

Задача №7

Двумерное уравнение теплопроводности

Постановка задачи:

Используя метод переменных направлений, решить краевую задачу:

$$\begin{cases} \frac{\partial u}{\partial t} = \Delta u, 0 < x < \pi/2, 0 < y < \pi/2, t > 0 \\ \frac{\partial u}{\partial x} \Big|_{x=0} = \frac{\partial u}{\partial x} \Big|_{x=\pi} = 0 \\ \frac{\partial u}{\partial y} \Big|_{y=0} = \frac{\partial u}{\partial y} \Big|_{y=\pi/2} = 0 \\ u \Big|_{t=0} = \cos 4x \cos 2y \end{cases} \quad (1)$$

Аналитическое решение задачи

Будем искать решение задачи в виде:

$$u(x, y, t) = T(t)V(x, y)$$

Тогда, решая исходную задачу методом разделения переменных, получаем:

$$\frac{T'(t)}{T(t)} = \frac{V''(x, y)}{V(x, y)} = -\lambda$$

Имеем задачу Штурма-Лиувилля для V :

$$\begin{cases} V'' + \lambda V = 0 \\ \frac{\partial V}{\partial x} \Big|_{x=0} = \frac{\partial V}{\partial x} \Big|_{x=\pi/2} = 0 \\ \frac{\partial V}{\partial y} \Big|_{y=0} = \frac{\partial V}{\partial y} \Big|_{y=\pi/2} = 0 \end{cases}$$

Теперь, представляя функцию V в виде $V(x, y) = X(x)Y(y)$ и применяя метод разделения переменных, получим две задачи Штурма-Лиувилля на отрезке:

$$\begin{cases} X'' + \nu X = 0 \\ X'|_{x=0} = X'|_{x=\pi/2} = 0 \end{cases} \Rightarrow X = \cos(\sqrt{\nu}x)$$

$$\begin{cases} Y'' + \mu Y = 0 \\ \frac{\partial Y}{\partial y}|_{y=0} = \frac{\partial Y}{\partial y}|_{y=\pi/2} = 0 \end{cases} \Rightarrow Y = \cos(\sqrt{\mu}y)$$

где $\nu = 4n^2$, $\mu = 4m^2$, $\lambda = \mu + \nu$

Тогда для функции V получаем: $V_{nm} = \cos(2nx)\cos(2my)$

Из начальных условий получаем, что $n = 2, m = 1 \Rightarrow \lambda = 20$.

Для $T(t)$ нужно решить следующую задачу:

$$\begin{cases} T'_{n,m} + \lambda_{n,m}T_{n,m} = 0, & t > 0, \\ T_{n,m}|_{t=0} = \varphi_{n,m}, \end{cases} \quad (2)$$

где $\varphi(x, y) = \cos(4x)\cos(2y)$, а $\varphi_{n,m}$ - коэффициенты разложения функции $\varphi(x, y)$ в ряд Фурье по системе функций $V_{n,m}(x, y)$.

Тогда, с учетом, что решение представляется в виде:

$$V = \sum_{n=0}^{\infty} \sum_{m=1}^{\infty} V_{nm}(x, y)T_{nm}(t)$$

Получаем решение:

$$u(x, y, t) = \cos(4x)\cos(2y)e^{-20t}$$

График аналитического решения

Построим график аналитического решения:

```
In [37]: import numpy as np
import matplotlib.pyplot as plt
from math import*
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from IPython.display import clear_output
```

```
In [38]: N, M, T = 100, 100, 100
x_start, x_end = 0, pi/2
y_start, y_end = 0, pi/2
t_start, t_end = 0, 1
tau = t_end / (T)
x = np.linspace(x_start, x_end, N)
y = np.linspace(y_start, y_end, M)
t = np.linspace(t_start, t_end, T)
```

```
In [39]: u_a = np.zeros((N,M,T))
print('Calculating...')
for n in range(0,N):
    clear_output(wait = True)
    print(f'Progress: {round(n/(N-1)*100)}%')
    for m in range(0,M):
        for j in range(0,T):
            u_a[n,m,j]=cos(4*x[n])*cos(2*y[m])*exp(-20*j*tau)
print('Done!')
```

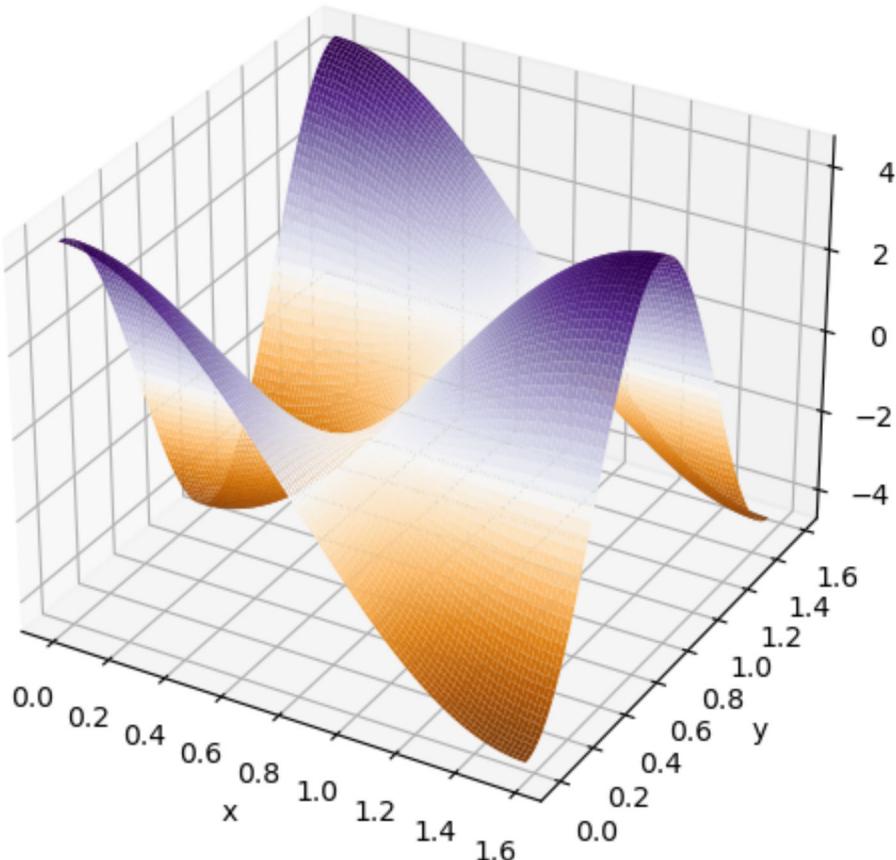
Progress: 100%

Done!

```
In [40]: x1,y1 = np.meshgrid(x,y)

fig1 = plt.figure(figsize=(8,6))
o1 = fig1.add_subplot(111, projection = '3d')
p1 = o1.plot_surface(x1, y1, u_a[:, :, int(T/2)], rstride = 1, cstride = 1, cmap =
# o1.view_init(30, 90)
plt.title('График аналитического решения в момент времени t = ' + str(round(tau*
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

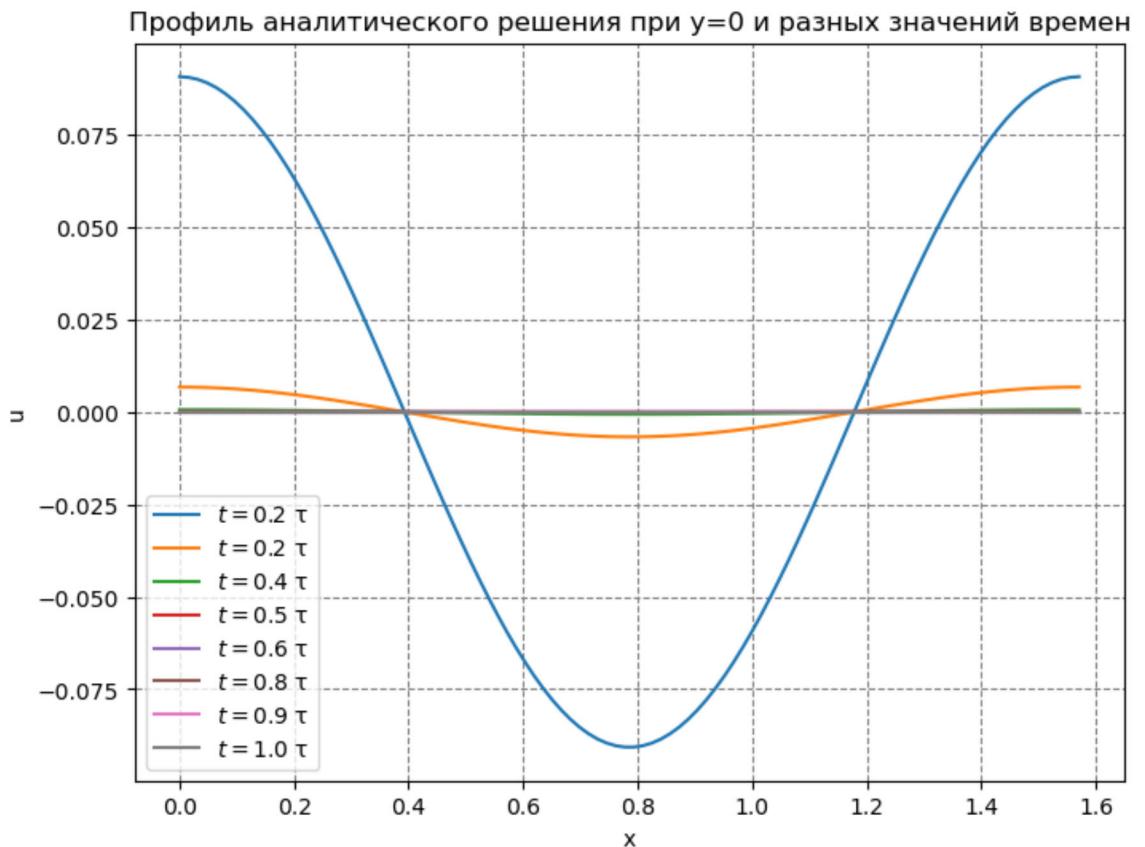
График аналитического решения в момент времени $t = 0.5$



```
In [41]: fig,ax=plt.subplots(figsize=(8,6))
x = np.linspace(x_start,x_end,N)
plt.xlabel('x')
plt.ylabel('u')
plt.title('Профиль аналитического решения при y=0 и разных значений времен')

plt.plot(x, u_a[:,0,int(T/8)],label='$t=' + str(round(tau*T/6,1)) + ' \tau$')
plt.plot(x, u_a[:,0,int(2*T/8)],label='$t=' + str(round(tau*2*T/8,1)) + ' \tau$')
plt.plot(x, u_a[:,0,int(3*T/8)],label='$t=' + str(round(tau*3*T/8,1)) + ' \tau$')
plt.plot(x, u_a[:,0,int(4*T/8)],label='$t=' + str(round(tau*4*T/8,1)) + ' \tau$')
plt.plot(x, u_a[:,0,int(5*T/8)],label='$t=' + str(round(tau*5*T/8,1)) + ' \tau$')
plt.plot(x, u_a[:,0,int(6*T/8)],label='$t=' + str(round(tau*6*T/8,1)) + ' \tau$')
plt.plot(x, u_a[:,0,int(7*T/8)],label='$t=' + str(round(tau*7*T/8,1)) + ' \tau$')
plt.plot(x, u_a[:,0,T-1],label='$t=' + str(round(tau*8*T/8,1)) + ' \tau$')
ax.grid(which='major', color = 'gray',ls='--')
ax.legend(loc='best')
```

Out[41]: <matplotlib.legend.Legend at 0x186582cc970>



Таким образом, можно видеть, что с течением времени перенос тепла будет уменьшаться. Причем скорость уменьшения определяется экспоненциальным множителем, а форма изменяться не будет и определяется множителями синусом и косинусом.

Численное решение

Разностная схема

Введем разностную сетку в области $D = G \times [0, T]$,

$G = \{(x, y) : 0 \leq x \leq \pi/2, 0 \leq y \leq \pi/2\}$ с N_x и N_y числом узлов вдоль оси x и y и шагами h_x и h_y соответственно, с шагом τ и числом узлов M по времени:

$$\begin{cases} \omega_x \equiv \{x_n = nh_x; n = 0, 1, \dots, N; h_x N_x = \pi/2\} \\ \omega_y \equiv \{y_m = mh_y; m = 0, 1, \dots, N_y; h_y N_y = \pi/2\} \\ \omega_t \equiv \{t_k = k\tau; k = 0, 1, \dots, M; \tau M = T\} \end{cases} \quad (3)$$

$$\omega_{xyt} = \omega_x \times \omega_y \times \omega_t$$

При решении задачи положим $T = 0.2$. Введем сеточную функцию:
 $u_{n,m}^k = u(x_n, y_m, t_k)$.

Запишем разностную аппроксимацию оператора Лапласа:

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad (4)$$

$$\Lambda u_{n,m}^k = \Lambda_x u_{n,m}^k + \Lambda_y u_{n,m}^k \quad (5)$$

$$\Lambda_x u_{n,m}^k = \frac{u_{n+1,m}^k - 2u_{n,m}^k + u_{n-1,m}^k}{h_x^2} \quad (6)$$

$$\Lambda_y u_{n,m}^k = \frac{u_{n,m+1}^k - 2u_{n,m}^k + u_{n,m-1}^k}{h_y^2} \quad (7)$$

Тогда уравнение для сеточной функции неявной схемы выглядит следующим образом:

$$\frac{u_{n,m}^{k+1} - u_{n,m}^k}{\tau} = \Lambda u_{n,m}^{k+1} \quad (8)$$

Аппроксимируем начальные и граничные условия задачи (1):

$$u_{n,m}^0 = \cos(4nh_x) \cos(2mh_y), \quad n = 0, 1, \dots, N_x, \quad m = 0, 1, \dots, N_y \quad (9)$$

$$\frac{u_{1,m}^k - u_{0,m}^k}{h_x} = \frac{u_{N_x,m}^k - u_{N_x-1,m}^k}{h_x} = 0, \quad m = 0, 1, \dots, N_y, \quad k = 0, 1, \dots, M \quad (10)$$

$$\frac{u_{n,1}^k - u_{n,0}^k}{h_y} = \frac{u_{n,N_y}^k - u_{n,N_y-1}^k}{h_y} = 0, \quad n = 0, 1, \dots, N_x, \quad k = 0, 1, \dots, M \quad (11)$$

Метод переменных направлений

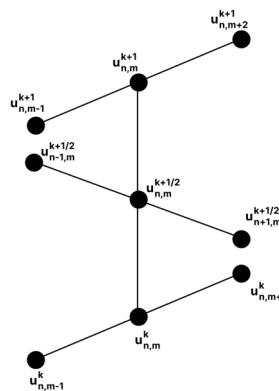
При решении будем использовать метод переменных направлений, для чего введем промежуточный временной слой $t_{k+1/2} = \tau(k + 1/2)$. Тогда на первом полуслое (от t_k до $t_{k+1/2}$) будем решать задачу, неявную по направлению x и явную по направлению y , а на втором полуслое (от $t_{k+1/2}$ до t_{k+1}) - явную по направлению x и неявную по направлению y . Данная схема абсолютно устойчива.

Для первого и второго полуслоя задачи ставятся следующим образом:

$$\frac{u_{n,m}^{k+1/2} - u_{n,m}^k}{\tau/2} = \Lambda_x u_{n,m}^{k+1/2} + \Lambda_y u_{n,m}^k \quad (12)$$

$$\frac{u_{n,m}^{k+1} - u_{n,m}^{k+1/2}}{\tau/2} = \Lambda_x u_{n,m}^{k+1/2} + \Lambda_y u_{n,m}^{k+1} \quad (13)$$

Чтобы определить $u_{n,m}^{k+1/2}$ нужно решить задачи (12), (13) методом прогонки.



Метод прогонки

Рассмотрим задачу (12). Раскроем оператор Лапласа и умножим на $\tau/2$ с обеих сторон:

$$u_{n,m}^{k+1/2} - u_{n,m}^k = \frac{\tau}{2} \frac{u_{n+1,m}^{k+1/2} - 2u_{n,m}^{k+1/2} + u_{n-1,m}^{k+1/2}}{h_x^2} + \frac{\tau}{2} \frac{u_{n,m+1}^k - 2u_{n,m}^k + u_{n,m-1}^k}{h_y^2} \quad (14)$$

Обозначим неизвестные - $u_{n+1,m}^{k+1/2}, u_{n,m}^{k+1/2}, u_{n-1,m}^{k+1/2}$ - соответственно y_{n+1}, y_n, y_{n-1} .

Перепишем уравнение (14) в виде:

$$A_n y_{n-1} + B_n y_n + C_n y_{n+1} = F_n, \quad (15)$$

где

$$A_n = \frac{\tau}{2h_x^2} \quad (16)$$

$$B_n = -1 - \frac{\tau}{h_x^2} \quad (17)$$

$$C_n = \frac{\tau}{2h_x^2} \quad (18)$$

$$F_n = -\frac{\tau}{2} \frac{u_{n,m+1}^k - 2u_{n,m}^k + u_{n,m-1}^k}{h_y^2} - u_{n,m}^k \quad (19)$$

Каждое предыдущее значение y_n можно выразить через последующее (формула обратной прогонки):

$$y_n = \alpha_{n+1} y_{n+1} + \beta_{n+1} \quad (20)$$

Подставим выражение для y_{n-1} в (15):

$$A_n(\alpha_n y_n + \beta_n) + B_n y_n + C_n y_{n+1} = F_n \quad (21)$$

$$y_n = -\frac{C_n}{\alpha_n A_n + B_n} y_{n+1} + \frac{F_n - A_n \beta_n}{\alpha_n A_n + B_n} \quad (22)$$

Сравнивая выражения (20) и (22), получаем:

$$\alpha_{n+1} = -\frac{C_n}{\alpha_n A_n + B_n} \quad (23)$$

$$\beta_{n+1} = \frac{F_n - A_n \beta_n}{\alpha_n A_n + B_n} \quad (24)$$

Мы получили формулы прямой прогонки. Из граничных условий мы получаем, что:

$$\frac{y_1 - y_0}{h_x} = 0 \quad (25)$$

$$y_0 = \alpha_1 y_1 + \beta_1 = y_1 \Rightarrow \alpha_1 = 1, \beta_1 = 0 \quad (26)$$

$$\frac{y_{N_x} - y_{N_x-1}}{h_x} = 0 \quad (27)$$

$$y_{N_x-1} = \alpha_{N_x} y_{N_x} + \beta_{N_x} = y_{N_x} \Rightarrow y_{N_x} = \frac{\beta_{N_x}}{1 - \alpha_{N_x}} \quad (28)$$

Аналогично мы решаем неявную задачу по y . Поскольку граничные условия в нашей задаче совпадают, разности не будет.

Порядок аппроксимации

Разложим разностный оператор Δ_x в ряд Тейлора

$$\frac{u_{n+1,m}^k - 2u_{n,m}^k + u_{n-1,m}^k}{h_x^2} = \frac{1}{h_x^2} \left(h_x^2 u_{n,m}^{k(2)} + \frac{h_x^4}{12} u_{n,m}^{k(4)} \right) \sim O(h_x^2)$$

$$\frac{u_{n,m+1}^k - 2u_{n,m}^k + u_{n,m-1}^k}{h_y^2} = \frac{1}{h_y^2} \left(h_y^2 u_{n,m}^{k(2)} + \frac{h_y^4}{12} u_{n,m}^{k(4)} \right) \sim O(h_y^2)$$

То есть порядок аппроксимации по пространственным координатам компонентам равен $O(|h|^2)$.

Чтобы найти порядок аппроксимации по временной компоненте разложим в ряд Тейлора до соответствующих порядков производные по времени

$$\begin{cases} u^{k+1} = u^{k+1/2} + \frac{\tau}{2} u_t^{k+1/2} + \frac{1}{2} \frac{\tau^2}{4} u_{tt}^{k+1/2} + \frac{1}{2} \frac{\tau^3}{8} u_{ttt}^{k+1/2} + O(\tau^4) \\ u^k = u^{k+1/2} - \frac{\tau}{2} u_t^{k+1/2} + \frac{1}{2} \frac{\tau^2}{4} u_{tt}^{k+1/2} - \frac{1}{2} \frac{\tau^3}{8} u_{ttt}^{k+1/2} + O(\tau^4) \end{cases} \quad (29)$$

Тогда в точке $t^{k+1/2}$ аппроксимация производной по времени будет выглядеть

$$\frac{u^{k+1} - u^k}{\tau} = \frac{1}{\tau} \left(\tau u_t^{k+1/2} + \frac{1}{3} \frac{\tau^3}{8} u_{tt}^{k+1/2} + O(\tau^4) \right) = u_t^{k+1/2} + O(\tau^2)$$

Невязка равна $O(|h|^2 + \tau^2)$.

Устойчивость

Устойчивость схемы по начальным данным будем исследовать с помощью спектрального метода Ньютона. Ищем решение в виде $u_{n,m}^k = \lambda_1^k e^{i(\alpha n + \beta m)}$.

Подставим решение в разностное уравнение и получим

$$\sqrt{\lambda_1^k} = \frac{1 - \frac{2\tau}{h_y^2} \sin^2 \frac{\beta}{2}}{1 + \frac{2\tau}{h_x^2} \sin^2 \frac{\alpha}{2}} < 1, \quad \forall \tau, h_x, h_y, \alpha, \beta.$$

Аналогично $u_{n,m}^k = \lambda_2^k e^{i(\alpha n + \beta m)}$.

$$\sqrt{\lambda_2^k} = \frac{1 - \frac{2\tau}{h_x^2} \sin^2 \frac{\alpha}{2}}{1 + \frac{2\tau}{h_y^2} \sin^2 \frac{\beta}{2}} < 1, \quad \forall \tau, h_x, h_y, \alpha, \beta.$$

Значит для каждого из уравнений выполняется критерий Неймана. Из $\lambda_1 \lambda_2 < 1, \forall \tau, h_x, h_y, \alpha, \beta$ следует, что он выполняется и переходе с j -ого слоя на $j+1$ -ый. Значит схема переменных направлений безусловно устойчива.

Программная реализация численного решения

In [42]:

```
N, M, T = 100, 100, 100

h_x=x_end/(N-1)
h_y=y_end/(M-2)
tau=t_end/T
gamma_1 = tau / (h_x**2)
gamma_2 = tau / (h_y**2)

x = np.linspace(x_start,x_end,N)
y = np.linspace(y_start-h_y/2,y_end+h_y/2,M)
t = np.linspace(t_start,t_end,T)
u = np.zeros((N,M,2*T+1))

for n in range(0,N):
    for m in range(0,M):
        u[n,m,0]=cos(4*x[n])*cos(2*y[m])
```

In [43]:

```
def progonka_X(m,j):
    alpha = np.zeros(N)      # последний элемент имеет индекс N-1
    beta = np.zeros(N)
    alpha[1] = 0
    beta[1] = 0
    A_x = 0.5 * gamma_1
    B_x = 1 + gamma_1
    C_x = 0.5 * gamma_1

    u[0,m,j] = 0
    for n in range (1,N-1): # прямая прогонка
        F_x = 0.5 * gamma_2*(u[n,m-1,j-1]+u[n,m+1,j-1])+(1-gamma_2)*u[n,m,j-1]
        # проверить индексы по j
        alpha[n+1] = C_x / (B_x - A_x * alpha[n])
        beta[n+1] = (F_x + A_x * beta[n]) / (B_x-A_x * alpha[n])

    u[N-1,m,j] = 0
    for n in range(N-1,0,-1): # обратная прогонка
        u[n-1,m,j]=alpha[n] * u[n,m,j] + beta[n]
```

In [44]:

```
def progonka_Y(n,j):
    alpha=np.zeros(M)
    beta = np.zeros(M)
    alpha[1] = 1
    beta[1] = 0
    A_y = 0.5 * gamma_2
    B_y= 1 + gamma_2
    C_y= 0.5 * gamma_2
    for m in range (1,M-1): # прямая прогонка
        F_y = 0.5 *gamma_1*(u[n-1,m,j-1]+u[n+1,m,j-1])+(1-gamma_1)*u[n,m,j-1]
        alpha[m+1] = C_y / (B_y - A_y * alpha[m])
        beta[m+1] = (F_y + A_y * beta[m]) / (B_y-A_y * alpha[m])

    u[n,M-1,j]=beta[-1]/(1-alpha[-1]) # обратная прогонка
    for m in range(M-1,0,-1):
        u[n,m-1,j]=alpha[m] * u[n,m,j] + beta[m]
```

```
In [45]: for j in range(1,2*T+1,2):
    clear_output(wait = True)
    print(f'Progress: {round(j/(2*T)*100)}%')
    for m in range(1,M-1):
        progonka_X(m,j)
    for n in range(1,N-1):
        progonka_Y(n,j+1)
    for m in range(0,M):
        u[0,m,j+1]=0
        u[N-1,m,j+1]=0
print('Done!')
```

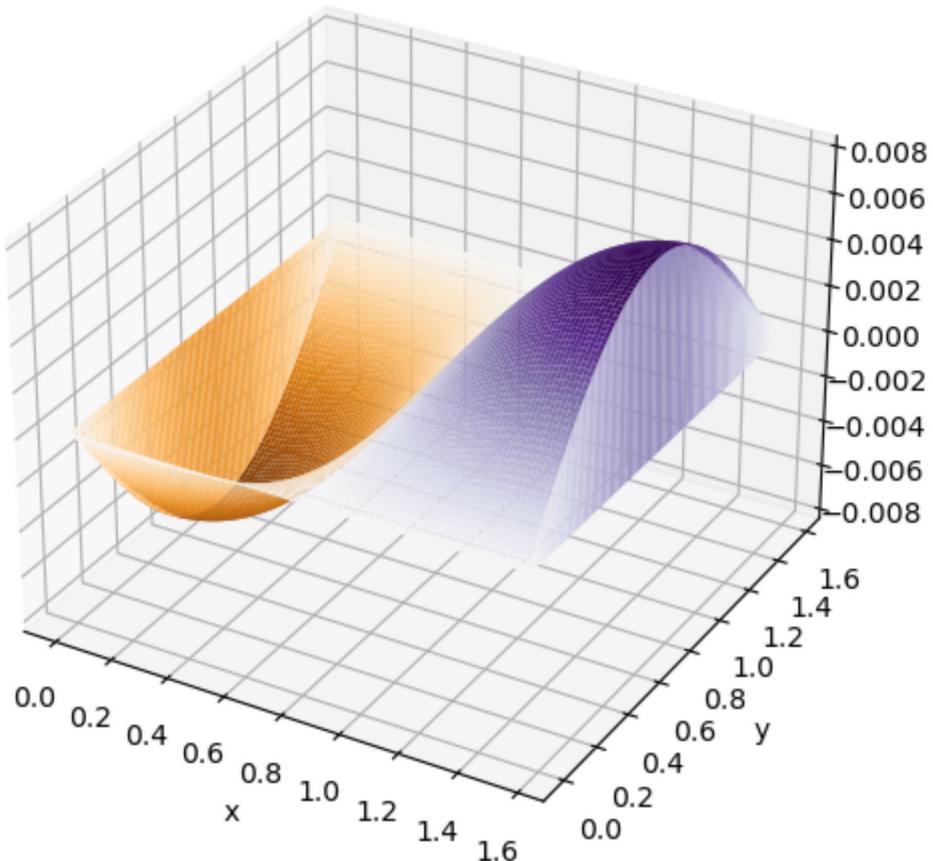
Progress: 100%

Done!

```
In [46]: x1,y1 = np.meshgrid(x,y)

fig1 = plt.figure(figsize=(9,6))
o1 = fig1.add_subplot(111, projection = '3d')
p1 = o1.plot_surface(x1, y1, u[:,:,:int(T-1)], rstride = 1, cstride = 1, cmap = cm
# o1.view_init(30, 90)
plt.title('График численного решения в момент времени t = ' +str(round(tau*T/2,2)))
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

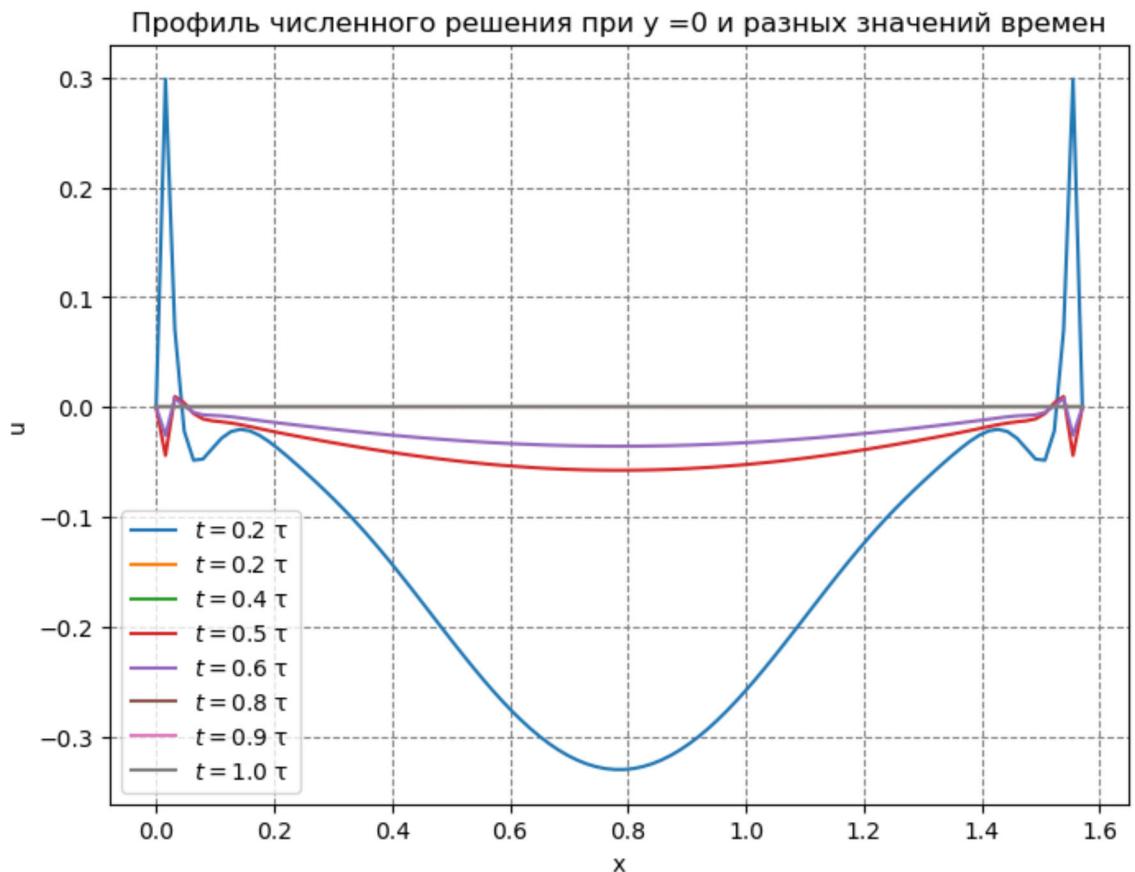
График численного решения в момент времени $t = 0.5$



```
In [47]: fig,ax=plt.subplots(figsize=(8,6))
x = np.linspace(x_start,x_end,N)
plt.xlabel('x')
plt.ylabel('u')
plt.title('Профиль численного решения при  $y = 0$  и разных значений време')

plt.plot(x, u[:,0,int(T/8)],label='$t=' + str(round(tau*T/6,1)) + ' \tau$')
plt.plot(x, u[:,0,int(2*T/8)],label='$t=' + str(round(tau*2*T/8,1)) + ' \tau$')
plt.plot(x, u[:,0,int(3*T/8)],label='$t=' + str(round(tau*3*T/8,1)) + ' \tau$')
plt.plot(x, u[:,0,int(4*T/8)],label='$t=' + str(round(tau*4*T/8,1)) + ' \tau$')
plt.plot(x, u[:,0,int(5*T/8)],label='$t=' + str(round(tau*5*T/8,1)) + ' \tau$')
plt.plot(x, u[:,0,int(6*T/8)],label='$t=' + str(round(tau*6*T/8,1)) + ' \tau$')
plt.plot(x, u[:,0,int(7*T/8)],label='$t=' + str(round(tau*7*T/8,1)) + ' \tau$')
plt.plot(x, u[:,0,T-1],label='$t=' + str(round(tau*8*T/8,1)) + ' \tau$')
ax.grid(which='major', color = 'gray',ls='--')
ax.legend(loc='best')
```

Out[47]: <matplotlib.legend.Legend at 0x18657ac35b0>



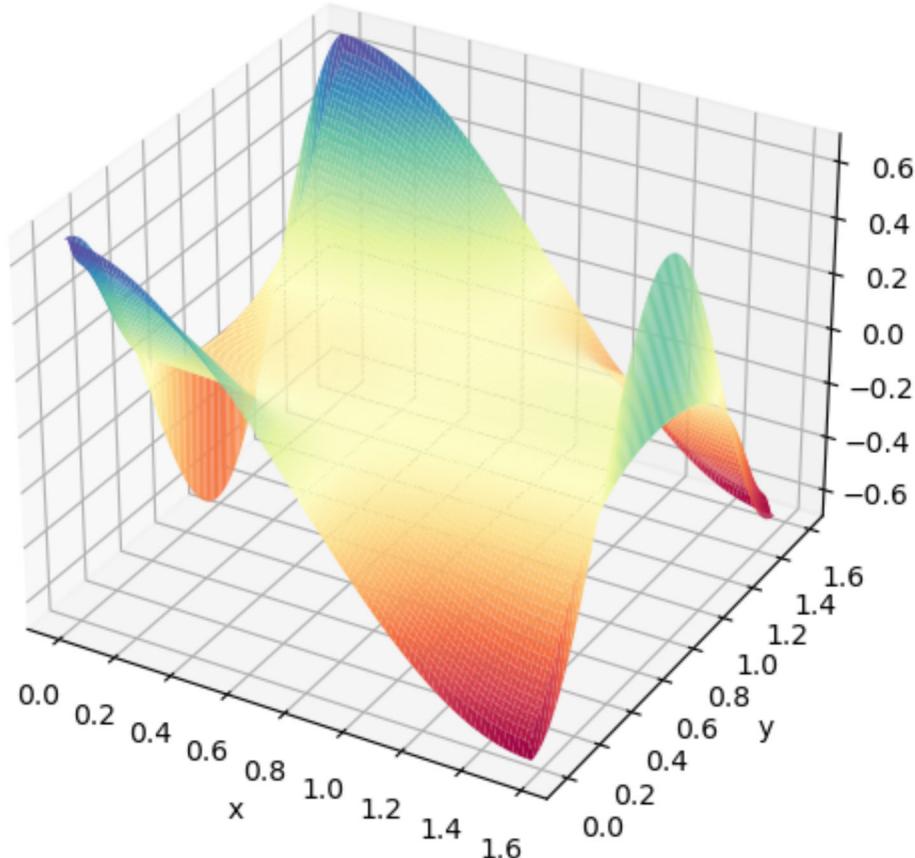
Как видно, у численного решения, также, как и у аналитического, наблюдается тенденция уменьшения с течением времени.

Погрешность

```
In [48]: x1,y1 = np.meshgrid(x,y)

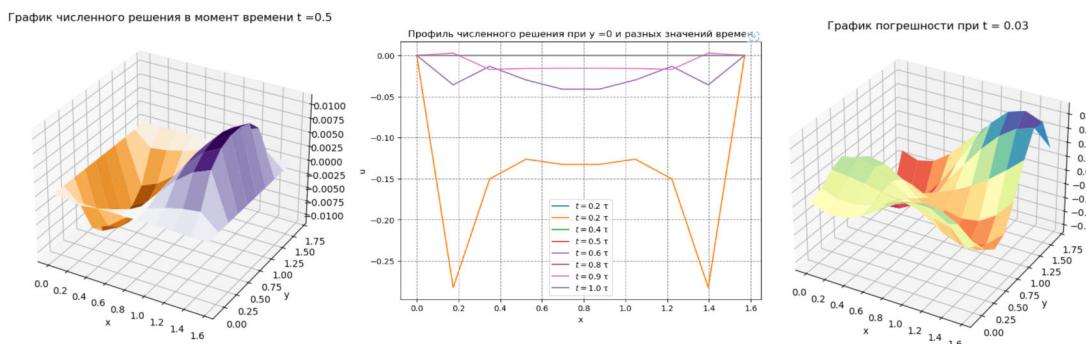
fig1 = plt.figure(figsize=(9,6))
o1 = fig1.add_subplot(111, projection = '3d')
p1 = o1.plot_surface(x1, y1, u_a[:, :, int(T/40)]-u[:, :, int(T/20)], rstride = 1, c
# o1.view_init(30, 90)
plt.title('График погрешности при t = ' +str(round(tau*T/40,2)))
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

График погрешности при $t = 0.03$



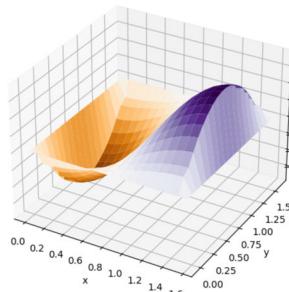
Сгущение сетки

$N = M = J = 10$



$N = M = J = 20$

График численного решения в момент времени $t = 0.5$



Профиль численного решения при $y = 0$ и разных значений времен

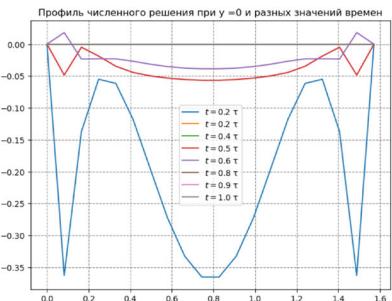
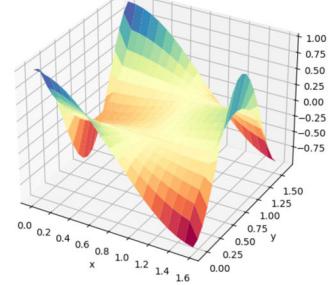
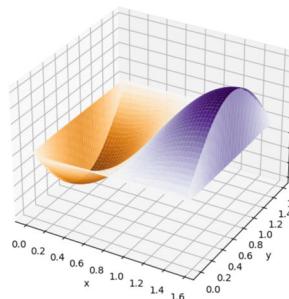


График погрешности при $t = 0.03$



$N = M = J = 50$

График численного решения в момент времени $t = 0.5$



Профиль численного решения при $y = 0$ и разных значений времен

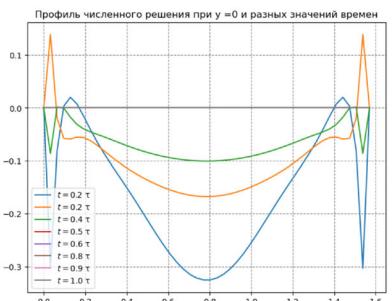
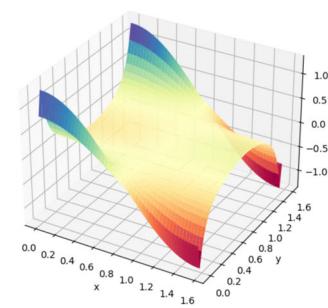
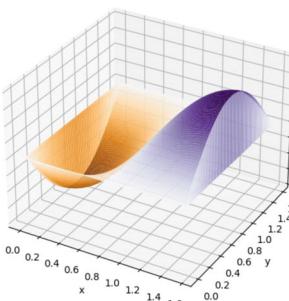


График погрешности при $t = 0.03$



$N = M = J = 100$

График численного решения в момент времени $t = 0.5$



Профиль численного решения при $y = 0$ и разных значений времен

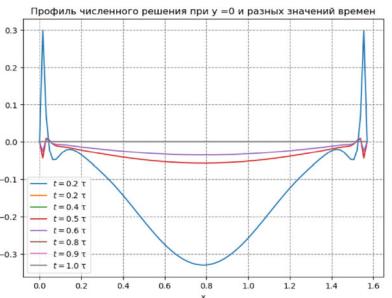


График погрешности при $t = 0.03$

