Содержание

1 Задание

Начально-краевая задача для уравнения колебаний струны

$$\begin{cases} y_{tt} = ay_{xx} + f,\ x \in (0,L), t \in (0,T), \\ y|_{t=0} = \varphi(x), y_t|_{t=0} = v(x), \\ y|_{x=0} = g_0(t), y|_{x=L} = g_L(t). \end{cases} \tag{1}$$

Здесь $a > 0$ - константа, решение $y = y(t,x)$.

Реализовать разностные схемы:

- с весом $\sigma = 0$
- с весом $\sigma = 1/4$
- с весом $\sigma = 1/2$

Подготовить несколько тестовых примеров. Среди обязательных должны быть представлены

1. плоскости $y(t,x) = ax + bt + c$(в том числе константы),
2. линейно-квадратичные функции $y(t,x) = ax^2 + bt + c$
3. параболоиды $y(t,x) = ax^2 + bt^2 + c$
4. $y(t,x) = e^{\sin^2 x + \cos^2 t}$

2 Теоретическая часть

2.1 Тестовые примеры

$a = 1$, $L = 2$, $T = 10L = 20$

2.1.1 Плоскость

$y(t,x) = 2x + 3t - 5$

$$\begin{cases} y_{tt} = y_{xx},\ x \in (0,L), t \in (0,T), \\ y|_{t=0} = 2x - 5, y_t|_{t=0} = 3, \\ y|_{x=0} = 3t - 5, y|_{x=L} = 3t - 1. \end{cases} \tag{2}$$

2.1.2 Линейно-квадратичная функция

$y(t,x) = 2x^2 + t - 3$

$$\begin{cases} y_{tt} = y_{xx} - 4,\ x \in (0,L), t \in (0,T), \\ y|_{t=0} = 2x^2 - 3, y_t|_{t=0} = 1, \\ y|_{x=0} = t - 3, y|_{x=L} = t + 5. \end{cases} \tag{3}$$

2.1.3 Параболоид

$y(t,x) = 3x^2 - 2t^2 - 1$

$$\begin{cases} y_{tt} = y_{xx} - 10,\ x \in (0,L), t \in (0,T), \\ y|_{t=0} = 3x^2 - 1, y_t|_{t=0} = 0, \\ y|_{x=0} = -2t^2 - 1, y|_{x=L} = -2t^2 + 11. \end{cases} \tag{4}$$

2.1.4 Индивидуальная функция

$y(t,x) = e^{\sin^2 x + \cos^2 t}$

$$\begin{cases} y_{tt} = y_{xx} + \left(e^{\sin^2 x + \cos^2 t}\right)\left(\sin^2 2t - 2\cos 2t - \sin^2 2x - 2\cos 2x\right), \\ x \in (0,L), t \in (0,T), \\ y|_{t=0} = e^{\sin^2 x + 1}, y_t|_{t=0} = 0, \\ y|_{x=0} = e^{\cos^2 t}, y|_{x=L} = e^{\sin^2 2 + \cos^2 t}. \end{cases} \tag{5}$$

2.2 Разностные схемы

2.2.1 Схема с весом $\sigma = 0$

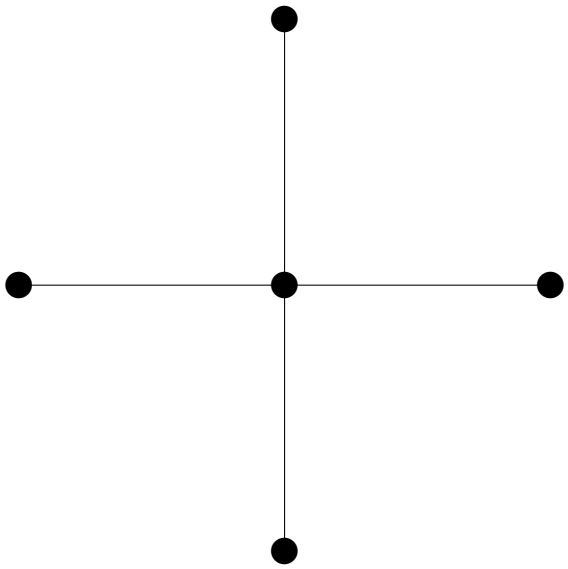
Формула разностного уравнения

$$U_i^{j+1} = \frac{\tau^2}{h^2}\left(U_{i+1}^j - 2U_i^j + U_{i-1}^j\right) + \tau^2 f_i^j + 2U_i^j - U_i^{j-1}$$

Так как разностная схема трехслойная, то возникает проблема старта. Для этого воспользуемся значением производной по t в точке 0. Заполним второй слой по формуле:

$$U_i^1 = \tau V_i + \frac{\tau^2}{2h^2}\left(U_{i+1}^0 - 2U_i^0 + U_{i-1}^0 + \frac{\tau^2}{2} f_i^0 + U_i^0\right)$$

Шаблон



Порядок аппроксимации

$$\psi = C_1\tau^2 + C_2h^2$$

Устойчивость

Схема устойчива по начальным данным, если $\tau < h$

2.2.2 Схемы с весом $\sigma = 1/4, 1/2$

Формула разностного уравнения

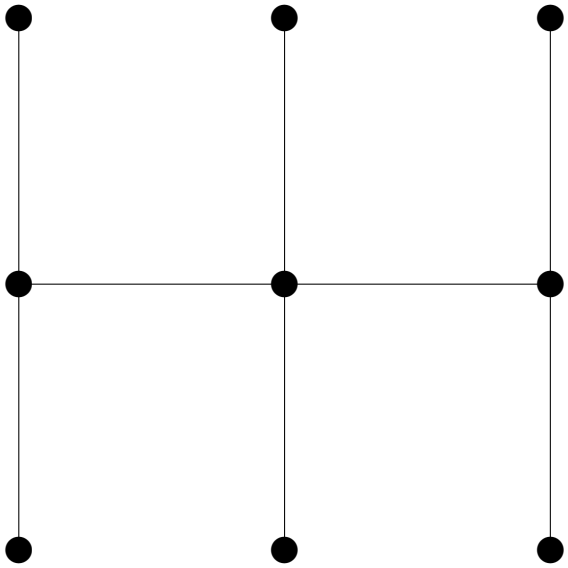
$$\sigma\frac{\tau^2}{h^2}\left(U_{i+1}^{j+1} + U_{i-1}^{j+1}\right) - \left(1 + 2\sigma\frac{\tau^2}{h^2}\right)U_i^{j+1} = \left(2U_i^j - U_i^{j-1}\right) +$$

$$+\tau^2\left(1 - 2\sigma\right)\frac{\left(U_{i+1}^j - 2U_i^j + U_{i-1}^j\right)}{h^2} + \sigma\tau^2\frac{\left(U_{i+1}^{j-1} - 2U_i^{j-1} + U_{i-1}^{j-1}\right)}{h^2} + \tau^2\varphi$$

Так как разностная схема трехслойная, то возникает проблема старта. Для решения этой проблемы воспользуемся значением производной по t в точке 0. Заполняем второй слой по формуле:

$$U_i^1 = \tau V_i + \frac{\tau^2}{2h^2}\left(U_{i+1}^0 - 2U_i^0 + U_{i-1}^0\right) + \frac{\tau^2}{2}f_i^0 + U_i^0$$

Шаблон



Порядок аппроксимации

$$\psi = C_1\tau^2 + C_2h^2$$

Устойчивость[1]

$$\sigma \geq \frac{1}{4} - \frac{h^2}{4\tau^2}$$

3 Практическая часть

3.1 Расчетная схема с весом $\sigma = 0$

3.1.1 Часть 1

Шаги: $h = 0.02$, $\tau = 0.01$, $L = 2$, $T = 20$

Тестовая задача №1

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	2.615E-12	1.168E+284	NAN	NAN	NAN
$\tau_0/2$	5.016E-12	3.375E-12	NAN	NAN	NAN
$\tau_0/4$	3.24E-11	1.125E-11	1.057E-11	NAN	NAN
$\tau_0/8$	1.479E-10	4.194E-11	2.509E-11	1.175E-11	NAN
$\tau_0/16$	1.766E-10	2.58E-10	7.199E-11	5.465E-11	3.42E-11

Тестовая задача №2

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	1.098E-12	3.893E+283	NAN	NAN	NAN
$\tau_0/2$	1.272E-12	1.382E-12	NAN	NAN	NAN
$\tau_0/4$	8.12E-12	5.013E-12	6.679E-12	NAN	NAN
$\tau_0/8$	6.273E-11	1.433E-11	1.431E-11	9.955E-12	NAN
$\tau_0/16$	1.526E-10	9.681E-11	3.226E-11	2.011E-11	2.046E-11

[1]Данное неравенство обеспечивает абсолютную устойчивость, тем не менее некорректно будет его называть условием устойчивости(отсутствует условие на шаги). Это скорее правило выбора веса

Тестовая задача №3

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	3.354E-11	2.508E+282	NAN	NAN	NAN
$\tau_0/2$	6.321E-11	1.063E-10	NAN	NAN	NAN
$\tau_0/4$	4.602E-10	2.998E-10	1.182E-10	NAN	NAN
$\tau_0/8$	1.108E-08	3.768E-10	6.395E-10	5.765E-10	NAN
$\tau_0/16$	7.4E-08	2.15E-08	9.373E-10	7.412E-10	8.388E-10

Тестовая задача №4

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	0.0008271	7.781E+287	NAN	NAN	NAN
$\tau_0/2$	0.0007173	0.0002055	NAN	NAN	NAN
$\tau_0/4$	0.0006938	0.0001776	5.121E-05	NAN	NAN
$\tau_0/8$	0.0006883	0.0001717	4.42E-05	1.278E-05	NAN
$\tau_0/16$	0.0006869	0.0001704	4.273E-05	1.102E-05	3.192E-06

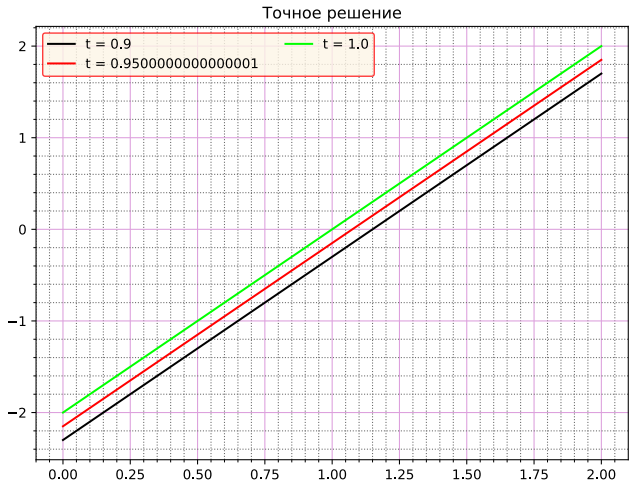
Вывод

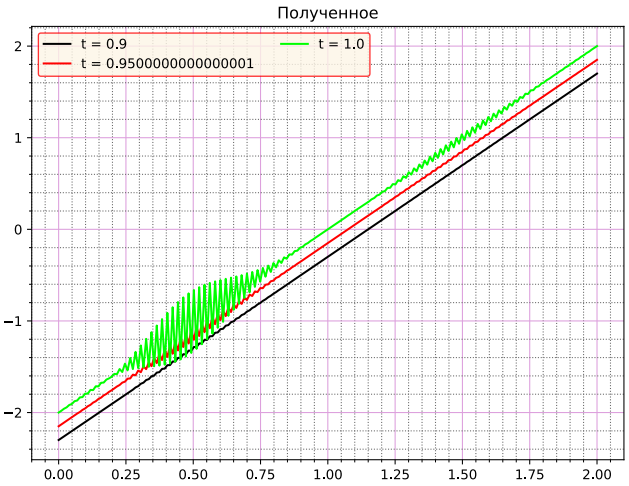
Полученный результат полностью соответствует теоретическим свойствам схемы. При невыполнении условия устойчивости решение во всех случаях расходится. В четвёртой задаче прослеживается зависимость погрешности аппроксимации от шага по h, но погрешность практически не изменяется при изменении шага по τ. Я считаю, что это происходит по причине входных данных. Ведь в точке $t = 0$ первая производная по t равна 0. В первых трёх задачах погрешность аппроксимации минимальна, так как вторые производные тестовых функций представляют собой константы.

3.1.2 Часть 2

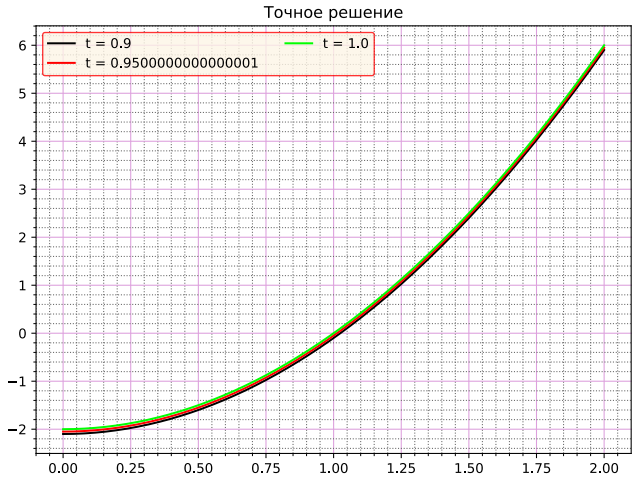
Определим визуально моменты времени, при которых решение становится неустойчивым для каждой из тестовых задач при $\tau = 0.01, h = 0.01$

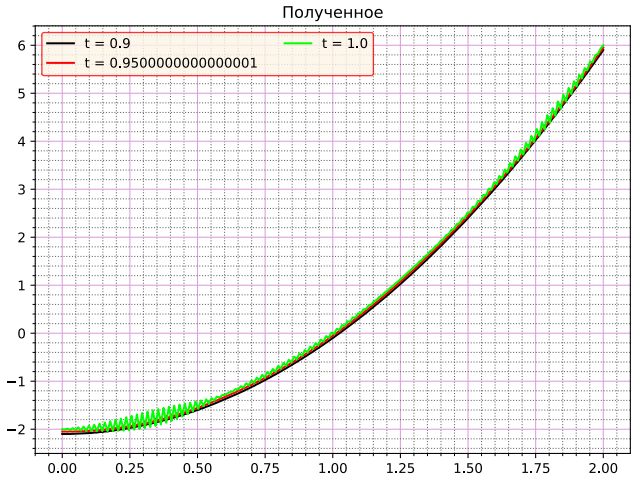
Тестовая задача №1



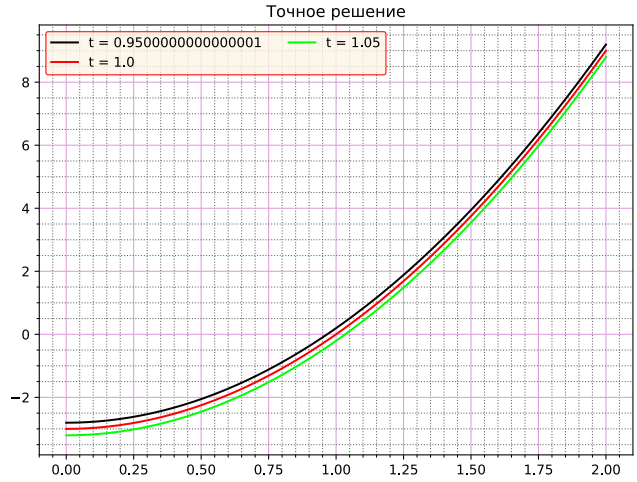


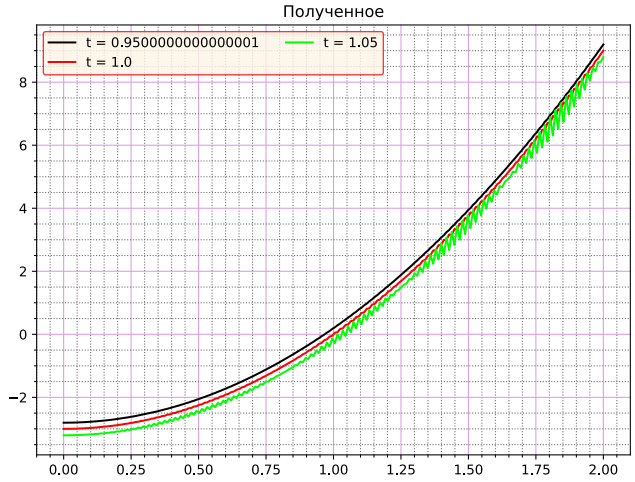
Тестовая задача №2



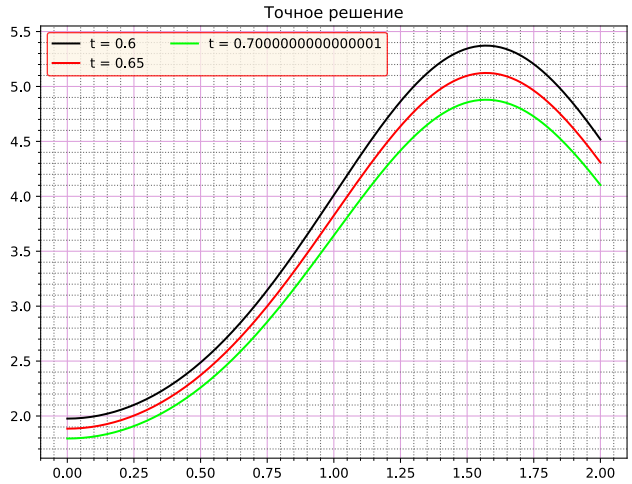


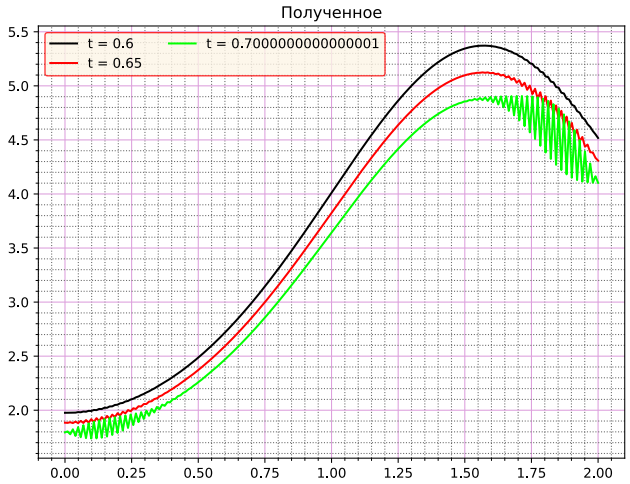
Тестовая задача №3





Тестовая задача №4





3.2 Расчетная схема с весом $\sigma = 1/4$

3.2.1 Часть 1

Шаги: $h = 0.02$, $\tau = 0.01$, $L = 2, T = 20$

Тестовая задача №1

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	5.468E-11	9.919E-12	5.332E-11	1.723E-10	6.473E-10
$\tau_0/2$	1.155E-10	4.869E-11	7.642E-11	2.667E-10	7.708E-10
$\tau_0/4$	1.263E-10	2.544E-10	6.198E-10	2.364E-10	1.399E-09
$\tau_0/8$	1.888E-09	1.551E-10	2.143E-10	1.509E-09	2.041E-09
$\tau_0/16$	8.006E-09	5.772E-09	7.125E-09	6.081E-09	1.153E-08

Тестовая задача №2

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	2.301E-11	7.002E-12	1.776E-11	5.163E-11	2.102E-10
$\tau_0/2$	2.925E-11	2.72E-11	1.71E-11	9.878E-11	2.469E-10
$\tau_0/4$	7.605E-11	1.301E-10	2.537E-10	1.221E-10	4.219E-10
$\tau_0/8$	6.454E-10	3.237E-10	3.112E-10	2.08E-10	1.028E-09
$\tau_0/16$	4.364E-09	1.871E-09	2.331E-09	2.011E-09	3.889E-09

Тестовая задача №3

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	7.493E-10	3.42E-10	8.625E-10	2.278E-09	8.619E-09
$\tau_0/2$	1.718E-09	6.025E-10	1.302E-09	3.6E-09	1.034E-08
$\tau_0/4$	4.355E-09	6.025E-09	1.033E-08	5.877E-09	1.721E-08
$\tau_0/8$	3.861E-08	2.384E-08	2.339E-08	2.645E-08	4.815E-08
$\tau_0/16$	1.443E-07	1.656E-07	1.814E-07	1.703E-07	1.686E-07

Тестовая задача №4

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	0.0008566	0.0003927	0.0003496	0.0003553	0.0003569
$\tau_0/2$	0.000726	0.0002124	9.785E-05	8.737E-05	8.879E-05
$\tau_0/4$	0.0006962	0.0001799	5.289E-05	2.444E-05	2.184E-05
$\tau_0/8$	0.0006889	0.0001723	4.476E-05	1.32E-05	6.108E-06
$\tau_0/16$	0.0006871	0.0001705	4.288E-05	1.116E-05	3.295E-06

3.3 Расчетная схема с весом $\sigma = 1/2$

3.3.1 Часть 1

Шаги: $h = 0.02$, $\tau = 0.01$, $L = 2, T = 20$

Тестовая задача №1

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	3.898E-11	1.953E-11	3.016E-11	2.465E-10	7.992E-10
$\tau_0/2$	1.691E-10	5.009E-11	2.626E-10	6.622E-11	6.236E-10
$\tau_0/4$	7.917E-11	8.053E-10	2.868E-10	6.154E-11	7.373E-10
$\tau_0/8$	2.388E-09	9.444E-10	1.107E-09	2.413E-09	4.708E-09
$\tau_0/16$	1.195E-09	3E-09	5.455E-09	1.514E-08	5.815E-09

Тестовая задача №2

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	1.328E-11	1.211E-11	1.087E-11	7.608E-11	2.639E-10
$\tau_0/2$	4.547E-11	1.328E-11	7.639E-11	1.577E-11	1.97E-10
$\tau_0/4$	3.881E-11	3.122E-10	1.417E-10	6.452E-11	2.033E-10
$\tau_0/8$	1.162E-09	6.727E-10	1.359E-10	1.163E-09	1.919E-09
$\tau_0/16$	5.772E-10	9.332E-10	1.851E-09	5.082E-09	1.964E-09

Тестовая задача №3

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	7.054E-10	2.798E-10	5.584E-10	3.261E-09	1.067E-08
$\tau_0/2$	2.294E-09	1.028E-09	3.537E-09	1.196E-09	8.376E-09
$\tau_0/4$	3.371E-09	1.282E-08	6.313E-09	4.066E-09	9.202E-09
$\tau_0/8$	5.351E-08	3.615E-08	2.369E-08	5.283E-08	7.865E-08
$\tau_0/16$	8.727E-08	1.35E-07	1.043E-07	2.167E-07	1.062E-07

Тестовая задача №4

	h_0	$h_0/2$	$h_0/4$	$h_0/8$	$h_0/16$
$\tau_0/1$	0.0008883	0.0003888	0.000333	0.0003237	0.0003214
$\tau_0/2$	0.0007352	0.0002203	9.7E-05	8.33E-05	8.096E-05
$\tau_0/4$	0.0006986	0.0001821	5.486E-05	2.423E-05	2.083E-05
$\tau_0/8$	0.0006895	0.0001729	4.532E-05	1.369E-05	6.053E-06
$\tau_0/16$	0.0006872	0.0001707	4.302E-05	1.13E-05	3.418E-06

Вывод

В РС с весами равными $\sigma = 1/4$, $\sigma = 1/2$ методы получаются абсолютно устойчивыми, так как, условие устойчивости как такового нет. Вместо этого имеется правило выбора веса. В задачах 1-3 наблюдается накапливание погрешности в силу простой структуры тестируемых функций. В задаче 4 функция более сложная и по полученным данным зависимость погрешности аппроксимации от шагов видна хорошо.

4 Приложение

```
import numpy as np
from matplotlib import pyplot as plt

L = 2
T = L * 10
Nl = 51
Nt = 2001


def y1(x, t):
    return 2 * x + 3 * t - 5


def f1(x, t):
    return 0


def y01(x):
    return 2 * x - 5


def g01(t):
    return 3 * t - 5

```

```
25
26  def gl1(t):
27      return 3 * t - 1
28
29
30  def v1(x):
31      return 3
32
33
34  def y2(x, t):
35      return 2 * x ** 2 + t - 3
36
37
38  def f2(x, t):
39      return -4
40
41
42  def y02(x):
43      return 2 * x ** 2 - 3
44
45
46  def g02(t):
47      return t - 3
48
49
50  def gl2(t):
51      return t + 5
52
53
54  def v2(x):
55      return 1
56
57
58  def y3(x, t):
59      return 3 * x ** 2 - 2 * t ** 2 - 1
60
61
62  def f3(x, t):
63      return -10
64
65
66  def y03(x):
67      return 3 * x ** 2 - 1
```

```
68
69  def g03(t):
70      return - 2 * t ** 2 - 1
71
72
73  def gl3(t):
74      return - 2 * t ** 2 + 11
75
76
77  def v3(x):
78      return 0
79
80
81  def y4(x, t):
82      return np.exp((np.sin(x))**2 + (np.cos(t))**2)
83
84
85  def f4(x, t):
86      return (np.exp((np.sin(x))**2 + (np.cos(t))**2))*\
87             ((np.sin(2*t))**2 - 2*np.cos(2*t) -
88              (np.sin(2*x))**2 - 2*np.cos(2*x))
89
90
91  def y04(x):
92      return np.exp((np.sin(x))**2 + 1)
93
94
95  def g04(t):
96      return np.exp((np.cos(t))**2)
97
98
99  def gl4(t):
100     return np.exp((np.sin(2))**2 + (np.cos(t))**2)
101
102
103 def v4(x):
104     return 0
105
106
107 def Krest(f, Nl, Nt, y0, g0, gl, v):
108     U = np.zeros((Nl, Nt))
109     x = np.linspace(0, L, Nl)
110
```

```
111         t = np.linspace(0, T, Nt)
112         x1 = L / (Nl - 1)
113         t1 = T / (Nt - 1)
114         for i in range(0, Nl):
115             U[i][0] = y0(x[i])
116         for j in range(0, Nt):
117             U[0][j] = g0(t[j])
118         for j in range(0, Nt):
119             U[Nl - 1][j] = gl(t[j])
120         for i in range(1, Nl - 1):
121             U[i][1] = v(x[i]) * t1 + (t1 ** 2 / (2 * x1 ** 2)) \
122                     * \
123                       (U[i + 1][0] - 2 * U[i][0] + U[i - 1][0])\
124                       + (t1 ** 2) / 2 * f(x[i], t[0]) + U[i][0]
125         for j in range(1, Nt - 1):
126             for i in range(1, Nl - 1):
127                 U[i][j + 1] = (t1 ** 2 / x1 ** 2) * \
128                     (U[i + 1][j] - 2 * U[i][j] + U[i - 1][j]) \
129                         + t1 ** 2 * f(x[i], t[j]) + \
130                               2 * U[i][j] - U[i][j - 1]
131         return U
132
133
134     def Weight(f, Nl, Nt, y0, g0, gl, v, sg):
135         U = np.zeros((Nl, Nt))
136         x = np.linspace(0, L, Nl)
137         t = np.linspace(0, T, Nt)
138         x1 = L / (Nl - 1)
139         t1 = T / (Nt - 1)
140         for i in range(0, Nl):
141             U[i][0] = y0(x[i])
142         for j in range(0, Nt):
143             U[0][j] = g0(t[j])
144         for j in range(0, Nt):
145             U[Nl - 1][j] = gl(t[j])
146         for i in range(1, Nl - 1):
147             U[i][1] = v(x[i]) * t1 + (t1 ** 2 / (2 * x1 ** 2)) \
148                       * (U[i + 1][0] - 2 * U[i][0] + U[i - 1][0])\
149                       + (
150                         t1 ** 2) / 2 * f(x[i], t[0]) + U[i][0]
151
152         A = np.zeros((Nl - 2, Nl - 2))
153         for i in range(0, Nl - 2):
```

```
154             for j in range(0, Nl - 2):
155                 if i == j:
156                     A[i][j] = -(1 + 2 * sg * (t1 / x1) ** 2)
157                 if i == j + 1 or i == j - 1:
158                     A[i][j] = sg * (t1 / x1) ** 2
159 
160         for j in range(1, Nt - 1):
161             b = np.zeros(Nl - 2)
162             for i in range(1, Nl - 1):
163                 if i == 1:
164                     b[i - 1] = -((2 * U[i][j] - U[i][j - 1])
165                                  + t1 ** 2 * (1 - 2 * sg) * (
166                                 U[i - 1][j] - 2 * U[i][j]
167                                 + U[i + 1][j]) / (x1 ** 2)
168                                  + sg * t1 ** 2 * (
169                                             U[i - 1][j - 1]
170                                             - 2 * U[i][j - 1]
171                             + U[i + 1][j - 1]) / (x1 ** 2)
172                                  + t1 ** 2 * f(
173                         x[i], t[j])) - sg * (t1 / x1) ** 2\
174                                * U[i - 1][j + 1]
175                 elif i == Nl - 2:
176                     b[i - 1] = -((2 * U[i][j] - U[i][j - 1])
177                                  + t1 ** 2 * (1 - 2 * sg) * (
178                                 U[i - 1][j] - 2 * U[i][j]
179                                 + U[i + 1][j]) / (x1 ** 2)
180                                  + sg * t1 ** 2 * (
181                                             U[i - 1][j - 1]
182                                             - 2 * U[i][j - 1]
183                                             + U[i + 1][j - 1])
184                                  / (x1 ** 2) + t1 ** 2 * f(
185                         x[i], t[j])) - sg * (t1 / x1) ** 2 \
186                                * U[i + 1][j + 1]
187                 else:
188                     b[i - 1] = -((2 * U[i][j] - U[i][j - 1])
189                                  + t1 ** 2 * (1 - 2 * sg) * (
190                                 U[i - 1][j] - 2 * U[i][j]
191                                 + U[i + 1][j]) / (x1 ** 2)
192                                  + sg * t1 ** 2 * (
193                                             U[i - 1][j - 1]
194                                             - 2 * U[i][j - 1]
195                                             + U[i + 1][j - 1])
196                                  / (x1 ** 2) + t1 ** 2 * f(
```

```
                    x[i], t[j]))
        Q = np.linalg.solve(A, b)
        for i in range(1, Nl - 1):
            U[i][j + 1] = Q[i - 1]
    return U


def Value(y, Nl, Nt):
    value = np.zeros((Nl, Nt))
    x = np.linspace(0, L, Nl)
    t = np.linspace(0, T, Nt)
    for j in range(0, Nt):
        for i in range(0, Nl):
            value[i][j] = y(x[i], t[j])
    return value


def Err(y, f, Nl, Nt, y0, g0, gl, v):
    Nl1 = Nl
    Nt1 = Nt
    err = np.zeros((5, 5))
    for i in range(0, 5):
        for j in range(0, 5):
            value = Value(y, Nl1, Nt1)
            err[i][j] = np.max(np.abs(value -
                        Krest(f, Nl1, Nt1, y0, g0, gl, v)))
            print('t = ', T / (Nt1 - 1), '   h = '
                  , L / (Nl1 - 1))
            print('Погрешность = ',"{:1.4G}".format(err[i][j]))
            Nl1 = Nl1 * 2
        Nt1 = Nt1 * 2
        Nl1 = Nl
    return err


def Err1(y, f, Nl, Nt, y0, g0, gl, v, sg):
    Nl1 = Nl
    Nt1 = Nt
    err = np.zeros((5, 5))
    for i in range(0, 5):
        for j in range(0, 5):
            value = Value(y, Nl1, Nt1)
            err[i][j] = np.max(np.abs(value -
```

```
                Weight(f, Nl1, Nt1, y0, g0, gl, v, sg)))
            print('t = ', T / (Nt1 - 1), '   h = ', L /
                  (Nl1 - 1))
            print('Погрешность = ',"{:1.4G}".format(err[i][j]))
            Nl1 = Nl1 * 2
        Nt1 = Nt1 * 2
        Nl1 = Nl
    return err


def Table(name, y, f, Nl, Nt, y0, g0, gl, v):
    R = Err(y, f, Nl, Nt, y0, g0, gl, v)
    with open(name, 'w') as f:
        f.write('& ' + 'h_{0} & ' + 'h_{0}/2 & '
                + 'h_{0}/4 & ' + 'h_{0}/8 &' + 'h_{0}/16 '
                + ' \\\ ' + '\hline' + '\n')
        for i in range(0, 5):
            for j in range(0, 6):
                if j == 5:
                    f.write(' ' + "{:1.4G}".format(R[i][j - 1]))
                    f.write(' \\\ ' + '\hline')
                    f.write('\n')
                elif j == 0:
                    f.write('\\' + 'tau_{0}/{'
                            + str(2 ** i) + '} & ')
                    print(2 ** i)
                else:
                    f.write("{:1.4G}".format(R[i][j - 1]))
                    f.write(' & ')


def Table1(name, y, f, Nl, Nt, y0, g0, gl, v, sg):
    R = Err1(y, f, Nl, Nt, y0, g0, gl, v, sg)
    with open(name, 'w') as f:
        f.write('& ' + 'h_{0} & ' + 'h_{0}/2 & '
                + 'h_{0}/4 & ' + 'h_{0}/8 &' + 'h_{0}/16 '
                + ' \\\ ' + '\hline' + '\n')
        for i in range(0, 5):
            for j in range(0, 6):
                if j == 5:
                    f.write(' ' + "{:1.4G}".format(R[i][j
                                                      - 1]))
                    f.write(' \\\ ' + '\hline')
```

```
283                     f.write('\n')
284                 elif j == 0:
285                     f.write('\\' + 'tau_{0}/{'
286                             + str(2 ** i) + '} & ')
287                     print(2 ** i)
288                 else:
289                     f.write("{:1.4G}".format(R[i][j - 1]))
290                     f.write(' & ')
```