

**Московский авиационный институт**  
(национальный исследовательский университет)

Институт №8 "Информационные технологии и прикладная математика"  
Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №8**  
по дисциплине:  
**Численные методы**  
Вариант №5

Выполнил: студент группы М8О-409Б-20  
Искренкова А.В.  
Принял: Пивоваров Е.Д.  
Оценка: \_\_\_\_\_

Москва, 2023 г.

# 1. Задание

Используя схемы переменных направлений и дробных шагов, решить двумерную начально-краевую задачу для дифференциального уравнения параболического типа. В различные моменты времени вычислить погрешность численного решения путем сравнения результатов с приведенным в задании аналитическим решением  $U(x, t)$ . Исследовать зависимость погрешности от сеточных параметров  $\tau, h_x, h_y$ .

Уравнение:

$$\frac{\partial u}{\partial t} = a \frac{\partial^2 u}{\partial x^2} + a \frac{\partial^2 u}{\partial y^2}, a > 0$$

$$u(0, y, t) = \sinh(y) \exp(-3at)$$

$$u\left(\frac{\pi}{2}, y, t\right) = -\sinh(y) \exp(-3at)$$

$$u_y(x, 0, t) = \cos(2x) \exp(-3at)$$

$$u(x, \ln(2), t) = \frac{3}{4} \cos(2x) \exp(-3at)$$

$$u(x, y, 0) = \cos(2x) \sinh(y)$$

Аналитическое решение:

$$U(x, y, t) = \cos(2x) \sinh(y) \exp(-3at)$$

# 2. Решение

- Метод переменных направлений:

- 1 этап:

$$\frac{u_{i,j}^{k+0.5} - u_{i,j}^k}{\frac{\tau}{2}} = a \frac{u_{i+1,j}^{k+0.5} - 2u_{i,j}^{k+0.5} + u_{i-1,j}^{k+0.5}}{h_x^2} + a \frac{u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k}{h_y^2}$$

- 2 этап:

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k+0.5}}{\frac{\tau}{2}} = a \frac{u_{i+1,j}^{k+0.5} - 2u_{i,j}^{k+0.5} + u_{i-1,j}^{k+0.5}}{h_x^2} + a \frac{u_{i,j+1}^{k+1} - 2u_{i,j}^{k+1} + u_{i,j-1}^{k+1}}{h_y^2}$$

- Метод дробных шагов:
  - 1 дробный шаг:

$$\frac{u_{i,j}^{k+0.5} - u_{i,j}^k}{\tau} = a \frac{u_{i+1,j}^{k+0.5} - 2u_{i,j}^{k+0.5} + u_{i-1,j}^{k+0.5}}{h_x^2}$$

- 2 дробный шаг:

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k+0.5}}{\tau} = a \frac{u_{i,j+1}^{k+1} - 2u_{i,j}^{k+1} + u_{i,j-1}^{k+1}}{h_y^2}$$

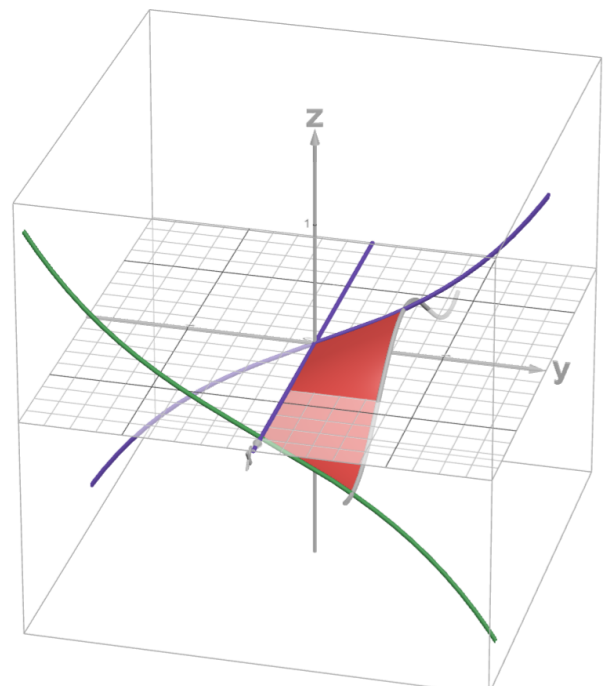
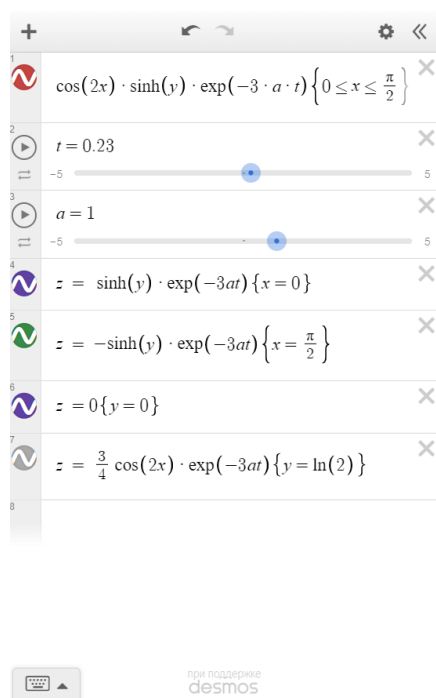
Аппроксимация граничного условия  $u_y(x, 0, t) = \cos(2x) \exp(-3at) = \varphi_3(x, 0, t)$ :

$$\frac{u_{i,1}^k - u_{i,0}^k}{h_y} = \varphi_3(x_i, 0, t^k) \Rightarrow u_{i,0}^k = u_{i,1}^k - h_y * \varphi_3(x_i, 0, t^k)$$

Погрешность между численным и аналитическим решением рассчитывается как абсолютная.

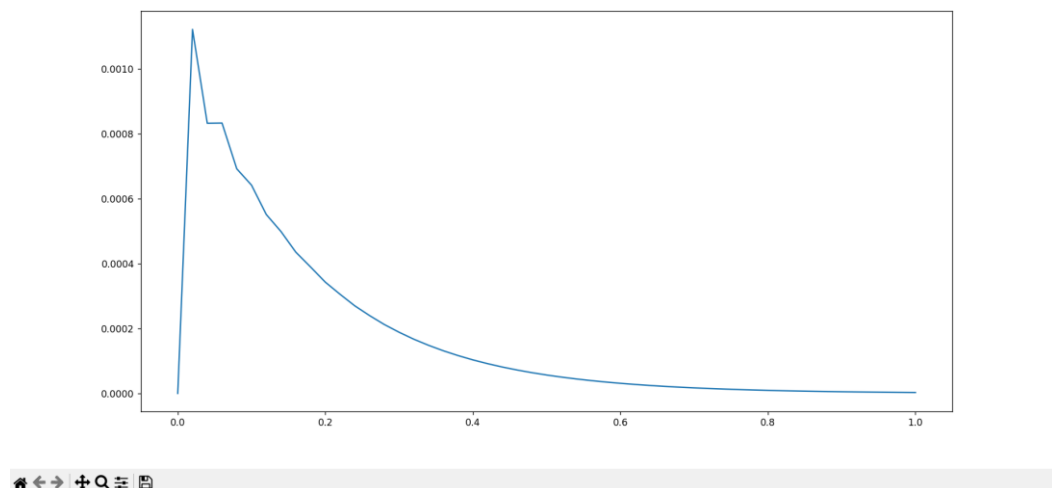
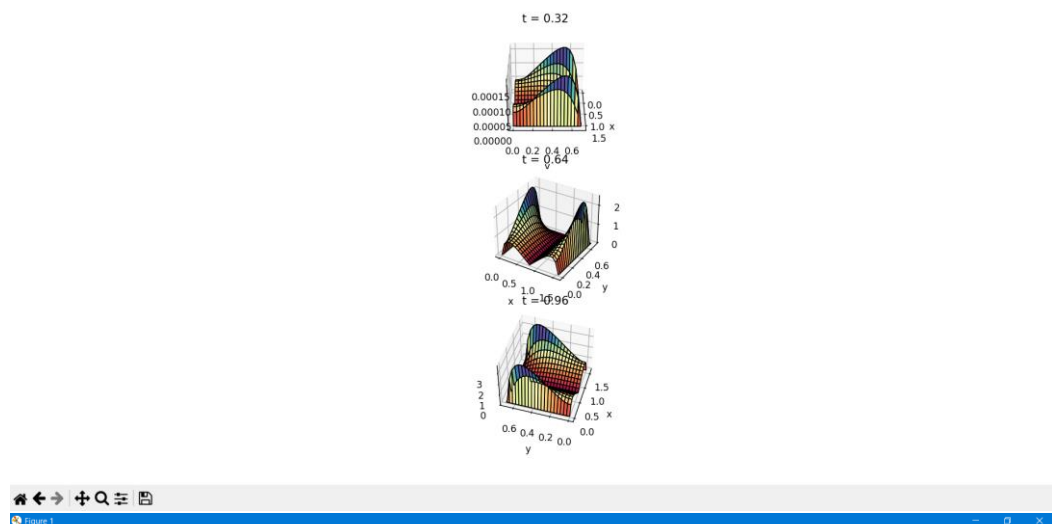
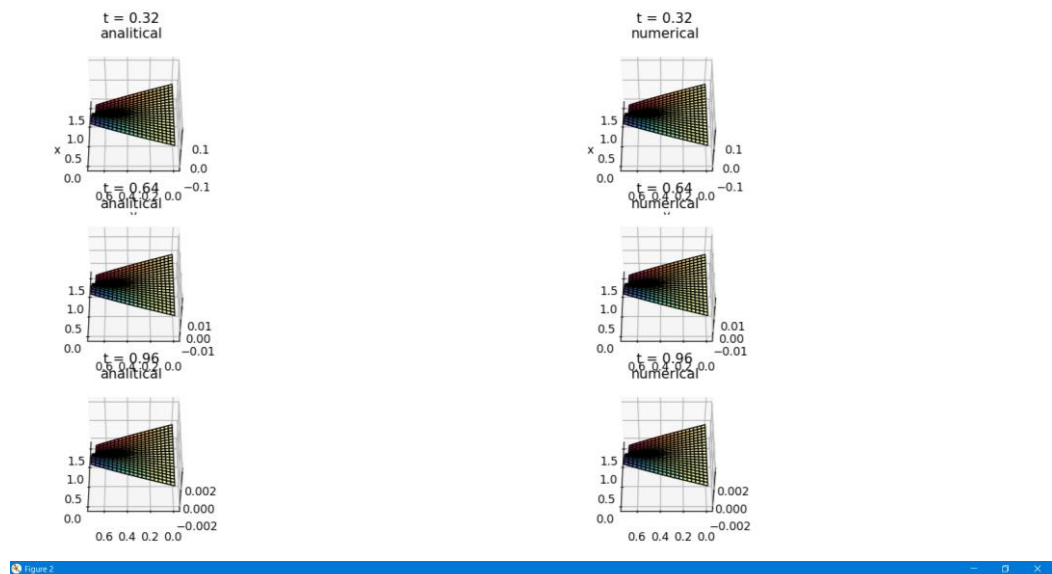
Параметры задачи:  $a = 2$ , разбиение по  $x$  и по  $y = 20$ ,  $T = 1$ , разбиение по времени = 50

Для наглядности приведен график точного решения в системе Desmos 3D:



### 3. Вывод программы

- Метод переменных направлений



- Метод дробных шагов

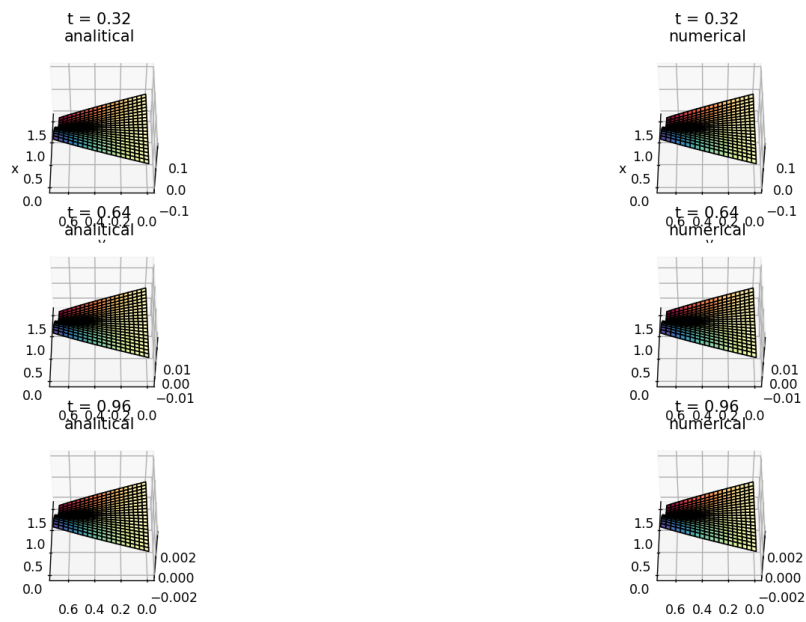


Figure 2

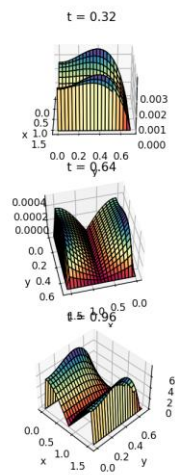


Figure 1

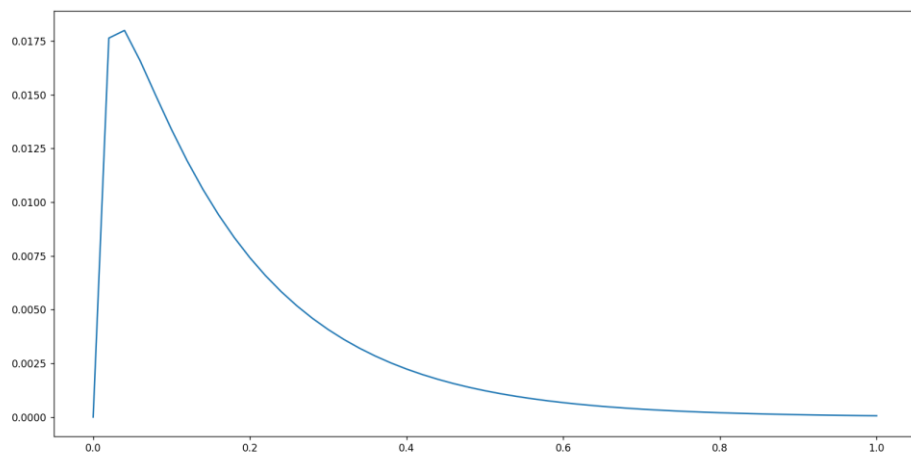


Figure 3

## 4. ЛИСТИНГ

```
5. import numpy as np
6. import matplotlib.pyplot as plt
7. import matplotlib
8.
9. # вариант 5
10.
11. def true_fval(x, y, t, a):
12.     return np.cos(2*x) * np.sinh(y) * np.exp(-3*a* t)
13.
14. def phi1(y, t, a):
15.     return np.sinh(y) * np.exp(-3*a* t)
16.
17. def phi2(y, t, a):
18.     return -np.sinh(y) * np.exp(-3*a* t)
19.
20. def phi3(x, t, a):
21.     return np.cos(2*x) * np.exp(-3*a* t)
22.
23. def phi4(x, t, a):
24.     return np.cos(2*x) * np.exp(-3*a* t)*0.75
25.
26. def psi(x, y):
27.     return np.cos(2*x) * np.sinh(y)
28.
29. def tridiagonal(a, b, c, d):
30.     n = len(d)
31.     x = np.zeros(n)
32.     p = [-c[0] / b[0]]
33.     q = [d[0] / b[0]]
34.     for i in range(1, n):
35.         p.append(-c[i] / (b[i] + a[i] * p[i - 1]))
36.         q.append((d[i] - a[i] * q[i - 1]) / (b[i] + a[i] * p[i - 1]))
37.     x[-1] = q[-1]
38.     for i in reversed(range(n - 1)):
39.         x[i] = p[i] * x[i + 1] + q[i]
40.     return x
41.
42. def method1(a, lbx, ubx, nx, lby, uby, ny, T, K):
43.     hx = (ubx - lbx) / nx
44.     x = np.arange(lbx, ubx + hx, hx)
45.
46.     hy = (uby - lby) / ny
47.     y = np.arange(lby, uby + hy, hy)
48.
49.     tau = T / K
50.     t = np.arange(0, T + tau, tau)
51.
52.     UU = np.zeros((len(x), len(y), len(t)))
```

```

53.     for i in range(len(x)):
54.         for j in range(len(y)):
55.             UU[i, j, 0] = psi(x[i], y[j])
56.
57.     for k in range(1, len(t)):
58.         U1 = np.zeros((len(x), len(y)))
59.         t2 = t[k] - tau / 2
60.         # первый дробный шаг
61.         L = np.zeros((len(x), len(y)))
62.         L = UU[:, :, k - 1]
63.         for j in range(len(y) - 1):
64.             aa = np.zeros(len(x))
65.             bb = np.zeros(len(x))
66.             cc = np.zeros(len(x))
67.             dd = np.zeros(len(x))
68.             bb[0] = hx
69.             bb[-1] = hx
70.             cc[0] = 0
71.             aa[-1] = 0
72.             dd[0] = phi1(y[j], t2, a) * hx
73.             dd[-1] = phi2(y[j], t2, a) * hx
74.             for i in range(1, len(x) - 1):
75.                 aa[i] = a
76.                 bb[i] = -2 * (hx**2) / tau - 2*a
77.                 cc[i] = a
78.                 dd[i] = -2 * (hx**2) * L[i, j] / tau - a*(hx**2) *
(L[i, j+1] - 2*L[i, j] + L[i, j-1]) / (hy**2)
79.             xx = tridiagonal(aa, bb, cc, dd)
80.             for i in range(len(x)):
81.                 U1[i, j] = xx[i]
82.                 U1[i, 0] = (phi3(x[i], t2, a) - U1[i, 1] / hy) / (-1/
hy)
83.                 U1[i, -1] = phi4(x[i], t2, a)
84.         for j in range(len(y)):
85.             U1[0, j] = phi1(y[j], t2, a)
86.             U1[-1, j] = phi2(y[j], t2, a)
87.             # второй дробный шаг
88.         U2 = np.zeros((len(x), len(y)))
89.
90.         for i in range(len(x) - 1):
91.             aa = np.zeros(len(x))
92.             bb = np.zeros(len(x))
93.             cc = np.zeros(len(x))
94.             dd = np.zeros(len(x))
95.             bb[0] = -1
96.             bb[-1] = hy
97.             cc[0] = 1
98.             aa[-1] = 0
99.             dd[0] = phi3(x[i], t[k], a) * hy
100.            dd[-1] = phi4(x[i], t[k], a) * hy
101.            for j in range(1, len(y) - 1):

```

```

102.         aa[j] = a
103.         bb[j] = - 2 * (hy**2) / tau - 2*a
104.         cc[j] = a
105.         dd[j] = -2* (hy**2) * U1[i,j] / tau - a*(hy**2)*(U1[i +
1,j] - 2*U1[i, j] + U1[i - 1, j]) / (hx**2)
106.         xx = tridiagonal(aa, bb, cc, dd)
107.         for j in range(len(y)):
108.             U2[i, j] = xx[j]
109.             U2[0, j] = phi1(y[j], t[k], a)
110.             U2[-1, j] = phi2(y[j], t[k], a)
111.         for i in range(len(x)):
112.             U2[i, 0] = (phi3(x[i], t[k], a) - U2[i, 1] / hy) / (-1/ hy)
113.             U2[i, -1] = phi4(x[i], t[k], a)
114.             # print(U2)
115.         for i in range(len(x)):
116.             for j in range(len(y)):
117.                 UU[i, j, k] = U2[i, j]
118.     return UU
119.
120. def method2(a, lbx, ubx, nx, lby, uby, ny, T, K):
121.     hx = (ubx - lbx) / nx
122.     x = np.arange(lbx, ubx + hx, hx)
123.
124.     hy = (uby - lby) / ny
125.     y = np.arange(lby, uby + hy, hy)
126.
127.     tau = T / K
128.     t = np.arange(0, T + tau, tau)
129.
130.     UU = np.zeros((len(x), len(y), len(t)))
131.     for i in range(len(x)):
132.         for j in range(len(y)):
133.             UU[i, j, 0] = psi(x[i], y[j])
134.
135.     for k in range(1, len(t)):
136.         U1 = np.zeros((len(x), len(y)))
137.         t2 = t[k] - tau / 2
138.         # первый дробный шаг
139.         L = np.zeros((len(x), len(y)))
140.         L = UU[:, :, k - 1]
141.
142.         for j in range(len(y) - 1):
143.             aa = np.zeros(len(x))
144.             bb = np.zeros(len(x))
145.             cc = np.zeros(len(x))
146.             dd = np.zeros(len(x))
147.             bb[0] = hx
148.             bb[-1] = hx
149.             cc[0] = 0
150.             aa[-1] = 0
151.             dd[0] = phi1(y[j], t2, a) * hx

```



```

152.         dd[-1] = phi2(y[j], t2, a) * hx
153.     for i in range(1, len(x) - 1):
154.         aa[i] = a
155.         bb[i] = - (hx ** 2) / tau - 2 * a
156.         cc[i] = a
157.         dd[i] = -(hx ** 2) * L[i, j] / tau
158.     xx = tridiagonal(aa, bb, cc, dd)
159.     for i in range(len(x)):
160.         U1[i, j] = xx[i]
161.         U1[i, 0] = (phi3(x[i], t2, a) - U1[i, 1] / hy) / (-1 /
hy)
162.         U1[i, -1] = phi4(x[i], t2, a)
163.     for j in range(len(y)):
164.         U1[0, j] = phi1(y[j], t2, a)
165.         U1[-1, j] = phi2(y[j], t2, a)
166.         # второй дробный шаг
167.     U2 = np.zeros((len(x), len(y)))
168.
169.     for i in range(len(x) - 1):
170.         aa = np.zeros(len(x))
171.         bb = np.zeros(len(x))
172.         cc = np.zeros(len(x))
173.         dd = np.zeros(len(x))
174.         bb[0] = -1
175.         bb[-1] = hy
176.         cc[0] = 1
177.         aa[-1] = 0
178.         dd[0] = phi3(x[i], t[k], a) * hy
179.         dd[-1] = phi4(x[i], t[k], a) * hy
180.         for j in range(1, len(y) - 1):
181.             aa[j] = a
182.             bb[j] = - (hy**2) / tau - 2 * a
183.             cc[j] = a
184.             dd[j] = -(hy**2) * U1[i, j] / tau
185.         xx = tridiagonal(aa, bb, cc, dd)
186.         for j in range(len(y)):
187.             U2[i, j] = xx[j]
188.             U2[0, j] = phi1(y[j], t[k], a)
189.             U2[-1, j] = phi2(y[j], t[k], a)
190.         for i in range(len(x)):
191.             U2[i, 0] = (phi3(x[i], t[k], a) - U2[i, 1] / hy) / (-1 /
hy)
192.             U2[i, -1] = phi4(x[i], t[k], a)
193.         for i in range(len(x)):
194.             for j in range(len(y)):
195.                 UU[i, j, k] = U2[i, j]
196.     return UU
197.
198.     a = 2

```

```

199.
200. lbx = 0
201. ubx = np.pi/2
202. nx = 20
203. hx = (ubx - lbx) / nx
204.
205. lby = 0
206. uby = np.log(2)
207. ny = 20
208. hy = (uby - lby) / ny
209.
210. T = 1
211. K = 50
212. tau = T / K
213.
214. x = np.arange(lbx, ubx + hx, hx)
215. y = np.arange(lby, uby + hy, hy)
216. t = np.arange(0, T + tau, tau)
217.
218. step = len(t) // 3 - 1
219. yy, xx = np.meshgrid(y, x)
220. z = []
221. z.append(true_fval(xx, yy, 0, a))
222. z.append(true_fval(xx, yy, t[step], a))
223. z.append(true_fval(xx, yy, t[step * 2], a))
224. z.append(true_fval(xx, yy, t[step * 3], a))
225.
226. #u = method1(a, lbx, ubx, nx, lby, uby, ny, T, K)
227. u = method2(a, lbx, ubx, nx, lby, uby, ny, T, K)
228.
229. resz = []
230.
231. for q in range(3):
232.     resz.append([])
233.     for i in range(len(x)):
234.         resz[q].append([])
235.         for j in range(len(y)):
236.             resz[q][i].append(u[i][j][step*(q+1)])
237. resz = np.array(resz)
238.
239. def plt_res_error(xx, yy, t, z, resz, step):
240.
241.     error_t = []
242.
243.     for i in range(len(t)):
244.         zz = true_fval(xx, yy, t[i], a)
245.         zz = np.array(zz)
246.         error_t.append(np.abs(zz - np.array(u[:, :, i])).max(axis =
(0,1)))
247.
248.     figure = plt.figure(figsize = (20, 10))

```

```

249.     plt.plot(t, error_t)
250.
251.     fig1, ax1 = plt.subplots(3, 1, figsize = (20, 20), subplot_kw =
{"projection": "3d"})
252.
253.     ax1[0].set_title("t = " + str(t[step]))
254.     ax1[0].view_init(30, 0)
255.     ax1[0].set(xlabel='x', ylabel='y')
256.     surf = ax1[0].plot_surface(xx, yy, np.abs(z[1] - resz[0]),
257.                                edgecolors = ["black"], linewidth = 1,
258.                                cmap = matplotlib.cm.Spectral, shade =
True, antialiased = True)
259.
260.     ax1[1].set_title("t = " + str(t[step * 2]))
261.     ax1[1].view_init(30, 0)
262.     ax1[1].set(xlabel='x', ylabel='y')
263.     surf = ax1[1].plot_surface(xx, yy, np.abs(z[2] - resz[1]),
264.                                edgecolors = ["black"], linewidth = 1,
265.                                cmap = matplotlib.cm.Spectral, shade =
True, antialiased = True)
266.
267.     ax1[2].set_title("t = " + str(t[step * 3]))
268.     ax1[2].view_init(30, 0)
269.     ax1[2].set(xlabel='x', ylabel='y')
270.     surf = ax1[2].plot_surface(xx, yy, np.abs(z[3] - resz[2]),
271.                                edgecolors = ["black"], linewidth = 1,
272.                                cmap = matplotlib.cm.Spectral, shade =
True, antialiased = True)
273.
274.     fig, ax = plt.subplots(3, 2, figsize = (20, 20), subplot_kw =
{"projection": "3d"})
275.
276.     ax[0][0].set_title("t = " + str(t[step]) + "\n" + "analitical")
277.     ax[0][0].view_init(50, 180)
278.     ax[0][0].set(xlabel='x', ylabel='y')
279.     surf = ax[0][0].plot_surface(xx, yy, z[1],
280.                                edgecolors = ["black"], linewidth = 1,
281.                                cmap = matplotlib.cm.Spectral, shade =
True, antialiased = True)
282.
283.     ax[0][1].set_title("t = " + str(t[step]) + "\n" + "numerical")
284.     ax[0][1].view_init(50, 180)
285.     ax[0][1].set(xlabel='x', ylabel='y')
286.     surf = ax[0][1].plot_surface(xx, yy, resz[0],
287.                                edgecolors = ["black"], linewidth = 1,
288.                                cmap = matplotlib.cm.Spectral, shade =
True, antialiased = True)
289.
290.     ax[1][0].set_title("t = " + str(t[step * 2]) + "\n" + "analitical")
291.     ax[1][0].view_init(50, 180)
292.     ax[0][1].set(xlabel='x', ylabel='y')

```

```

293.         surf = ax[1][0].plot_surface(xx, yy, z[2],
294.                                         edgecolors = ["black"], linewidth = 1,
295.                                         cmap = matplotlib.cm.Spectral, shade =
True, antialiased = True)
296.
297.         ax[1][1].set_title("t = " + str(t[step * 2]) + "\n" + "numerical")
298.         ax[1][1].view_init(50, 180)
299.         ax[0][1].set(xlabel='x', ylabel='y')
300.         surf = ax[1][1].plot_surface(xx, yy, resz[1],
301.                                         edgecolors = ["black"], linewidth = 1,
302.                                         cmap = matplotlib.cm.Spectral, shade =
True, antialiased = True)
303.
304.         ax[2][0].set_title("t = " + str(t[step * 3]) + "\n" + "analitical")
305.         ax[2][0].view_init(50, 180)
306.         ax[0][1].set(xlabel='x', ylabel='y')
307.         surf = ax[2][0].plot_surface(xx, yy, z[3],
308.                                         edgecolors = ["black"], linewidth = 1,
309.                                         cmap = matplotlib.cm.Spectral, shade =
True, antialiased = True)
310.
311.         ax[2][1].set_title("t = " + str(t[step * 3]) + "\n" + "numerical")
312.         ax[2][1].view_init(50, 180)
313.         ax[0][1].set(xlabel='x', ylabel='y')
314.         surf = ax[2][1].plot_surface(xx, yy, resz[2],
315.                                         edgecolors = ["black"], linewidth = 1,
316.                                         cmap = matplotlib.cm.Spectral, shade =
True, antialiased = True)
317.
318.         plt.show()
319.
320.     def plt_error_t(xx, yy, t, u,a):
321.         error_t = []
322.
323.         for i in range(len(t)):
324.             zz = true_fval(xx, yy, t[i],a)
325.             zz = np.array(zz)
326.             error_t.append(np.abs(zz - np.array(u[:, :, i])).max(axis =
(0,1)))
327.
328.         figure = plt.figure(figsize = (20, 10))
329.         plt.plot(t, error_t)
330.         plt.show()
331.
332.     plt_res_error(xx, yy, t, z, resz, step)

```

## 5. Вывод

В ходе выполнения лабораторной работы были изучены методы переменных направлений и дробных шагов решений двумерной начально-краевой задачи для дифференциального уравнения параболического типа. Была применена двухточечная аппроксимация краевого условия первого порядка. Были получены результаты в графическом представлении и подсчитаны погрешности для каждого варианта решения.