append(x**2 *(1-x**2) + 0.1)
```

[5]:
```
U = [U0]
for n in tqdm(range(0, N)):

    A, B, C, f = coeffs(n, U=U)

    C[0]=0
    B[0]=1
    A[0]=-1

    C[M]=1
    B[M]=-1
    A[M]=0

    #f = copy(U[n])
    f[0] = 0
    f[M] = h*(U[n][M])**2

    x = TDMA(A,B,C,f)
    U.append(x)
    #if not n%1000:
    #    plt.plot(x)
```

```
  0%|          | 0/10000 [00:00<?, ?it/s]
```

График решения в последний момент времени:

[6]:
```
#решение в последний момент времени
fig, ax = plt.subplots()
ax.set_ylim(0, 0.5)
ax.plot(U[-1])
```

[6]:
```
[<matplotlib.lines.Line2D at 0x21a72ecac48>]
```

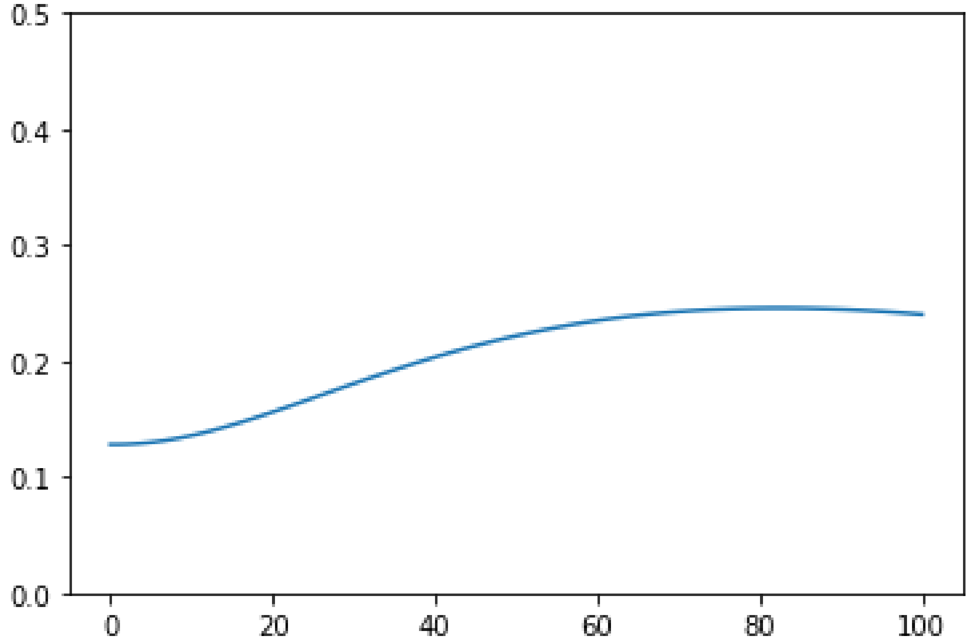
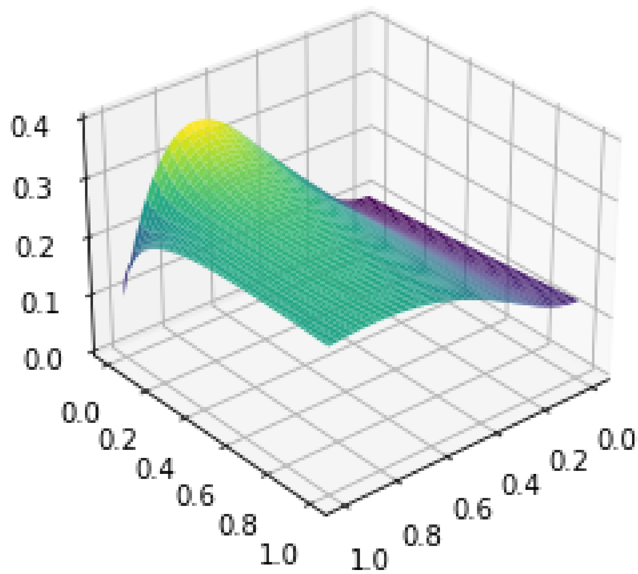


График в пространстве u, x, t:

[7]:
```
X = np.arange(0, 1+h, h)
Y = np.arange(0, 1+dt, dt)
X, Y = np.meshgrid(X, Y)
fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
ax.set_zlim(0, 0.4)
ax.view_init(30, 50)
ax.plot_surface(X, Y, np.array(U), cmap=plt.cm.viridis)
```

[7]:
```
<mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x21a72bd3d88>
```

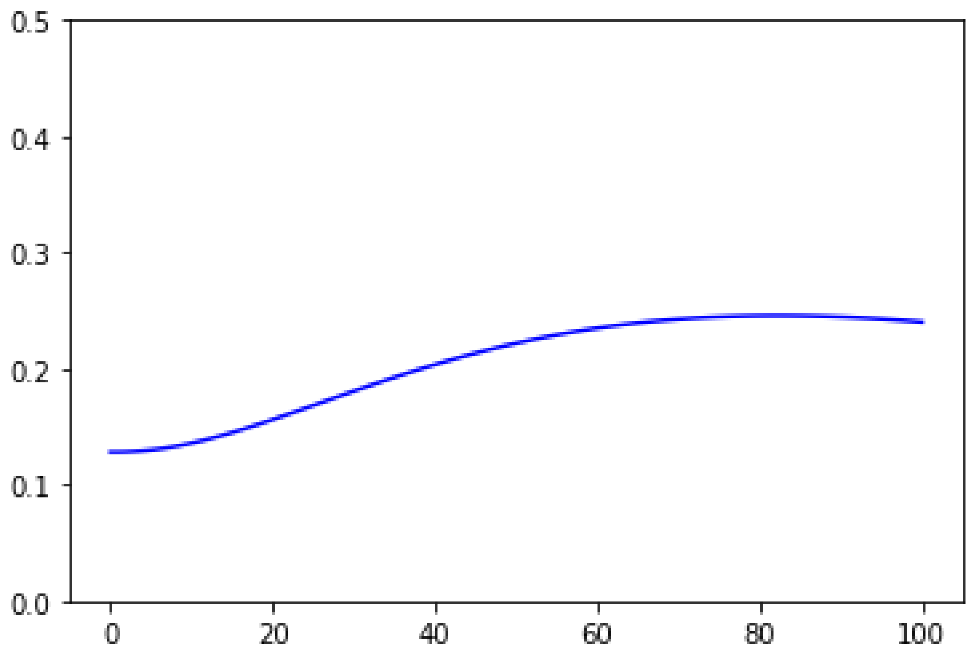


Gif анимация решения и вывод в файл:

[8]:
```
gif_(U, pth, method = 'im')
```

```
  0%|          | 0/100 [00:00<?, ?it/s]
```

```
MovieWriter imagemagick unavailable; using Pillow instead.
```



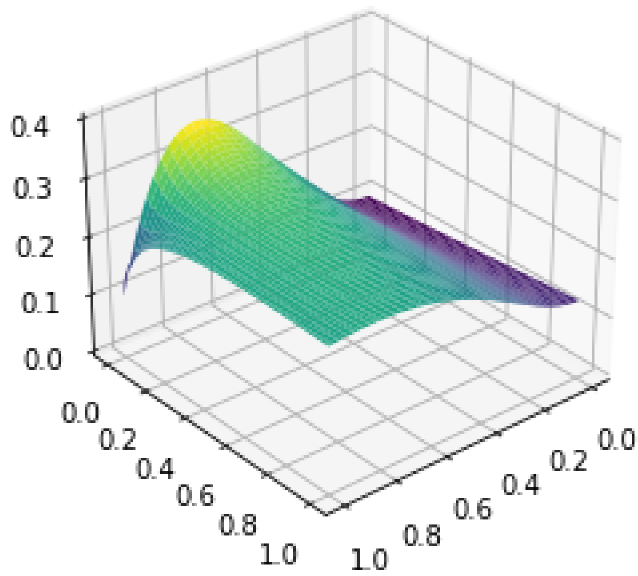
4 Явный метод

[9]:
```
#зполняем слой при t=0
Uu = []
Uu.append(U0)

## прямое по времени заполнение слоев
try:
    for n in tqdm(range(1, N+1)):
        if len(Uu) == n:
            Uu.append([0]*(M+1))
        for m in range(M-1, 0, -1):
            Uu[n][m] = Uu[n-1][m] + dt * (Uu[n-1][m])**2 * ((Uu[n-1][m-1] -␣
↪2*Uu[n-1][m] + Uu[n-1][m+1])/h**2 + alpha)
        Uu[n][0] = Uu[n][1]
        Uu[n][M] = Uu[n][M-1] - h*(Uu[n-1][M])**2
except OverflowError as er:
    print(er)
```

```
  0%|          | 0/10000 [00:00<?, ?it/s]
```

[10]:
```
try:
    X = np.arange(0, 1+h, h)
    Y = np.arange(0, 1+dt, dt)
    X, Y = np.meshgrid(X, Y)
    fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
    ax.set_zlim(0, 0.4)
    ax.view_init(30, 50)
    ax.plot_surface(X, Y, np.array(Uu), cmap=plt.cm.viridis)
except ValueError as er:
    print(er)
```



[11]:
```
gif_(Uu, pth, method = 'ex')
```

```
  0%|          | 0/100 [00:00<?, ?it/s]
```

```
MovieWriter imagemagick unavailable; using Pillow instead.
```