Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ

к лабораторной работе
на тему

Аппроксимация граничных условий второго рода в методе конечных разностей на примере уравнения теплопроводности

Выполнил: студент группы 153501
Тимофеев Кирилл Андреевич

Проверил: Анисимов Владимир Яковлевич

Минск 2023

Цели выполнения задания

Ознакомиться с наиболее часто применяемыми способами аппроксимации граничных условий второго рода (граничных условий (ГУ) Неймана) в методе конечных разностей (на примере ГУ для одномерного нестационарного уравнения теплопроводности).

Краткие теоретические сведения

Задачи, которые будут использоваться для анализа свойств численных решений с ГУ второго рода, формулируются так: в стержне длиной L с *теплоизолированной* боковой поверхностью торец $x = 0$ *поддерживается* при *постоянной* температуре T_0 (ГУ первого рода), а торец $x = 1$ – *теплоизолирован* (ГУ второго рода); температуропроводность материала стержня постоянна и равна a; в начальный момент времени $t = 0$ стержень нагрет до температуры $T_{нач}(x)$ (координата x отсчитывается от левого торца стержня (рис. 2.4)). Найти распределение температуры по стержню в любой момент времени, т. е. найти функцию $T(x, t)$ для $0 < x \le L$ и $t > 0$.

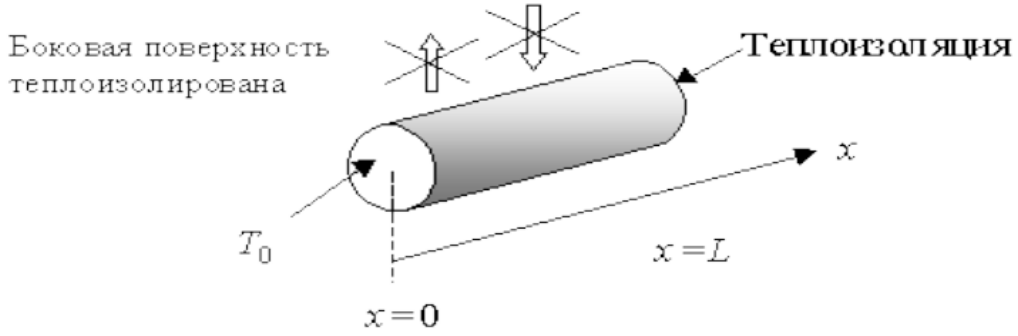


Рис. 2.4.

Стержень круглого сечения нарисован условно – сечение может иметь любую форму, и если боковая поверхность теплоизолирована, то температура любой точки стержня может зависеть только от координаты x и не будет зависеть от координаты поперек стержня).

Искомая функция $T(x,t)$ является решением одномерного уравнения теплопроводности, которое в безразмерных координатах имеет вид:

$$\frac{\partial T}{\partial t} = k\frac{\partial^2 T}{\partial x^2},\ 0 < x < 1,\ 0 < t \le T.$$

Граничные условия:

$$T(0,t)=T_0,\quad \frac{\partial T(1,t)}{\partial x}=0,\ \text{для}\ 0<t\le T$$

(на границе $x = 0$ граничное условие первого рода, а при $x = 1$ – второго).

Начальные условия: $T(x,0) = T_{\text{нач}}(x)$ при $0<x<1$.

Способы реализации ГУ второго рода

Методы конечных разностей, применяемые для численного решения задач с граничными условиями второго (и третьего) рода, не имеют *никаких принципиальных отличий* от методов, применяемых для задач с ГУ первого рода. Для решения поставленной задачи методом конечных разностей необходимо представить граничное условие второго рода в «естественном» для этого метода виде, т. е. с использованием численного решения (величин T_i^n). Иными словами, производную в граничном условии надо заменить ее разностной аппроксимацией, а это можно сделать многими способами. Рассмотрим только два способа реализации ГУ второго рода, которые будут использованы в расчетах. При рассмотрении используем ту же равномерную сетку, что и в лабораторной работе №13.

Первый способ. Приближенно значение производной при $x = 1$ можно записать, используя аппроксимацию производной по x левой разностью:

$$\frac{\partial T}{\partial x}=\frac{T_N^n-T_{N-1}^n}{h}, \qquad (2.43)$$

таблица 2.1.

а поскольку значение этой производной в рассматриваемой задаче равно нулю, то аппроксимация граничного условия выглядит так:

$$T_N^n-T_{N-1}^n=0. \qquad (2.44)$$

Численное решение ДУ с граничным условием второго рода при $x = 1$ происходит почти так же, как и с ГУ первого рода: на каждом шаге по времени с помощью разностной схемы вычисляются все T_i^n при, $1 \le i \le N-1$, а значение T_N^n (на границе) вычисляется по формуле (2.44). Это и есть первый способ реализации граничного условия второго рода. Следует обратить внимание на то, что разностная аппроксимация первой производной левой разностью (формула (2.43)) имеет первый порядок точности по h, т. е. $O(h)$.

Второй способ можно пояснить на примере явной разностной схемы аппроксимации уравнения теплопроводности. Алгоритм явной схемы можно записать так:

$$T_i^{n+1} = T_i^n + \frac{\tau}{h^2}(T_{i+1}^n - 2T_i^n + T_{i-1}^n) \qquad (2.45)$$

Из этого выражения следует, что для вычисления величины T_1^0 требуется какая-то величина T_{N+1}^n, которая *не входит* в расчетную область. Однако ее можно вычислить, используя аппроксимацию производной в граничном условии центральной разностью:

$$\frac{\partial T}{\partial x} = \frac{T_{N+1}^n - T_{N-1}^n}{2h}, \qquad (2.46)$$

а поскольку значение этой производной в рассматриваемой задаче равно нулю, то аппроксимация граничного условия выглядит так:

$$T_N^n = T_{N-1}^n. \qquad (2.47)$$

Способ реализации граничного условия здесь несколько иной: на каждом шаге по времени с помощью разностной схемы вычисляются все T_i^{n+1} при, $\mathbf{1 \le i \le N-1}$, а при вычислении T_1^0 в разностной схеме заменяются T_{N+1}^n на T_{N-1}^n (используется равенство (2.45)).

Обратите внимание на то, что разностная аппроксимация первой производной центральной разностью (формула (2.46)) имеет второй порядок точности по h, т.е. $O(h^2)$. Рассмотренному выше второму способу реализации ГУ второго рода можно дать другую интерпретацию, которая может оказаться более наглядной и полезной в сложных задачах. Эта другая интерпретация связана с введением «фиктивных» узлов (узлов вне зоны расчета). На рис.(2.5) показаны такие узлы (линия узлов, находящихся на

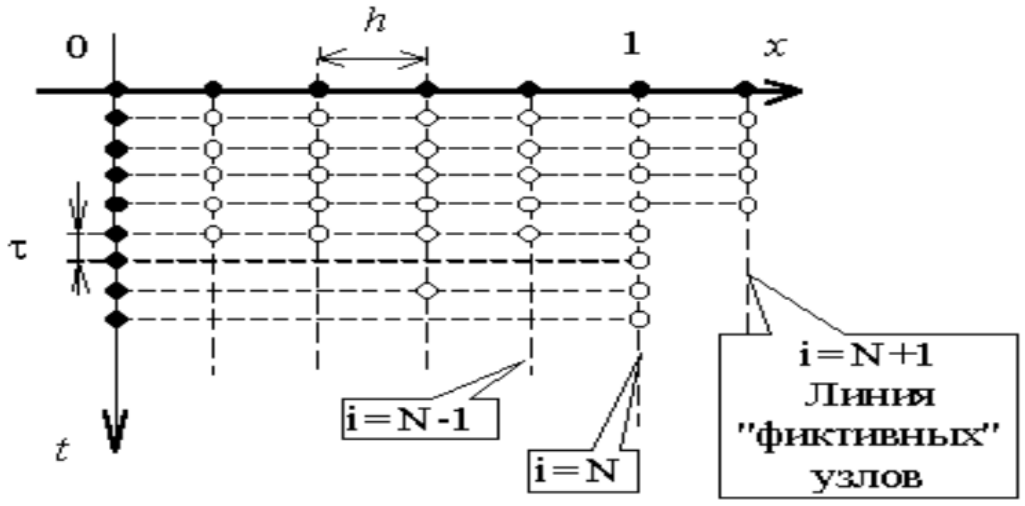


рис.(2.5)

расстоянии h от границы, на которой поставлено ГУ второго рода).

Если температуру в этих узлах задавать равной значениям температуры в соответствующих симметричных относительно границы узлах (согласно равенству (2.45)), то для расчета будет использоваться *одна и та же* разностная схема для всех узлов (включая и узлы при $i=N$). В работе должна быть предусмотрена возможность численного решения уравнения

таблица 2.1.

теплопроводности с помощью неявной и явной разностных схем. Возможность использования различных граничных и начальных условий ограничена задачами, которые позволяют в достаточной мере познакомиться с основными способами реализации ГУ второго рода и их свойствами.

Шаги сетки выбираются также как и в лабораторной работе №13. Расчетная область по времени, реализованная в программе, составляет во всех случаях отрезок [0, 1]. Результаты расчета выводятся в виде таблицы (2.10). После расчета необходимо построить такие же, как в лабораторной работе №13, графики решений.

Программная реализация

```
import numpy as np
import math
from matplotlib import pyplot as plt
from scipy import integrate
from mpl_toolkits.mplot3d import Axes3D

A = -1
B = 1
k = 0.5
phi = lambda x: abs(x)
g1 = lambda t: 1
g2 = lambda t: 1
f = lambda x, t: 0
T = 0.4


def explicit_method_1(h, tau, build_plot=False):
    M = int(T / tau) + 1
    N = int((B - A) / h) + 1

    x_values = np.linspace(A, B, N)
    t_values = np.linspace(0, tau, M)
    matrix = np.zeros((M, N))
    matrix[0] = [phi(x) for x in x_values]
    matrix[:, 0] = [g1(t) for t in t_values]

    for i in range(M - 1):
        for j in range(1, N - 1):
            matrix[i + 1][j] = k * tau / h ** 2 * matrix[i][j - 1]
            matrix[i + 1][j] += (1 - 2 * k * tau / h ** 2) * matrix[i][j]
            matrix[i + 1][j] += k * tau / h ** 2 * matrix[i][j + 1]
            matrix[i + 1][j] += tau * f(x_values[j], t_values[i])

        matrix[i + 1][-1] = matrix[i + 1][-2] + h * g2(t_values[i])

    if build_plot:
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')

        for i in range(len(t_values) - 1, -2, -1000):
            x = x_values
            y = [i * tau] * len(x)
            z = matrix[i]
            ax.plot(x, y, z)

        plt.show()

    return matrix
```

```
def explicit_method_2(h, tau, build_plot=False):
    M = int(T / tau) + 1
    N = int((B - A) / h) + 1

    x_values = np.linspace(A, B, N)
    t_values = np.linspace(0, tau, M)
    matrix = np.zeros((M, N))
    matrix[0] = [phi(x) for x in x_values]
    matrix[:, 0] = [g1(t) for t in t_values]

    for i in range(M - 1):
        for j in range(1, N - 1):
            matrix[i + 1][j] = k * tau / h ** 2 * matrix[i][j - 1]
            matrix[i + 1][j] += (1 - 2 * k * tau / h ** 2) * matrix[i][j]
            matrix[i + 1][j] += k * tau / h ** 2 * matrix[i][j + 1]
            matrix[i + 1][j] += tau * f(x_values[j], t_values[i])

        matrix[i + 1][-1] = k * tau / h ** 2 * matrix[i][-2]
        matrix[i + 1][-1] += (1 - 2 * k * tau / h ** 2) * matrix[i][-1]
        matrix[i + 1][-1] += k * tau / h ** 2 * (2 * h * g2(t_values[i]) +
matrix[i + 1][-2])
        matrix[i + 1][-1] += tau * f(x_values[j], t_values[i])

    if build_plot:
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')

        for i in range(len(t_values) - 1, -2, -1000):
            x = x_values
            y = [i * tau] * len(x)
            z = matrix[i]
            ax.plot(x, y, z)

        plt.show()

    return matrix


def implicit_method_1(h, tau, build_plot=False):
    M = int(T / tau) + 1
    N = int((B - A) / h) + 1

    x_values = np.linspace(A, B, N)
    t_values = np.linspace(0, tau, M)
    result = np.zeros((M, N))
    result[0] = [phi(x) for x in x_values]
    result[:, 0] = [g1(t) for t in t_values]

    matrix = np.zeros((N, N))

    for j in range(1, N - 1):
        matrix[j][j - 1] = - k * tau / h ** 2
        matrix[j][j] = 1 + 2 * k * tau / h ** 2
        matrix[j][j + 1] = - k * tau / h ** 2

    matrix[0][0] = 1
    matrix[-1][-1] = 1
    matrix[-1][-2] = - 1
```

```
    for i in range(1, M):
        b = np.array([tau * f(x, t_values[i] + tau) for x in x_values]) +
result[i - 1]
        b[0] = g1(t_values[i])
        b[-1] = h * g2(t_values[i])
        result[i] = np.linalg.solve(matrix, b)

    if build_plot:
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')

        for i in range(len(t_values) - 1, -2, -1000):
            x = x_values
            y = [i * tau] * len(x)
            z = result[i]
            ax.plot(x, y, z)

        plt.show()

    return result


def implicit_method_2(h, tau, build_plot=False):
    M = int(T / tau) + 1
    N = int((B - A) / h) + 1

    x_values = np.linspace(A, B, N)
    t_values = np.linspace(0, tau, M)
    result = np.zeros((M, N))
    result[0] = [phi(x) for x in x_values]
    result[:, 0] = [g1(t) for t in t_values]

    matrix = np.zeros((N + 1, N + 1))

    for j in range(1, N):
        matrix[j][j - 1] = - k * tau / h ** 2
        matrix[j][j] = 1 + 2 * k * tau / h ** 2
        matrix[j][j + 1] = - k * tau / h ** 2

    matrix[0][0] = 1
    matrix[-1][-1] = 1
    matrix[-1][-3] = -1

    for i in range(1, M):
        b = np.zeros(N + 1)
        b[1:-1] = np.array([tau * f(x, t_values[i] + tau) for x in
x_values[1:]]) + result[i - 1][1:]
        b[0] = g1(t_values[i])
        b[-1] = 2 * h * g2(t_values[i])
        result[i] = np.linalg.solve(matrix, b)[:-1]

    if build_plot:
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')

        for i in range(len(t_values) - 1, -2, -1000):
            x = x_values
            y = [i * tau] * len(x)
            z = result[i]
            ax.plot(x, y, z)
```

```
        plt.show()

    return result

def task():
    h = (B - A) / 500
    tau = 0.5 * (h ** 2) / k

    # explicit_method_1(h, tau, build_plot=True)
    # explicit_method_2(h, tau, build_plot=True)
    # implicit_method_1(h, tau, build_plot=True)
    # implicit_method_2(h, tau, build_plot=True)


from approximation import task
from statistics import stats

if __name__ == '__main__':
    task()
    stats()


from prettytable import PrettyTable
import matplotlib.pyplot as plt
import numpy as np

from approximation import explicit_method_1, explicit_method_2,
implicit_method_1, implicit_method_2, A, B


variants_1 = [
    # N, tau, t1, t2
    [10, 0.001, 0.25, 0.3],
    [10, 0.0005, 0.25, 0.3],
    [10, 0.00025, 0.25, 0.3],
    [10, 0.000125, 0.25, 0.3],
    [10, 0.0000625, 0.25, 0.3]
]

variants_2 = [
    # N, t1, t2
    [10, 0.25, 0.3],
    [20, 0.25, 0.3],
    [30, 0.25, 0.3],
    [40, 0.25, 0.3],
    [50, 0.25, 0.3]
]

methods = [
    [explicit_method_1, 'Explicit method #1'],
    [explicit_method_2, 'Explicit method #2'],
    [implicit_method_1, 'Implicit method #1'],
    [implicit_method_2, 'Implicit method #2']
]

table_1 = PrettyTable()
table_2 = PrettyTable()
table_3 = PrettyTable()
table_4 = PrettyTable()
table_1.field_names = ['N', 'tau', 's(t=t1)', 's(t=t2)', 'max(t=t1)',
'max(t=t2)']
```

```
table_2.field_names = ['N', 'tau', 's(t=t1)', 's(t=t2)', 'max(t=t1)',
'max(t=t2)']
table_3.field_names = ['N', 'tau', 's(t=t1)', 's(t=t2)', 'max(t=t1)',
'max(t=t2)']
table_4.field_names = ['N', 'tau', 's(t=t1)', 's(t=t2)', 'max(t=t1)',
'max(t=t2)']
errors_t1 = []
errors_t2 = []

def stats():
    for method, name in methods:
        print(name)

print('_____________________________________________________________________
__')
        for N, tau, t1, t2 in variants_1:
            h = (B - A) / N
            solution_1 = method(h, tau)
            solution_2 = method(h, tau / 2)
            err_t1 = (solution_1[int(t1 / tau)] - solution_2[2 * int(t1 /
tau)])
            err_t2 = (solution_1[int(t2 / tau)] - solution_2[2 * int(t2 /
tau)])
            errors_t1.append((tau, np.linalg.norm(err_t1)))
            errors_t2.append((tau, np.linalg.norm(err_t2)))

            table_1.add_row([N, tau, err_t1.std(), err_t2.std(),
max(abs(err_t1)), max(abs(err_t2))])

        print('h is fixed')
        print(table_1)
        table_1.clear_rows()

        plt.xscale('log')
        plt.xlabel('tau', fontsize=18)
        plt.ylabel('error', fontsize=18)
        plt.plot([tau for tau, err in errors_t1], [tau for tau, err in
errors_t1], label='t = t1')
        plt.legend()
        plt.show()
        errors_t1.clear()

        plt.xscale('log')
        plt.xlabel('tau', fontsize=18)
        plt.ylabel('error', fontsize=18)
        plt.plot([tau for tau, err in errors_t2], [tau for tau, err in
errors_t2], label='t = t2')
        plt.legend()
        plt.show()
        errors_t2.clear()

        for N, t1, t2 in variants_2:
            h = (B - A) / N
            tau = 0.0001
            solution_1 = method(h, tau)
            solution_2 = method(h / 2, tau)
            err_t1 = (solution_1[int(t1 / tau)] - solution_2[int(t1 /
tau)][::2])
            err_t2 = (solution_1[int(t2 / tau)] - solution_2[int(t2 /
tau)][::2])
            errors_t1.append((h, np.linalg.norm(err_t1)))
```

```
            errors_t2.append((h, np.linalg.norm(err_t2)))

            table_2.add_row([N, tau, err_t1.std(), err_t2.std(),
max(abs(err_t1)), max(abs(err_t2))])

        print('tau is fixed')
        print(table_2)
        table_2.clear_rows()

        plt.xscale('log')
        plt.xlabel('h', fontsize=18)
        plt.ylabel('error', fontsize=18)
        plt.plot([h for h, err in errors_t1], [h for h, err in errors_t1],
label='t = t1')
        plt.legend()
        plt.show()
        errors_t1.clear()

        plt.xscale('log')
        plt.xlabel('h', fontsize=18)
        plt.ylabel('error', fontsize=18)
        plt.plot([h for h, err in errors_t2], [h for h, err in errors_t2],
label='t = t2')
        plt.legend()
        plt.show()
        errors_t2.clear()

        print('tau is h ^ 2 / 6 (relative to tau)')

        for N, t1, t2 in variants_2:
            h = (B - A) / N
            tau = (h ** 2) / 6
            solution_1 = method(h, tau)
            solution_2 = method(h, tau / 2)
            err_t1 = (solution_1[int(t1 / tau)] - solution_2[2 * int(t1 /
tau)])
            err_t2 = (solution_1[int(t2 / tau)] - solution_2[2 * int(t2 /
tau)])

            table_3.add_row([N, tau, err_t1.std(), err_t2.std(),
max(abs(err_t1)), max(abs(err_t2))])

        print(table_3)
        table_3.clear_rows()

        print('tau is h ^ 2 / 6 (relative to h)')

        for N, t1, t2 in variants_2:
            h = (B - A) / N
            tau = (h ** 2) / 6
            solution_1 = method(h, tau)
            solution_2 = method(h / 2, tau)
            err_t1 = (solution_1[int(t1 / tau)] - solution_2[int(t1 /
tau)][::2])
            err_t2 = (solution_1[int(t2 / tau)] - solution_2[int(t2 /
tau)][::2])

            table_4.add_row([N, tau, err_t1.std(), err_t2.std(),
max(abs(err_t1)), max(abs(err_t2))])

        print(table_4)
```

```
table_4.clear_rows()
```

Полученные результаты

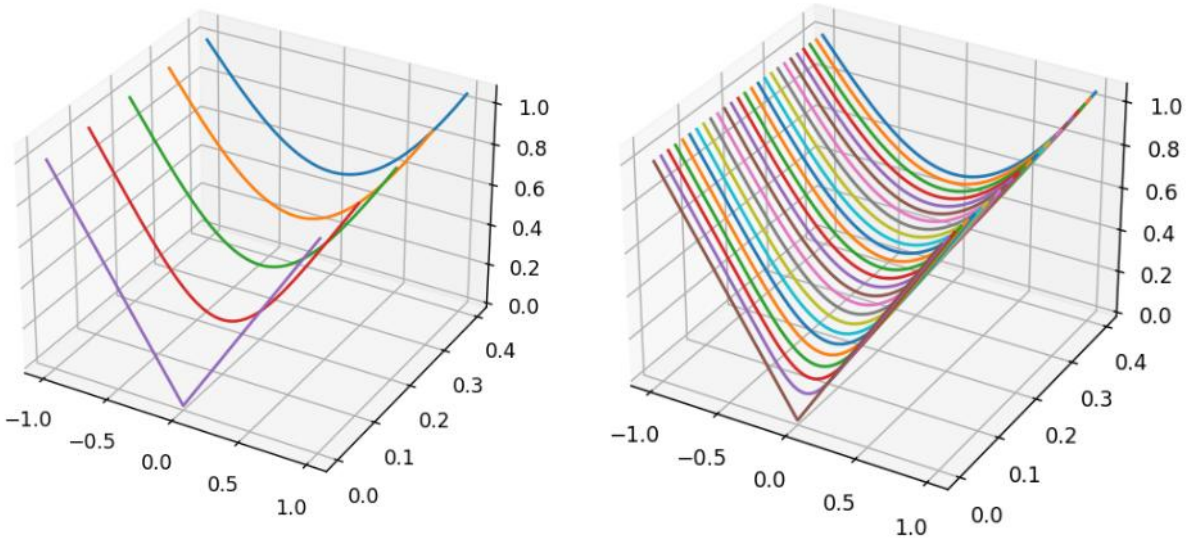
ЗАДАЧА1. Найти приближенное решение начально-краевой задачи для уравнения теплопроводности:

$$\begin{cases} \dfrac{\partial u}{\partial t} = k\dfrac{\partial^2 u}{\partial x^2} + f(x,t),\ a < x < b,\ 0 < t \le T, \\ u(a,t) = g_1(t),\ \dfrac{\partial u}{\partial t}(b,t) = g_2(t),\ 0 < t \le T, \\ u(x,0) = \varphi(x),\ a \le x \le b, \end{cases}$$

используя явную и неявную разностные схемы. Исходные данные указаны в табл. 2.9. Изобразить графики зависимости приближенного решения от x при $t = 0,\ 2\tau, 4\tau, \ldots T.$

№	a	b	k	T	$\varphi(x)$	$g_1(t)$	$g_2(t)$	$f(x,t)$
2	-1	1	0.5	0.4	$\lvert x\rvert$	1	1	0

1. Явная схема (аппроксимация производной левой разностью)



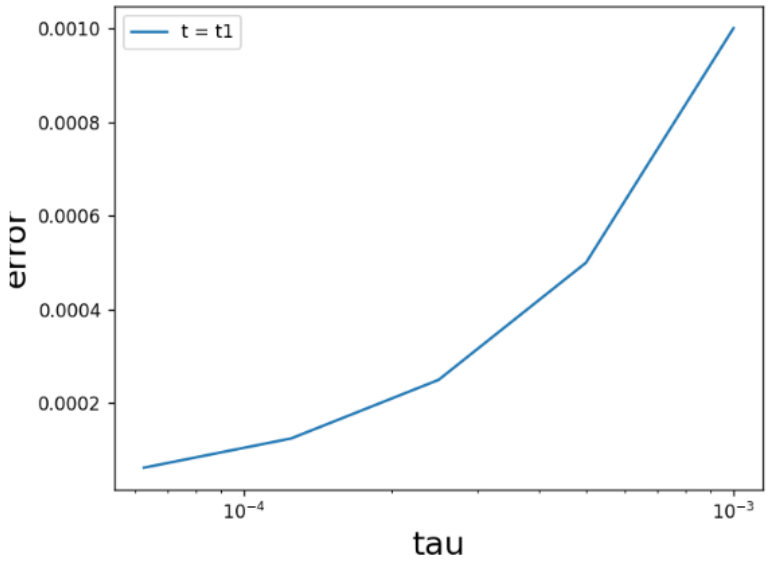
Выше представлены графики численного решения: в первом случае взяли шаг $h = \frac{b-a}{200}$, во втором – $h = \frac{b-a}{500}$.

Далее будут представлены таблицы при произвольном взятии $t_1 = 0.25,\ t_2 = 0.3$ с нахождением обеих характеристик ошибки численного решения: по среднеквадратичному отклонению и по максимуму модуля разности.

Таблица результатов для фиксированного h (N):

N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.001	6.083951305108566e-05	5.579226650329204e-05	0.000105520845257 61174	9.381030058186468e-05
10	0.0005	3.038586604090092 7e-05	2.787284158119085e-05	5.275691948863903e-05	4.690174271787528e-05
10	0.00025	1.518447421113331 9e-05	1.393059880920059 6e-05	2.637760095725028e-05	2.345002668863172e-05
10	0.000125	7.590124142342779e-06	6.963844047525788 4e-06	1.318858786719667e-05	1.172480303818 2241e-05
10	6.25e-05	3.794534076873064e-06	3.481558230682302e-06	6.594240875068547e-06	5.862348839036358e-06

Графики зависимости ошибки численного решения от τ в точках t_1 и t_2 соответственно:



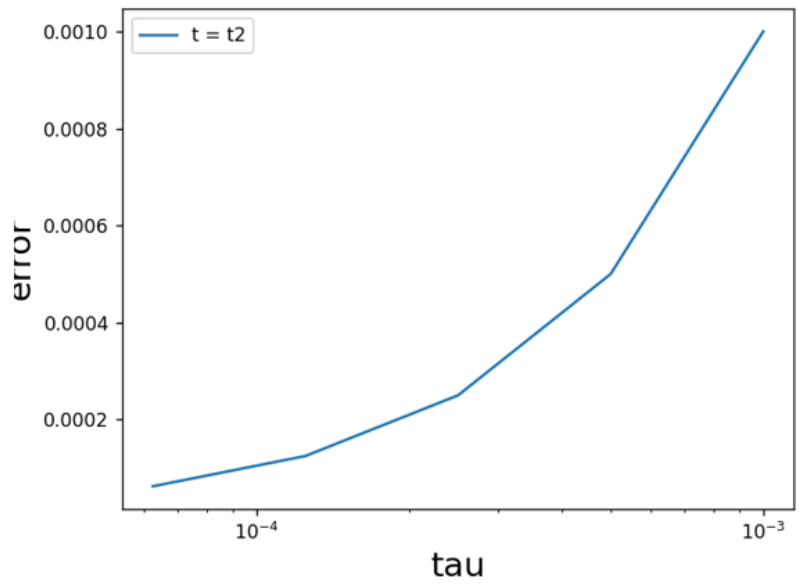
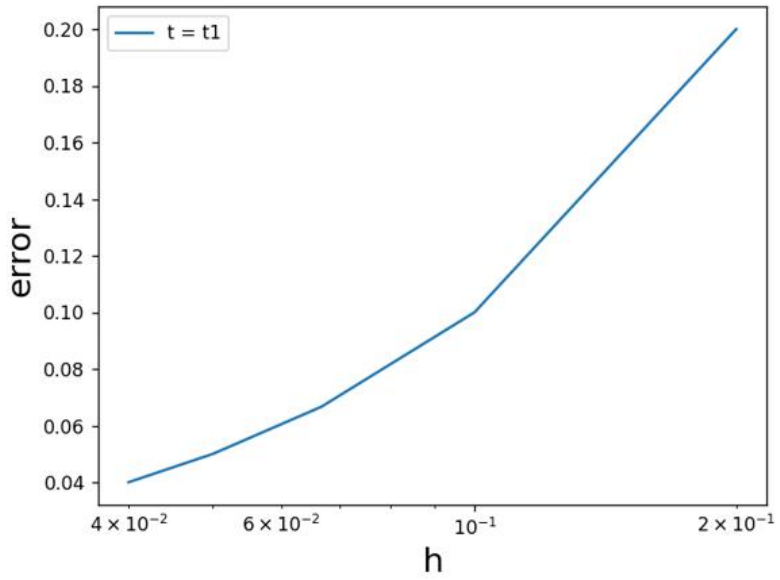


Таблица результатов для фиксированного τ:

N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.0001	0.004236065312913646	0.004862588812475398	0.008979433438118356	0.011674784784903247
20	0.0001	0.001199587972141 4236	0.001497207137499 9365	0.00330768609842 06924	0.00455339733510 5069
30	0.0001	0.000618034618214664	0.000811827909306 6644	0.00196313037658 7907	0.00276735429385 1776
40	0.0001	0.000399222789756966	0.000542576658474 2156	0.00138494632247 19916	0.00197766314869 31873
50	0.0001	0.000289670488603 71707	0.000403269142842 8413	0.00106690396937 62539	0.00153589749458 80455

Графики зависимости ошибки численного решения от h в точках t_1 и t_2 соответственно:



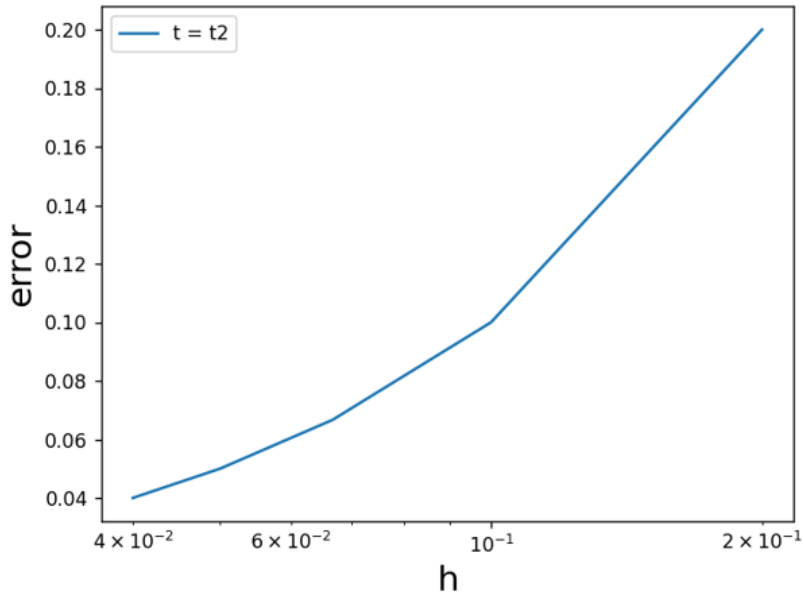


Таблица результатов для $\tau = \frac{h^2}{6}$ относительно h:

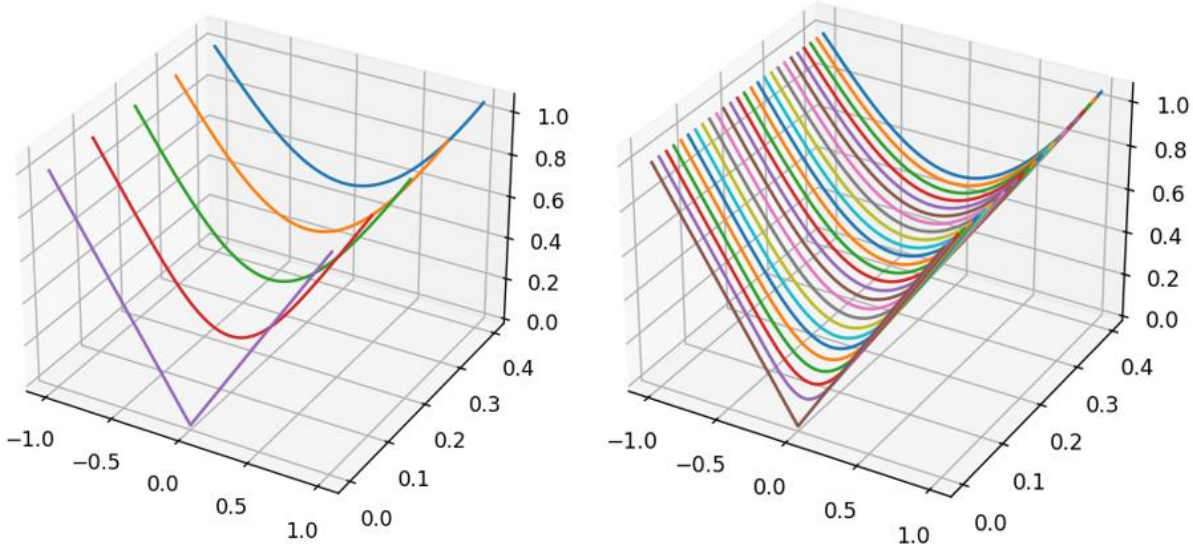
N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.0066666666666668	0.0004130985037660129	0.0003801786303818535355	0.0007101227510256658	0.0006352210211865161
20	0.0016666666666667	9.805396822435236e-05	9.099646777473177e-05	0.0001687083670265932	0.0001525604355864795 8
30	0.0007407407407407407	4.277486821661204e-05	3.979788207350345e-05	7.424199967115408e-05	6.73737453228962e-05
40	0.0004166666666666675	2.3862979368849412e-05	2.224036297833858e-05	4.167055612386283 4e-05	3.790002216552679e-05
50	0.0002666666666666667	1.519116093515303e-05	1.4163067547785727e-05	2.662798490299378 4e-05	2.423586502298125e-05

Таблица результатов для $\tau = \frac{h^2}{6}$ относительно τ:

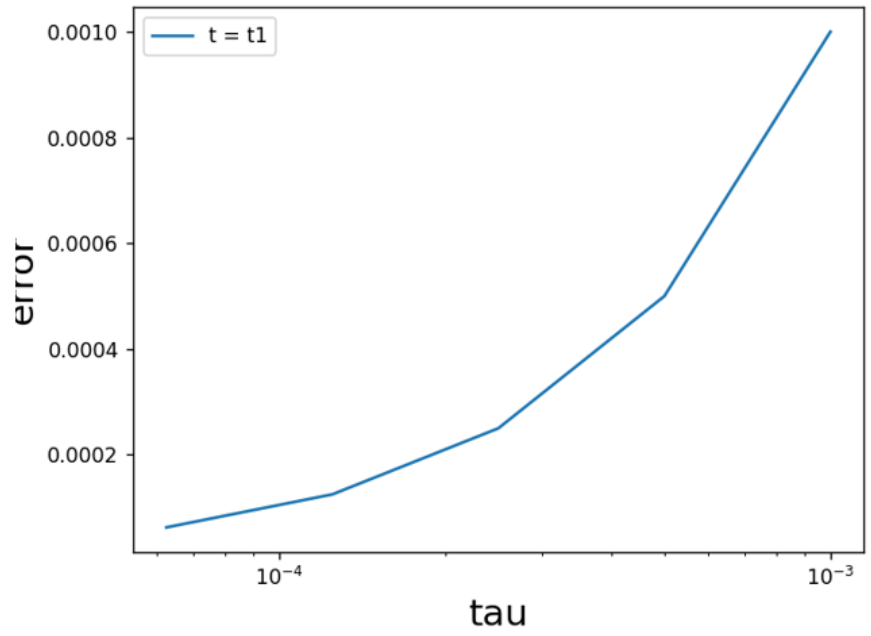
N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.0066666666666668	0.004188889875095468	0.004780857975347526	0.00881912147951125	0.011374175990844115
20	0.0016666666666667	0.001188120873403 3297	0.001486576907978 9994	0.00326332766525 1392	0.00451577770472 9551
30	0.0007407407407407407	0.000615985148416 8361	0.000811917814160 2111	0.00195594257888 97578	0.00276900272061 5754
40	0.0004166666666666675	0.000397829380985 44717	0.000541475115320 0336	0.00137951616900 06719	0.00197378941281 40088
50	0.0002666666666666667	0.000289265297062 5794	0.000403444265563 7222	0.00106541245423 33054	0.00153680192825 71144

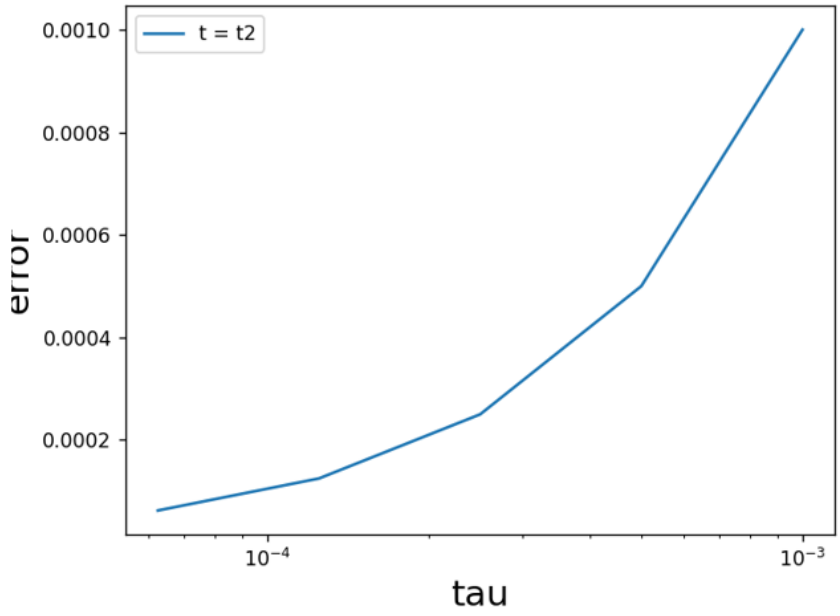
Для последующих способов я не буду писать данные пояснения, так как они будут идентичны.

2. **Явная схема (аппроксимация производной центральной разностью)**

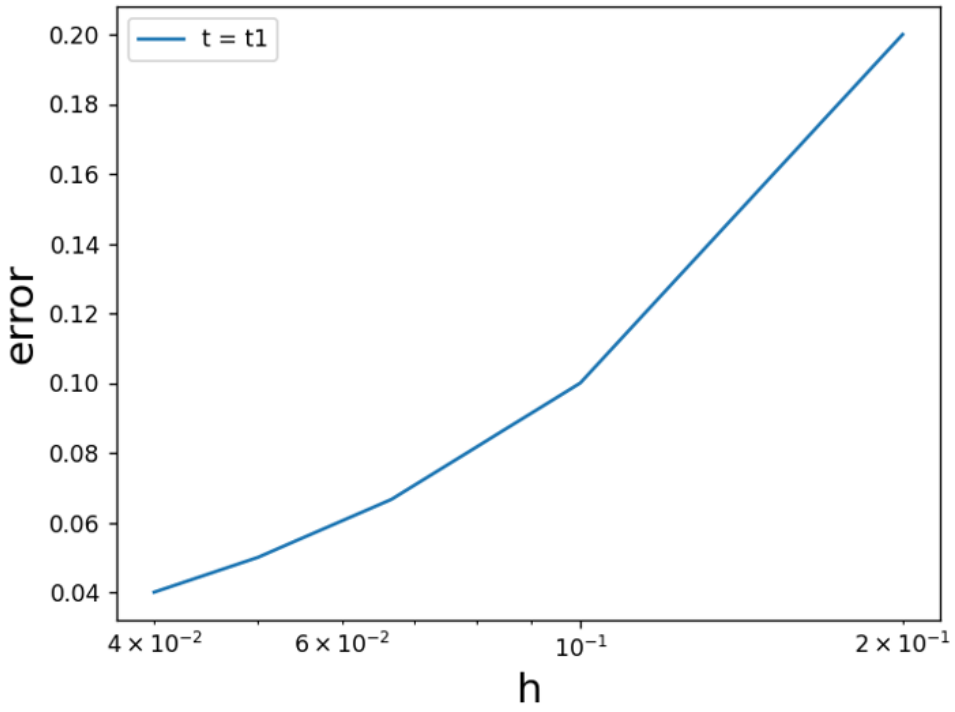


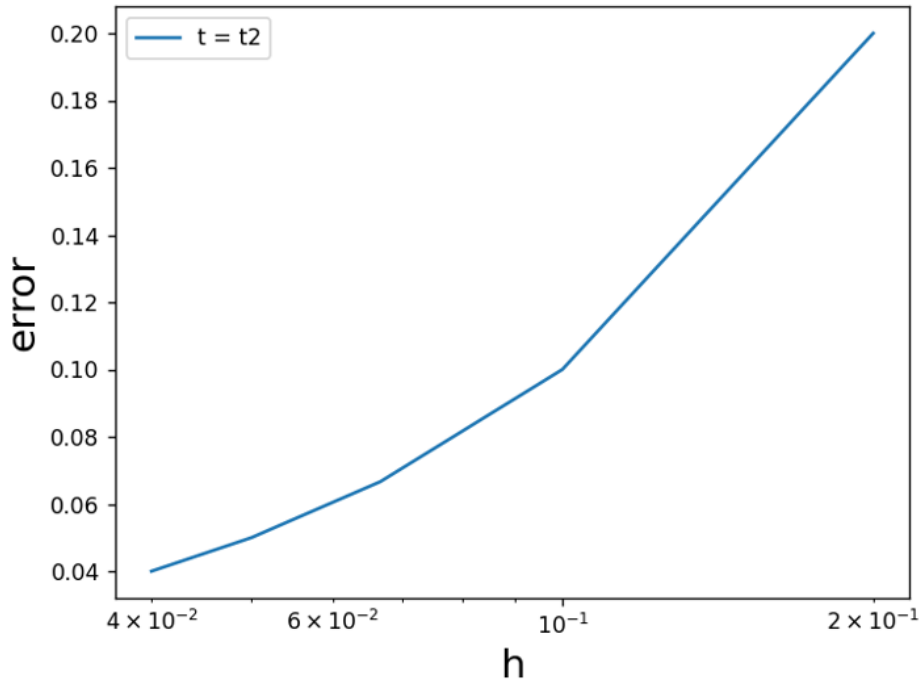
N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.001	4.723245373328425e-05	3.889864988051209 5e-05	0.000107247125214 42692	9.714134818894 493e-05
10	0.0005	2.359392547529224e-05	1.944151449908392 3e-05	5.362866942265620 6e-05	4.857750775411 063e-05
10	0.00025	1.179139986622064 8e-05	9.718803172684687e-06	2.681562436368523e-05	2.429046300644 888e-05
10	0.000125	5.894310769373536e-06	4.858912881981175e-06	1.340813617112246 4e-05	1.214565883123 0755e-05
10	6.25e-05	2.946808313883246 4e-06	2.429334253459118e-06	6.704149119407532e-06	6.072936045375 954e-06





N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.0001	0.0023553099544022264	0.0020877108276168076	0.006236858220045993	0.005651469813093834 6
20	0.0001	0.0005625461032685054	0.0004938801825253 94	0.001510229826957 8096	0.001376758048161 5638
30	0.0001	0.0002445056940484604	0.0002121000299683911	0.000667646136017 8854	0.000609303950342 6989
40	0.0001	0.0001339587284185 7618	0.0001140567353921 5733	0.000374892557554074	0.000342355137801 9529
50	0.0001	8.272192365673557 e-05	6.878731656990887 e-05	0.000239765389848 10114	0.000219110492467 36702

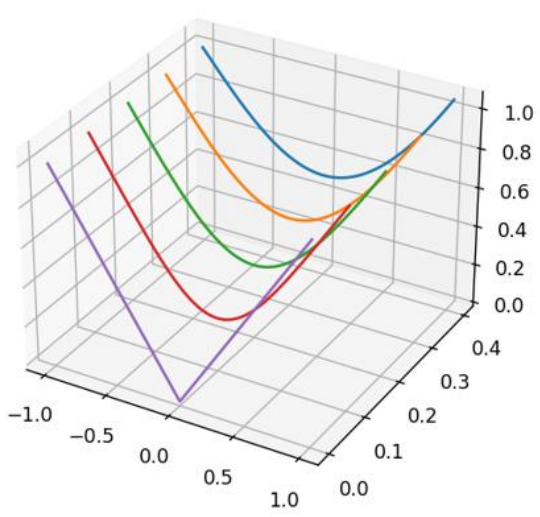

t = t1
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
error
4×10^{-2}
6×10^{-2}
10^{-1}
2×10^{-1}
h

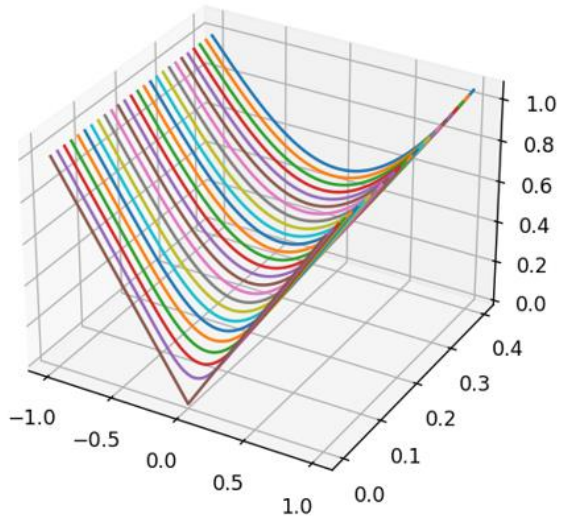

t = t2
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
error
4×10^{-2}
6×10^{-2}
10^{-1}
2×10^{-1}
h

N	τ	$s(t=t_{n1})$	$s(t=t_{n2})$	$max(t=t_{n1})$	$max(t=t_{n2})$
10	0.0066666666666668	0.0003223678218278181	0.0002676821241652717	0.0007197449361439623	0.0006542552219780395
20	0.0016666666666667	7.290729291957187e-05	6.112253843373224e-05	0.0001698286984131991 4	0.0001635813542595343 5
30	0.0007407407407407407	3.167979235727355e-05	3.188795504795445e-05	7.465668029882355e-05	0.0001354779752587021
40	0.0004166666666666675	1.878520148434441e-05	2.287392064601840 8e-05	6.342135109216329e-05	0.0001107068948453 8043
50	0.0002666666666666667	1.333756488513028e-05	1.860121858697299 5e-05	5.57208755758154e-05	9.35756276965182 8e-05

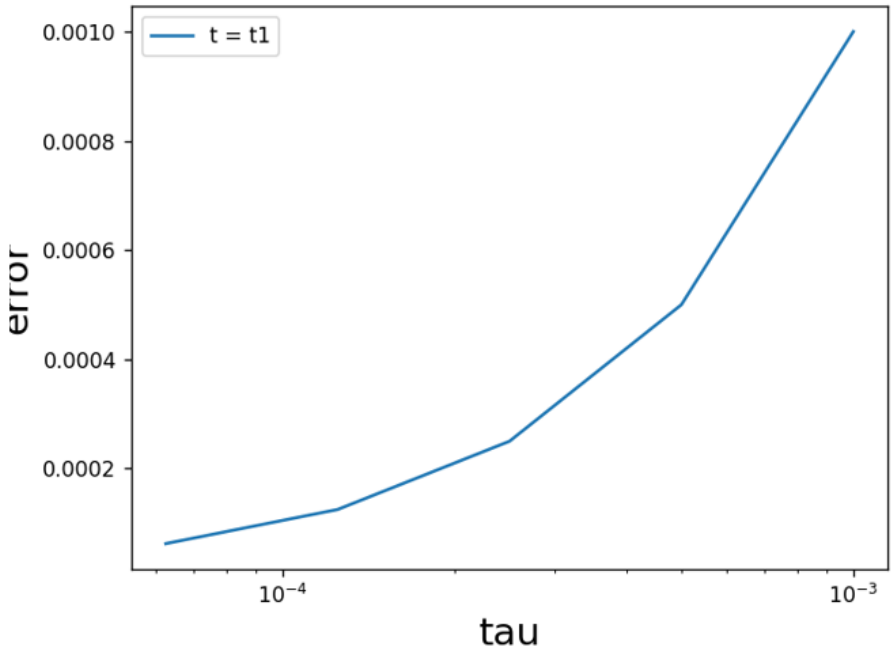
N	τ	$s(t=t_{n1})$	$s(t=t_{n2})$	$max(t=t_{n1})$	$max(t=t_{n2})$
10	0.0066666666666668	0.0022674741544877535	0.0019562737170215607	0.0062074545932790 94	0.0056634297520218 87
20	0.0016666666666667	0.0005234564947474022	0.0004352389508019068 3	0.0015119359189149084	0.0013791454934390934
30	0.0007407407407407407	0.0002224101393639191947	0.0001828712778376997 4	0.0006677436354470268	0.0006098676236349632
40	0.0004166666666666675	0.0001215215736805122 8	0.0001007694606146534	0.0003752087913165836	0.0003431929399962397
50	0.0002666666666666667	7.601905147162651e-05	6.467562906542223e-05	0.0002398955662842317	0.0002205481949049 1262

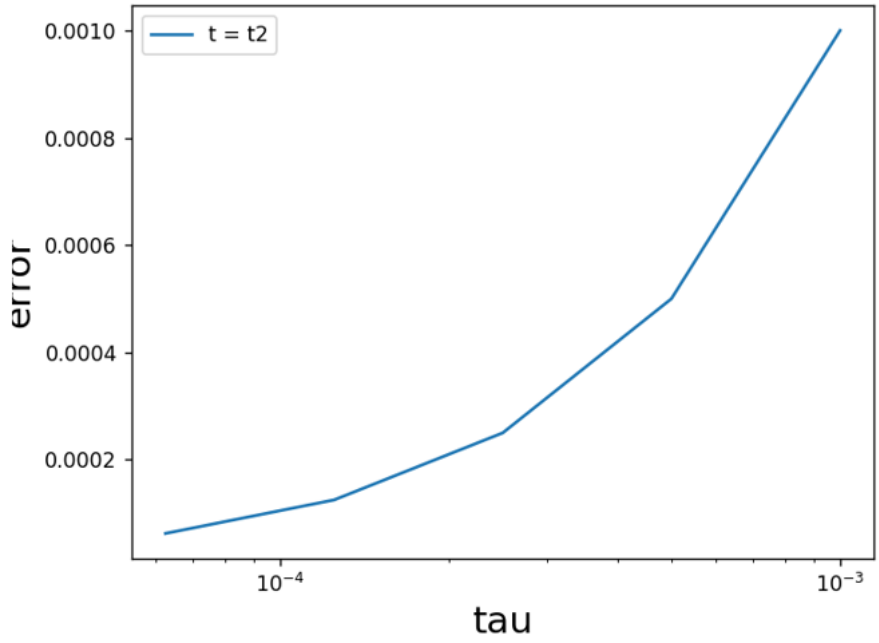
3. Неявная схема (аппроксимация производной левой разностью)



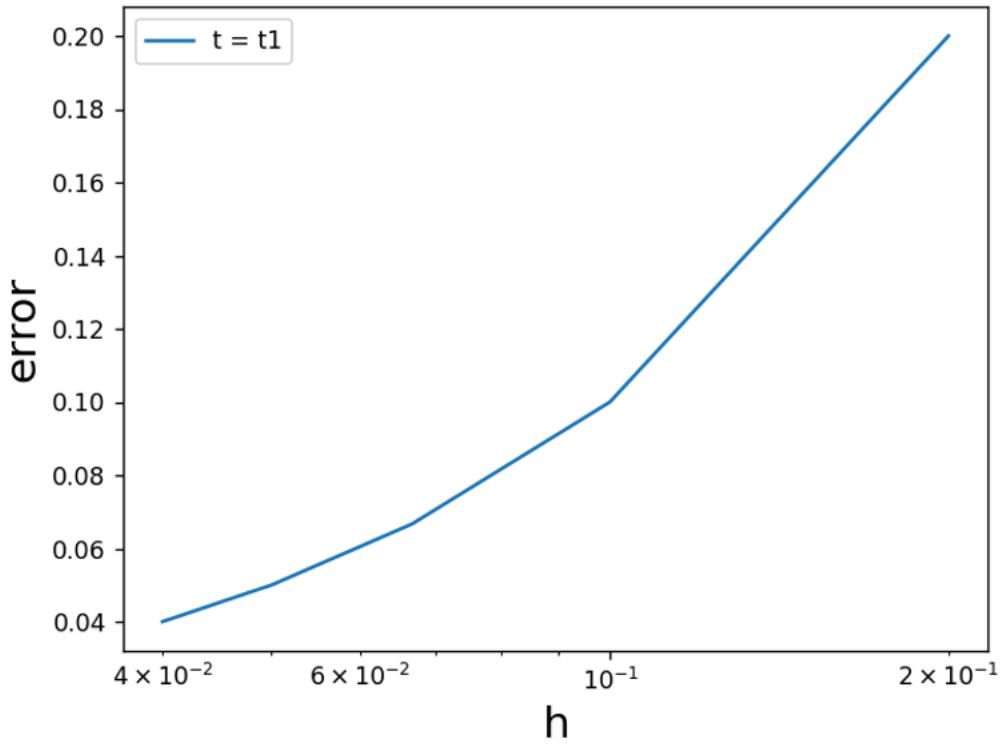


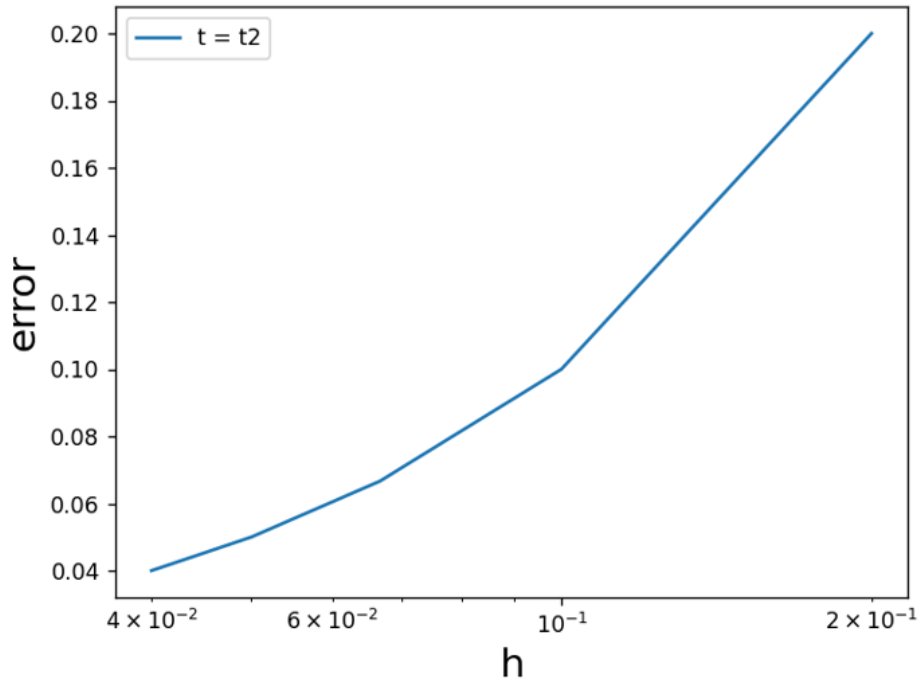
N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.001	6.056927515732742e-05	5.560599999818003e-05	0.000105493898444 05344	9.378349806493 658e-05
10	0.0005	3.031830644446104 8e-05	2.782627433776835e-05	5.275018347339344e-05	4.689504213761 797e-05
10	0.00025	1.516758430693080 4e-05	1.391895696167813 6e-05	2.637591699411468e-05	2.344835154427 516e-05
10	0.000125	7.585901676694193e-06	6.960933602714832 5e-06	1.318816686618484 7e-05	1.172438423713 2845e-05
10	6.25e-05	3.793478417572562e-06	3.480830575032597 5e-06	6.594135842974413 5e-06	5.862244419452 267e-06


t = t1
0.0010
0.0008
0.0006
0.0004
0.0002
error
10^{-4}
10^{-3}
tau


t = t2
0.0010
0.0008
0.0006
0.0004
0.0002
error
10^{-4}
10^{-3}
tau

N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.0001	0.004236296975373478	0.00486241780649932	0.008978720872951484	0.0116732829923063
20	0.0001	0.0011998755819615935	0.0014973009371620804	0.0033080552589628276	0.0045531991388958956
30	0.0001	0.0006182574891691393	0.0008119282247440024	0.0019635367170449967	0.0027673486989721052
40	0.0001	0.0003994006738963085	0.0005426658708599314	0.0013853106848866759	0.0019777071484201603
50	0.0001	0.0002898177519063827 4	0.0004033472071960195	0.0010672239035709463	0.0015359560271459571

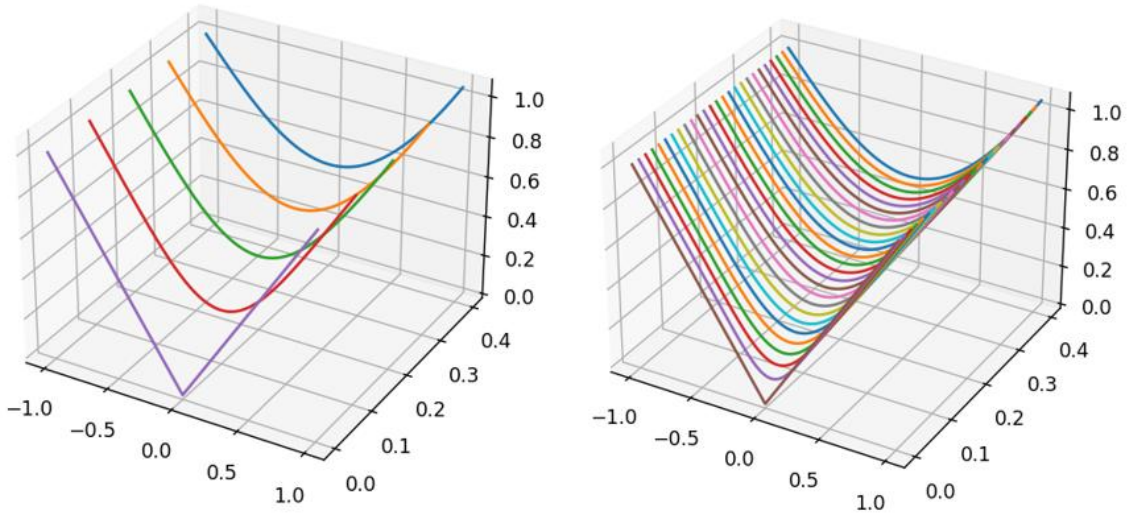




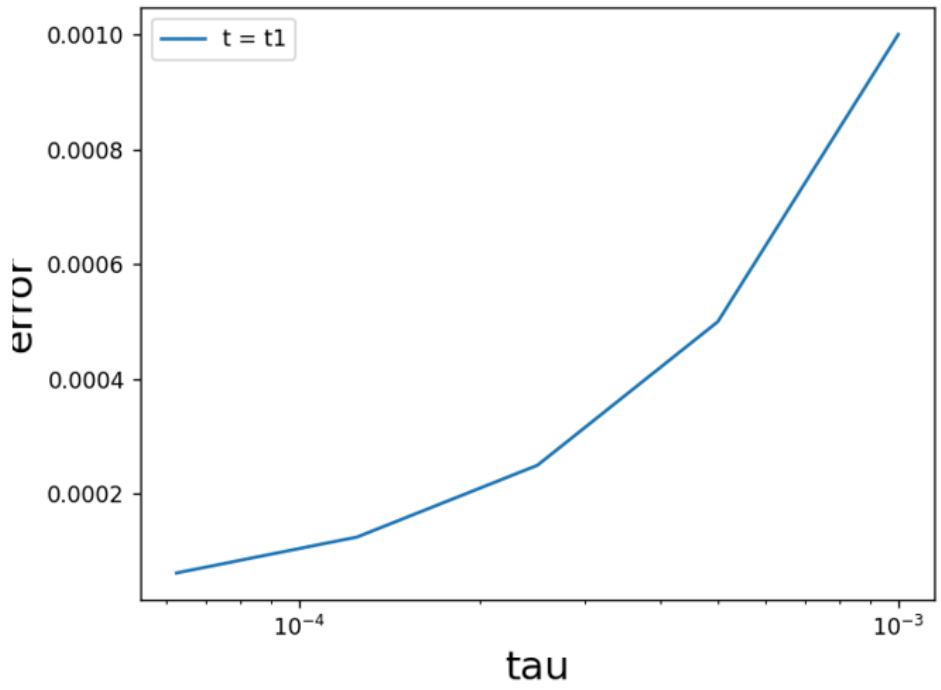
N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.0066666666666668	0.00040081308343474926	0.00037146485586443323	0.0007089772220322255	0.0006339588696744891
20	0.0016666666666667	9.736899553480697e-05	9.050399379662022e-05	0.00016848025288257462	0.00015239298590802308
30	0.0007407407407407407	4.2648490024168075e-05	3.9705400962739546e-05	7.419672907482688e-05	6.734024484350565e-05
40	0.0004166666666666675	2.382437189004899e-05	2.2211752208835558e-05	4.165638872394295e-05	3.788943604232653e-05
50	0.0002666666666666667	1.517573414030119e-05	1.4151572651145403e-05	2.662225534383334e-05	2.4231570025567972e-05

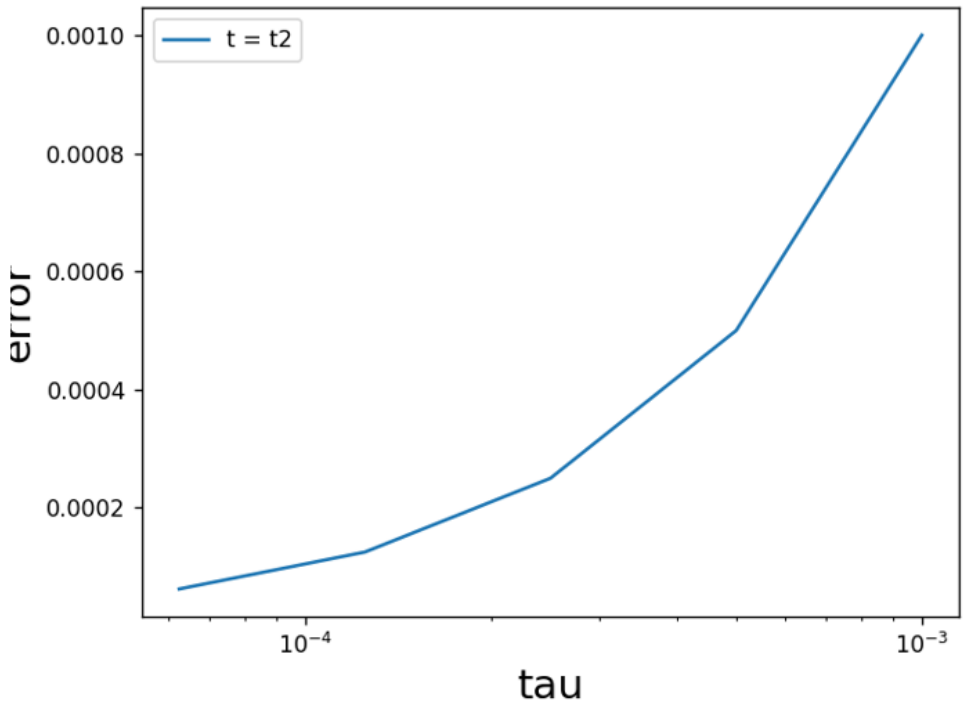
N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.0066666666666668	0.0042063733949325 59	0.0047725854201661 7	0.0087759276495944 5	0.0112798385839840 7
20	0.0016666666666667	0.0011930259781652009	0.0014882345193873666	0.0032698407819962 14	0.0045127301973402 2
30	0.0007407407407407407	0.0006176428249157221	0.0008126591305455549	0.0019589778509545575	0.0027689558620047627
40	0.0004166666666666675	0.0003985736329506855 6	0.0005418490909154982	0.0013810467528181 38	0.0019739801780684907
50	0.0002666666666666667	0.0002896584858765553	0.0004036520773485 3085	0.0010662676737043242	0.0015369567819936147

4. **Неявная схема (аппроксимация производной центральной разностью)**

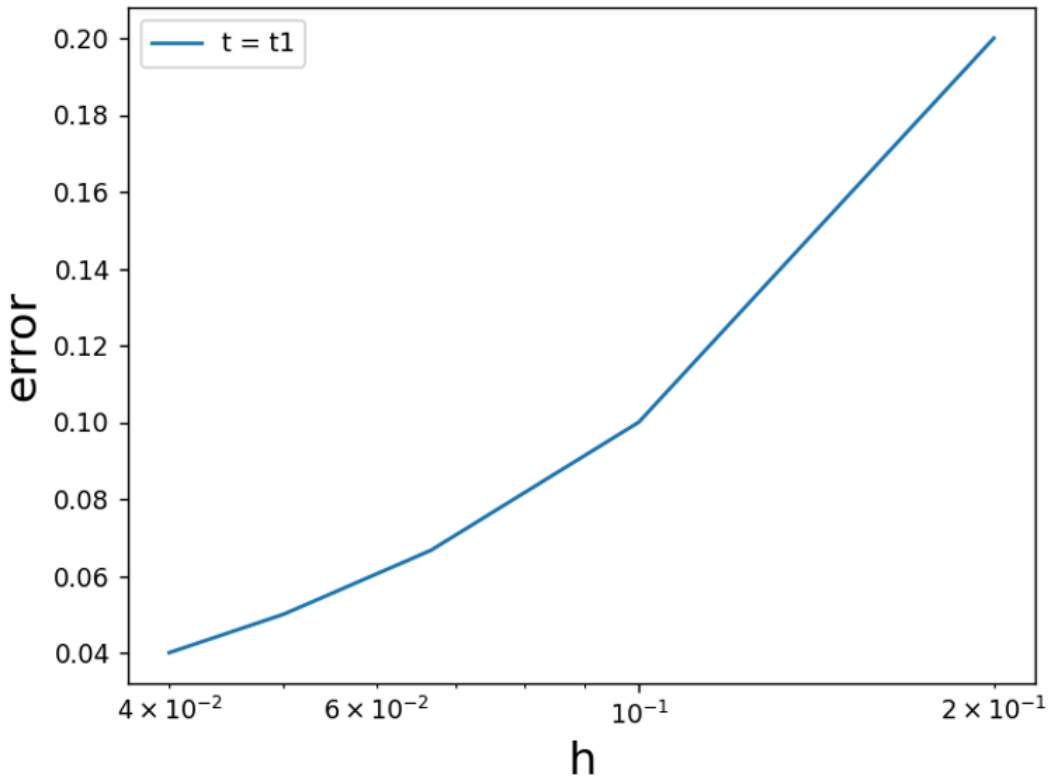


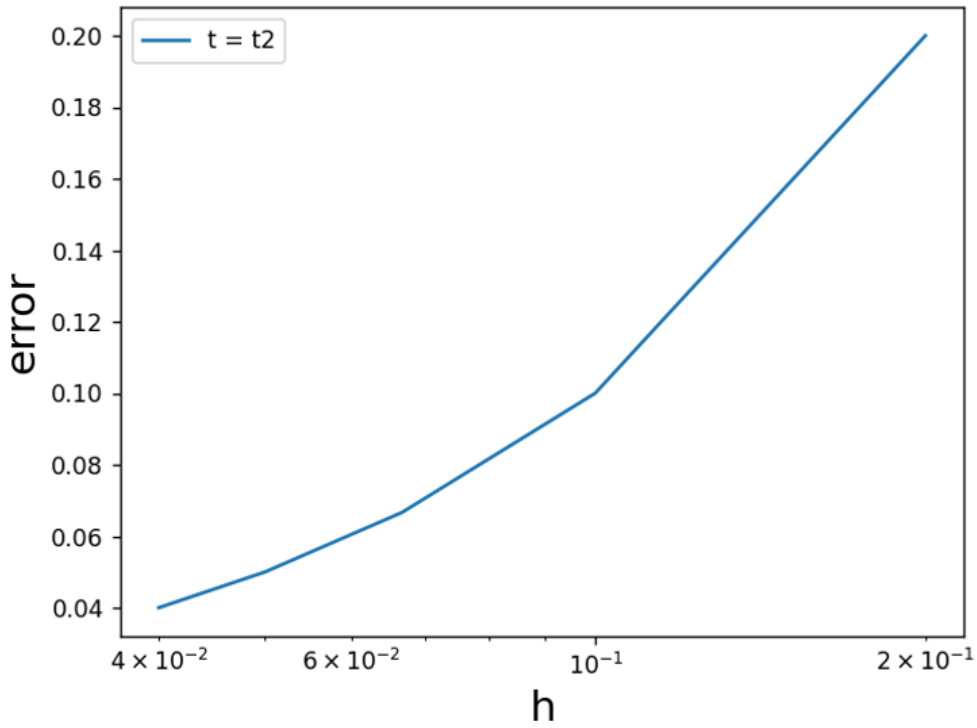
N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.001	5.793107408402738e-05	5.477784808614859e-05	0.00010695081977263321	9.631608672167058e-05
10	0.0005	2.899395034130058e-05	2.741185009459592e-05	5.347543604822258e-05	4.816015395597839e-05
10	0.00025	1.450409070466340e-05	1.371166286450940e-05	2.673773292838355e-05	2.408060695741 0655e-05
10	0.000125	7.253825789322637e-06	6.857266708236671e-06	1.336887118980056 9e-05	1.204043620511 8521e-05
10	6.25e-05	3.627358175224580 4e-06	3.428992245303698e-06	6.684436968162899e-06	6.020251394067 699e-06





N	τ	$s(t = t_{n1})$	$s(t = t_{n2})$	$max(t = t_{n1})$	$max(t = t_{n2})$
10	0.0001	0.002357161539402026	0.0020905698820374023	0.006239133761849314	0.005653102438690771
20	0.0001	0.0005655221631385558	0.0004986599566044883	0.0015106801340841192	0.0013770085310397806
30	0.0001	0.0002486718413881213 4	0.0002187860965882 1296	0.0006678091034396649	0.0006092792549728387
40	0.0001	0.0001392156002678628 6	0.0001223422317388 3315	0.0003749475685057 5683	0.0003421922878909 5224
50	0.0001	8.886207258034275 e-05	7.803580724370769 e-05	0.0002397621196171 2623	0.0002188478543581 0628


t = t1
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
error
4×10^{-2}
6×10^{-2}
10^{-1}
2×10^{-1}
h


t = t2
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
error
4×10^{-2}
6×10^{-2}
10^{-1}
2×10^{-1}
h

N	τ	$s(t=t_{n1})$	$s(t=t_{n2})$	$max(t=t_{n1})$	$max(t=t_{n2})$
10	0.0066666666666668	0.00038334368250451034	0.00036443619715620395	0.0007187136501662916	0.0006500466023758866
20	0.0016666666666667	9.451522671518745e-05	8.890945898801281e-05	0.00016931208449499913	0.0001540713314651665
30	0.0007407407407407407	4.1762317606881365e-05	3.916545349419991e-05	7.44156612785262 8e-05	6.78063713632215 4e-05
40	0.0004166666666666675	2.344265952858733e-05	2.196667400805825 8e-05	4.17425701582652 1e-05	3.80769960668381 2e-05
50	0.0002666666666666667	1.497866877734244e-05	1.402208723052107 2e-05	2.66649066800739 3e-05	2.43256661921487 1e-05

N	τ	$s(t=t_{n1})$	$s(t=t_{n2})$	$max(t=t_{n1})$	$max(t=t_{n2})$
10	0.0066666666666668	0.002389066819482074	0.0021357747123157394	0.00636338510639517	0.0057779636290029 02
20	0.0016666666666667	0.000568821503227 8197	0.000501448927838 9154	0.00151958788233 01978	0.0013834973571926756
30	0.0007407407407407407	0.000249078231272 1299	0.000218897193576 7515	0.00066896410634 82576	0.0006096724840926049
40	0.0004166666666666675	0.000139417155030 80693	0.000122486547507 44882	0.00037544241780 43503	0.0003425107473073452
50	0.0002666666666666667	8.890985387392975e-05	7.803216760063505e-05	0.00023988668445 190653	0.0002188571149 4162852

Выводы

Явная разностная схема решает уравнение в каждый последующий момент времени, используя значения в предыдущие моменты времени. Таким образом, она явно выражает следующее значение функции через предыдущие значения. Это означает, что для вычисления значения в каждый момент времени необходимо знать значения функции в предыдущие моменты времени. Явная схема легко реализуется, но может иметь ограничения по устойчивости и точности, которые зависят от выбора шага по времени и пространству.

Неявная разностная схема использует значения функции в следующем моменте времени, выражая их через значения функции в текущий момент времени. Таким образом, она неявно выражает следующее значение через текущее значение. Это означает, что для вычисления значения в каждый момент времени необходимо решать систему уравнений, что требует больших вычислительных затрат. Неявная схема более устойчива и точна, но её реализация может быть более сложной и требовательной к ресурсам.

Для явной разностной схемы с аппроксимацией первой производной левой разностью порядок сходимости по пространственному шагу h будет равен $O(h)$, а по временному шагу $\tau - O(\tau)$. Для явной разностной схемы с аппроксимацией первой производной центральной разностью порядок сходимости по пространственному шагу h будет равен $O(h^2)$, а по временному шагу $\tau - O(\tau)$.

Для неявной разностной схемы с аппроксимацией первой производной левой разностью порядок сходимости по пространственному шагу h будет равен $O(h)$, а по временному шагу $\tau - O(\tau)$. Для неявной разностной схемы с аппроксимацией первой производной центральной разностью порядок сходимости по пространственному шагу h будет равен $O(h^2)$, а по временному шагу $\tau - O(\tau)$.

Более точные аппроксимации граничных условий второго рода приводят к увеличению порядка сходимости разностной схемы. Это связано с тем, что при более точной аппроксимации граничных условий уменьшается ошибка на границе области, что приводит к уменьшению глобальной ошибки во всей области и, следовательно, к увеличению порядка сходимости. Однако, более точные аппроксимации могут привести к увеличению сложности вычислений и увеличению количества вычислительных операций, что может оказать негативное влияние на производительность и время вычислений.