**Лабораторная работа №4**

**«Численное решение смешанной задачи для уравнения теплопроводности»**

Вариант 10

Выполнил студент 3 курса 2 группы ФПМИ

Сараев Владислав Максимович

Минск, 2020

**Постановка задачи**

На сетке узлов найти численное решение смешанной задачи для одномерного уравнения теплопроводности с использованием:

* явной разностной схемы с и
* чисто неявной разностной схемы с
* разностной схемы Кранка-Николсон с

Выписать соответствующие разностные схемы, указать их порядок аппроксимации, указать являются ли схемы абсолютно устойчивыми по начальным данным. Вычислить погрешность численного решения (т.е. найти ). Построить графики, демонстрирующие устойчивое и неустойчивое поведение явной разностной схемы.

Задача:

**Краткие теоретические сведения**

Имеем одномерное уравнение теплопроводности с постоянными коэффициентами в прямоугольнике :

при этом функции, задающие дополнительные условия, должны быть согласованы, т.е. и .

Введем равномерную пространственно-временную сетку

.

Для аппроксимации уравнения будем использовать следующую разностную схему:

где – сеточная функция, аппроксимация .

Данная схема в индексной форме имеет вид:

Дальнейшее решение зависит от параметра :

1. . Тогда получаем разностная схема становится **явной**:

или в индексной форме:

Тогда порядок расчетов будет следующим:

1. заполняем нулевой слой по формулам:
2. для всех заполняем очередной -ый слой по формулам:

, где

1. . Тогда получаем **неявную** разностную схему:

или

где . Решение данной системы можно найти с помощью метода прогонки. Данная схема при называется **чисто неявной схемой** или **схемой с опережением**, а при **симметричной схемой** или **схемой Кранка-Николсон**.

Порядок аппроксимации и устойчивость:

1. явная разностная схема:

* не является абсолютно устойчивой (устойчива при )

1. чисто неявная разностная схема:

* является абсолютно устойчивой

1. разностная схема Кранка-Николсон

* , при
* является абсолютно устойчивой

**Листинг**

**import** numpy **as** np

**from** math **import** e**,** cos**,** sin**,** fabs

**import** matplotlib**.**pyplot **as** plt

L1 **=** 0

R1 **=** 1

L2 **=** 0

R2 **=** 0.5

**def** u**(**x**,** t**):**

**return** e**\*\*(-**t**)** **\*** cos**(**x **+** t**)**

**def** u0**(**x**):**

**return** cos**(**x**)**

**def** mu0**(**t**):**

**return** e**\*\*(-**t**)** **\*** cos**(**t**)**

**def** mu1**(**t**):**

**return** e**\*\*(-**t**)** **\*** cos**(**1 **+** t**)**

**def** f**(**x**,** t**):**

**return** **-**e**\*\*(-**t**)** **\*** sin**(**x **+** t**)**

# значения функций в узлах

**def** init\_values**(**h**,** tau**):**

x\_values **=** np**.**linspace**(**L1**,** R1**,** **int((**R1 **-** L1**)** **/** h**)** **+** 1**)**

t\_values **=** np**.**linspace**(**L2**,** R2**,** **int((**R2 **-** L2**)** **/** tau**)** **+** 1**)**

u0\_values **=** np**.**array**([**u0**(**x**)** **for** x **in** x\_values**])**

mu0\_values **=** np**.**array**([**mu0**(**t**)** **for** t **in** t\_values**])**

mu1\_values **=** np**.**array**([**mu1**(**t**)** **for** t **in** t\_values**])**

y\_values **=** np**.**zeros**((len(**t\_values**),** **len(**x\_values**)))**

u\_values **=** np**.**array**([**np**.**array**([**u**(**x**,** t**)** **for** x **in** x\_values**])** **for** t **in** t\_values**])**

**return** x\_values**,** t\_values**,** u0\_values**,** mu0\_values**,** mu1\_values**,** y\_values**,** u\_values

# расчет погрешности

**def** fault**(**y\_values**,** u\_values**,** N1**,** N2**):**

max\_delta **=** 0

**for** i **in** **range(**N2**):**

**for** j **in** **range(**N1**):**

delta **=** fabs**(**y\_values**[**i**][**j**]** **-** u\_values**[**i**][**j**])**

**if** delta **>** max\_delta**:**

max\_delta **=** delta

**return** max\_delta

# вывод графика

**def** show\_graphic**(**x\_values**,** t\_values**,** y\_values**,** u\_values**):**

ax **=** plt**.**axes**(**projection**=**"3d"**)**

xgrid**,** tgrid **=** np**.**meshgrid**(**x\_values**,** t\_values**)**

ax**.**plot\_wireframe**(**xgrid**,** tgrid**,** u\_values**,** color**=**'green'**,** label**=**"Function"**)**

ax**.**plot\_wireframe**(**xgrid**,** tgrid**,** y\_values**,** color**=**'red'**,** label**=**"Approximation"**)**

ax**.**set\_xlabel**(**'x'**)**

ax**.**set\_ylabel**(**'t'**)**

ax**.**set\_zlabel**(**'y'**)**

plt**.**legend**()**

plt**.**show**()**

# прогонка

**def** tridiagonal\_matrix\_algorithm**(**matrix**,** f**):**

n **=** **len(**f**)**

matrix**[**2**][**0**]** **/=** matrix**[**1**][**0**]**

f**[**0**]** **/=** matrix**[**1**][**0**]**

matrix**[**1**][**0**]** **=** 1

**for** i **in** **range(**1**,** n **-** 1**):**

coeff **=** matrix**[**1**][**i**]** **-** matrix**[**2**][**i **-** 1**]** **\*** matrix**[**0**][**i **-** 1**]**

matrix**[**2**][**i**]** **/=** coeff

f**[**i**]** **=** **(**f**[**i**]** **-** f**[**i **-** 1**]** **\*** matrix**[**0**][**i **-** 1**])** **/** coeff

matrix**[**1**][**i**]** **=** 1

matrix**[**0**][**i **-** 1**]** **=** 0

f**[**n **-** 1**]** **=** **(**f**[**n **-** 1**]** **-** f**[**n **-** 2**]** **\*** matrix**[**0**][**n **-** 2**])** **/** **(**matrix**[**1**][**n **-** 1**]** **-** matrix**[**2**][**n **-** 2**]** **\*** matrix**[**0**][**n **-** 2**])**

matrix**[**1**][**n **-** 1**]** **=** 1

**for** i **in** **range(**n **-** 2**,** **-**1**,** **-**1**):**

f**[**i**]** **-=** f**[**i **+** 1**]** **\*** matrix**[**2**][**i**]**

matrix**[**2**][**i**]** **=** 0

# явный метод

**def** explicit\_method**(**h**,** tau**):**

gamma **=** tau **/** **(**h **\*\*** 2**)**

x\_values**,** t\_values**,** u0\_values**,** mu0\_values**,** mu1\_values**,** y\_values**,** u\_values **=** init\_values**(**h**,** tau**)**

N1 **=** **len(**x\_values**)**

N2 **=** **len(**t\_values**)**

**def** fi**(**x**,** t**):**

**return** f**(**x**,** t **+** tau**)**

fi\_values **=** np**.**array**([**np**.**array**([**fi**(**x**,** t**)** **for** x **in** x\_values**])** **for** t **in** t\_values**])**

# заполняю нулевой слой

**for** i **in** **range(**N1**):**

y\_values**[**0**][**i**]** **=** u0\_values**[**i**]**

# для всех слоев j от 1 до N1-1

**for** j **in** **range(**1**,** N2**):**

# берем y\_0 из левого граничного условия

y\_values**[**j**][**0**]** **=** mu0\_values**[**j**]**

# рекуррентно вычисляем y\_1 - y\_(N1-2) через предыдущий слой

**for** i **in** **range(**1**,** N1 **-** 1**):**

y\_values**[**j**][**i**]** **=** **(**1 **-** 2 **\*** gamma**)** **\*** y\_values**[**j **-** 1**][**i**]** **+** gamma **\*** **(**

y\_values**[**j **-** 1**][**i **-** 1**]** **+** y\_values**[**j **-** 1**][**i **+** 1**])** **+** tau **\*** fi\_values**[**j **-** 1**][**i**]**

# берем y\_(N1-1) из правого граничного условия

y\_values**[**j**][**N1 **-** 1**]** **=** mu1\_values**[**j**]**

max\_delta **=** fault**(**y\_values**,** u\_values**,** N1**,** N2**)**

show\_graphic**(**x\_values**,** t\_values**,** y\_values**,** u\_values**)**

**return** max\_delta

# неявный метод

**def** implicit\_method**(**h**,** tau**,** sigma**):**

x\_values**,** t\_values**,** u0\_values**,** mu0\_values**,** mu1\_values**,** y\_values**,** u\_values **=** init\_values**(**h**,** tau**)**

N1 **=** **len(**x\_values**)**

N2 **=** **len(**t\_values**)**

**def** fi**(**x**,** t**):**

**if** sigma **==** 1 **/** 2**:**

**return** f**(**x**,** t **+** tau **/** 2**)**

**return** f**(**x**,** t **+** tau**)**

fi\_values **=** np**.**array**([**np**.**array**([**fi**(**x**,** t**)** **for** x **in** x\_values**])** **for** t **in** t\_values**])**

# заполняю нулевой слой

**for** i **in** **range(**N1**):**

y\_values**[**0**][**i**]** **=** u0\_values**[**i**]**

# коэффициент на главной диагонали трехдиагональной матрицы

coef1 **=** sigma **/** **(**h **\*\*** 2**)**

# коэффициенты на поддиагонали и наддиагонали

coef2 **=** **-(**1 **/** tau **+** 2 **\*** sigma **/** **(**h **\*\*** 2**))**

# для всех слоев j от 1 до N1-1

**for** j **in** **range(**1**,** N2**):**

# создаю трехдиагональную матрицу, где на главной диагонали стоит coef1,

# а на поддиагонали и наддиагонали стоят coef2

# матрица хранится ввиде трех векторов

# |coef1 coef2 0 ... 0 0| - строка для вычисления y\_1 = b\_1

# |coef2 coef1 coef2 0 ... 0| - строка для вычисления y\_2 = b\_2

# |0 coef2 coef1 coef2 0 ... 0| - строка для вычисления y\_3 = b\_3

# | ..... |

# |0 ... 0 coef2 coef1 coef2| - строка для вычисления y\_(N1-3) = b\_(N1-3)

# |0 ... 0 0 coef2 coef1 | - строка для вычисления y\_(N1-2) = b\_(N1-2)

matrix **=** np**.**array**(**

**[**np**.**full**(**N1 **-** 3**,** coef1**),** np**.**full**(**N1 **-** 2**,** coef2**),** np**.**full**(**N1 **-** 3**,** coef1**)]**

**)**

# правая часть уравнения

b **=** np**.**zeros**(**N1 **-** 2**)**

# вычисление y\_0 и y\_(N1-1)

y\_values**[**j**][**0**]** **=** mu0\_values**[**j**]** # (1)

y\_values**[**j**][**N1 **-** 1**]** **=** mu1\_values**[**j**]** # (2)

# вычисляю b\_1 как -F\_ji - coef1 \* y\_0

b**[**0**]** **=** **-**coef1 **\*** y\_values**[**j**][**0**]** **-** **(**y\_values**[**j **-** 1**][**1**]** **/** tau **+** **(**1 **-** sigma**)** **\***

**(**y\_values**[**j **-** 1**][**0**]** **-** 2 **\*** y\_values**[**j **-** 1**][**1**]** **+** y\_values**[**j **-** 1**][**2**])** **/** **(**

h **\*\*** 2**)** **+**

fi\_values**[**j **-** 1**][**1**])**

# вычисляю b\_2 - b\_(N1-3) включительно как -F\_ji

**for** i **in** **range(**1**,** N1 **-** 3**):**

b**[**i**]** **=** **-** **(**y\_values**[**j **-** 1**][**i **+** 1**]** **/** tau **+** **(**1 **-** sigma**)** **\***

**(**y\_values**[**j **-** 1**][**i**]** **-** 2 **\*** y\_values**[**j **-** 1**][**i **+** 1**]** **+** y\_values**[**j **-** 1**][**i **+** 2**])** **/** **(**h **\*\*** 2**)** **+**

fi\_values**[**j **-** 1**][**i **+** 1**])**

# вычисляю b\_(N-2) как -F\_ji - coef1 \* y\_(N1-1)

b**[**N1 **-** 3**]** **=** **-**coef1 **\*** y\_values**[**j**][**N1 **-** 1**]** **-** \

**(**y\_values**[**j **-** 1**][**N1 **-** 2**]** **/** tau **+** **(**1 **-** sigma**)** **\***

**(**y\_values**[**j **-** 1**][**N1 **-** 3**]** **-** 2 **\*** y\_values**[**j **-** 1**][**N1 **-** 2**]**

**+** y\_values**[**j **-** 1**][**N1 **-** 1**])** **/** **(**h **\*\*** 2**)** **+** fi\_values**[**j **-** 1**][**N1 **-** 2**])**

# метод прогонки

tridiagonal\_matrix\_algorithm**(**matrix**,** b**)**

# заполняю y с 1 по (N1-2) индексы включительно

# y\_0 и y\_(N1-2) заполнены в (1) и (2)

**for** i **in** **range(**1**,** N1 **-** 1**):**

y\_values**[**j**][**i**]** **=** b**[**i **-** 1**]**

max\_delta **=** fault**(**y\_values**,** u\_values**,** N1**,** N2**)**

show\_graphic**(**x\_values**,** t\_values**,** y\_values**,** u\_values**)**

**return** max\_delta

**if** \_\_name\_\_ **==** "\_\_main\_\_"**:**

# явная разностная схема с tau = h = 0.1

**print(**explicit\_method**(**0.1**,** 0.1**))**

# явная разностная схема с h = 0.1 и tau = h^2/2

**print(**explicit\_method**(**0.1**,** 0.005**))**

# чисто неявная разностная схема с tau = h = 0.1

**print(**implicit\_method**(**0.1**,** 0.1**,** 1**))**

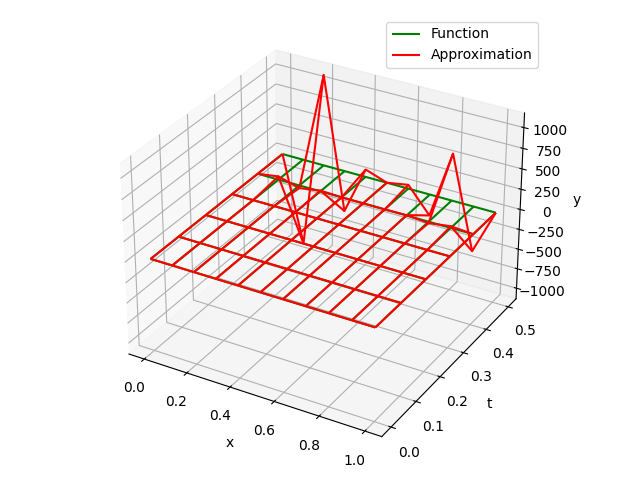
# разностная схема Кранка-Николсон с tau = h = 0.1

**print(**implicit\_method**(**0.1**,** 0.1**,** 1 **/** 2**))**

**Результаты**

1. Явная разностная схема с

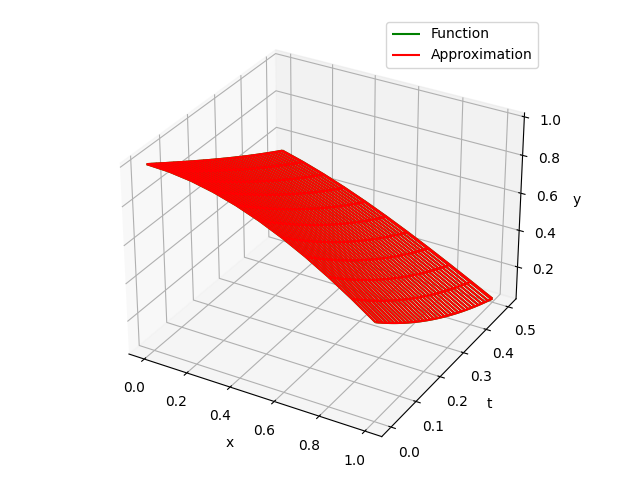
= 1127.868933310336



Как видно из графика и погрешности, метод неустойчив при , т.к. условие его устойчивости не выполнено.

1. Явная разностная схема с

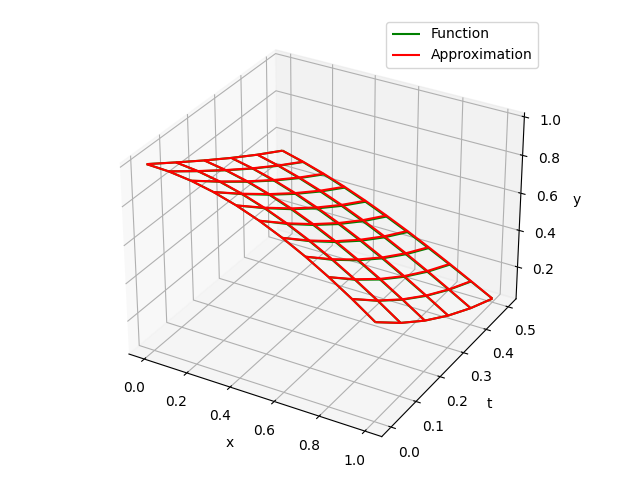
= 0.00031326119148944453



Как видно из графика и погрешности, метод устойчив при , т.к. условие его устойчивости выполнено.

1. Чисто неявная разностная схемы с

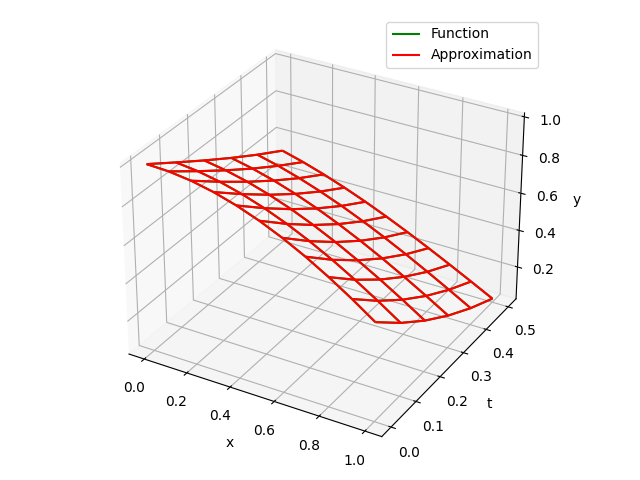
= 0.006208571445543987



Как видно из графика и погрешности, метод устойчив при , т.к. он является абсолютно устойчивым.

1. Разностная схема Кранка-Николсон с

= 0.00010642457938037087



Как видно из графика и погрешности, метод устойчив при , т.к. он является абсолютно устойчивым.

**Выводы**

С помощью разностных схем можно достаточно точно найти численное решение смешанной задачи для одномерного уравнения теплопроводности. Для этого можно использовать как явный метод, так и неявные. Явный метод проще в реализации, но для его устойчивости необходим достаточно малый шаг по t. В свою очередь, неявный метод хоть и требует решение (t - 1) СЛАУ с трехдиагональной матрицей, абсолютно устойчивым и точнее, если использовать схему Кранка-Николсон.