Министерство образования РФ

Государственное образовательное учреждение

высшего профессионального образования

“Тверской государственный университет”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет прикладной математики и кибернетики

**ВЫПУСКНАЯ РАБОТА БАКАЛАВРА**

Направление: «*Прикладная математика и информатика»*

Специализация: *Бакалавриат*

**Тема:** *«Численный анализ тепловых процессов с учетом фазового перехода с использованием метода сеток»*

**Автор:**

студент 44 группы

Доронин Виталий Евгеньевич

*Тверь – 2018*

Оглавление

[Исходная задача 3](#_Toc500083170)

[Неявная схема 3](#_Toc500083171)

[Расчеты на ЭВМ 4](#_Toc500083172)

[Нахождение численного решения 4](#_Toc500083173)

[Анализ результатов 5](#_Toc500083174)

[Код программы 7](#_Toc500083175)

[Список использованной литературы 11](#_Toc500083176)

# Постановка задачи

При изменении температуры тела может происходить изменение его физического состояния, в частности при переходе температуры через точку плавления – переход из твёрдой фазы в жидкую или наоборот. На поверхности фазового перехода всё время сохраняется постоянная температура. При движении поверхности фазового перехода происходит выделение скрытой теплоты плавления (затвердевания).

Математической моделью, описывающей процесс фазового

перехода, является задача с подвижной границей (задача

Стефана).

Рассмотрим процесс замерзания воды, при котором температура фазового переходы равна нулю. Будем рассматривать массу воды , ограниченную с одной стороны плоскостью . В начальный момент вода обладает постоянной температурой . Если на поверхности всё время поддерживается постоянная температура , то граница замерзания будет со временем проникать вглубь жидкости.

Задача о распределении температуры при наличии фазового перехода и о скорости движения границы раздела фаз (например, внутри замерзающей воды) сводится к решению уравнений

(1)

с дополнительными условиями

(2)

и условиями на границе замерзания

(3)

*, (4)*

где – коэффициенты теплопроводности и температуропроводности твердой и, соответственно, жидкой фаз.

Задачу (1) – (4) часто называют задачей Стефана, задачей о фазовом переходе или задачей о промерзании.

# Решение

Предположим, что имеется две фазы 1, 2 с коэффициентами теплоёмкости .

Имеем слой конечной толщины, . Требуется найти поле температур :

Аппроксимируем производные с помощью их разностных аналогов:

Получим :

Чисто неявной разностной схемой для уравнения теплопроводности (схемой с опережением) называется разностная схема, имеющая вид:

, (4)

, .

В нашей задаче будем предполагать, что . Схема имеет первый порядок аппроксимации по τ и второй – по h. Решение системы (4) находится по слоям, начиная с n = 1. Однако здесь, в отличие от явной схемы, для нахождения по известным требуется решить систему уравнений

, где , ,

.

В нашей задаче будем считать, что на границе задана функция

*.*

Граничные условия:

# Расчеты на ЭВМ

## Нахождение численного решения

Для численного значения интеграла будем использовать R – язык программирования для статистической обработки данных и работы с графикой, а также свободная программная среда вычислений с открытым исходным кодом в рамках проекта GNU. R широко используется как статистическое программное обеспечение для анализа данных и фактически стал стандартом для статистических программ.

Для расчётов будем использовать начальные значения:

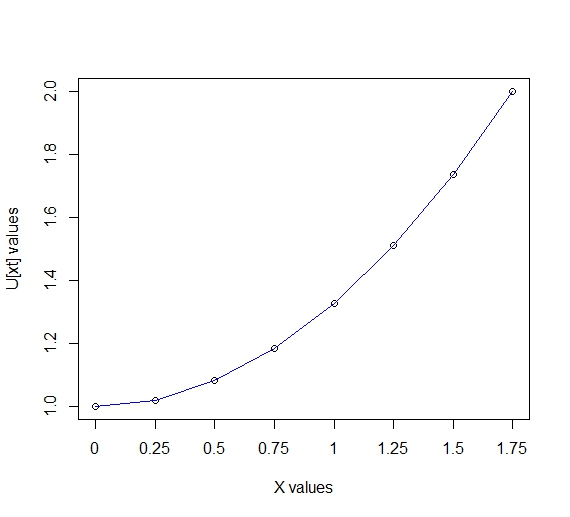
k1=k2 =0, h = 0,25, a = 1, τ = 0.01, L = 1.75, A = , B = ,

n = 5, где n - Число слоёв неявной схемы.

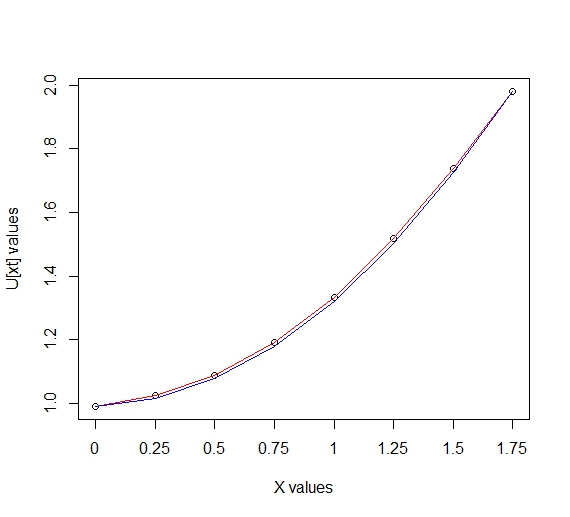
Для сравнения численного решения с точным , будем использовать формулу:

## Анализ результатов

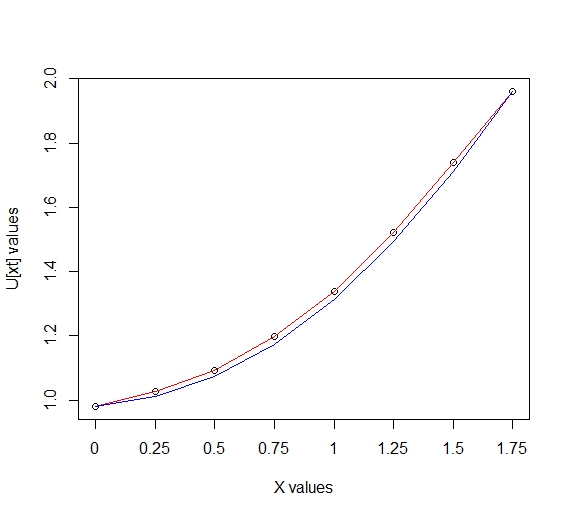
При =0 :



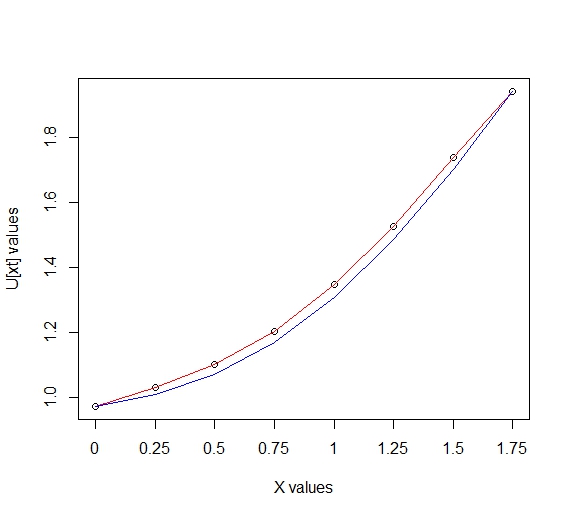
При = 1 :



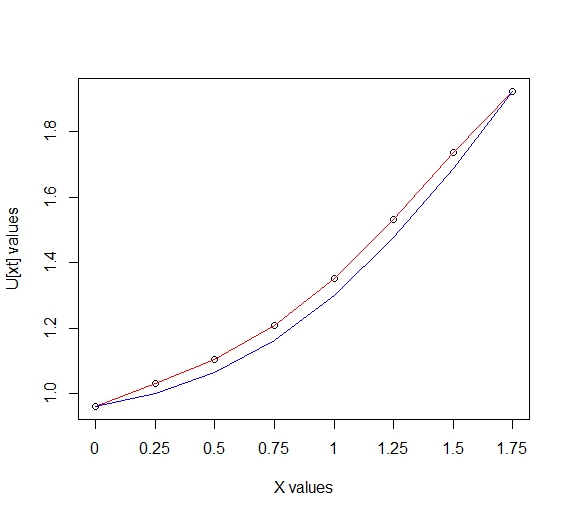
При =2 :



При =3 :



При = 4 :



Здесь синим цветом выделено точное решение, а красным - численное

На нулевом шаге численное и точное решения полностью совпадают. Мы видим, что при увеличении числа шагов, численное решение всё больше отстаёт от точного.

## Код программы

rm**(**list **=** ls**())**

k1**=**k2**=**1

k2\_U**=** 0.5786

k1\_U**=** 2.3

c2\_U **=** 4195

c1\_U **=** 2000

a**=(**k1\_U**/**c1\_U**)^(**1**/**2**)**

h **=** 0.250 #шаг по x

print**(**h**)**

tau **=** h**^**2**/**a**^**2

L **=** 2 #длина

N **=** round**(**L**/**h**)** #Число шагов

n **=** 30 #Число шагов по времени

tj **=** numeric**(**n**)** #tau j

nj **=** numeric**(**n**)** #ветор моментов времени

kU **=** **function(**U\_i**)**

**{**

**if(** U\_i **<** 0**)**

return**(**k1\_U**)**

return**(**k2\_U**)**

**}**

CU **=** **function(**U\_i**)**

**{**

**if(** U\_i **<** 0**)**

return**(**c1\_U**)**

return**(**c2\_U**)**

**}**

A **=** **function(**t**)**

**{**

#0

1**/(**1**+**t**)**

**}**

B **=** **function(**t**)**

**{**

#0

1**/(**1**+**t**)**

#1

**}**

x **=** numeric**(**N**+**1**)**

fij **=** numeric**(**N**+**1**)** #сила

Ai **=** numeric**(**N**)** #коэффициенты

Bi **=** numeric**(**N**)**

Ci **=** numeric**(**N**)**

Fi **=** numeric**(**N**-**1**)**

alpha **=** numeric**(**N**)**

beta **=** numeric**(**N**)**

#j=1 0-й слой

t\_j **=** **function(**r**)**

**{**

r**^**2**/**a**^**2

**}**

**for(**j **in** 1**:**n**)**

**{**

tj**[**j**]=(**j**-**1**)\***tau

nj**[**j**]=**j**-**1

**}**

Ux0**=function(**x**)**

**{**

#A(tj[1]) + (B(tj[1])-A(tj[1]))\*(x/L)^2;

#1

0

**}**

**for(**i **in** 1**:**N**+**1**)**

**{**

x**[**i**]=** **(**i**-**1**)\***h

fij**[**i**]=**0

**}**

U **=** matrix**(**data**=NA**,nrow**=**n,ncol**=**N**+**1**)**

colnames**(**U**)=**x

row.names**(**U**)=** c**(**0**:(**n**-**1**))**

**for(**i **in** 0**:**N**+**1**)** #Считаем 0-й слой

**{**

U**[**1,i**]** **=** Ux0**(**x**[**i**])**

**}**

IterF **=** **function(**j**)** #Считаем остальные слои

**{**

alpha**[**1**]=**k1

beta**[**1**]=-**h**\***A**(**tj**[**j**])**

#print(beta[1])

**for(**i **in** 1**:**N**)** # Считаем очередные Ai, Bi , Ci

**{**

Ai**[**i**]** **=** **-** kU**(**U**[**j**-**1,i**])** **/** h**^**2

Bi**[**i**]** **=** **-** kU**(**U**[**j**-**1,i**])** **/** h**^**2

Ci**[**i**]** **=** CU**(**U**[**j**-**1,i**])** **/** tau **+** 2 **\*** kU**(**U**[**j**-**1,i**])** **/** h**^**2

**}**

**for(**i **in** 1**:(**N**-**1**))** #Считаем Fi

Fi**[**i**]** **=** CU**(**U**[**j**-**1,i**])\***U**[**j**-**1,i**+**1**]/**tau

#print(Fi)

**for(**i **in** 2**:**N**)** #Считаем альфа и бета коэффициенты

**{**

alpha**[**i**]=-**Bi**[**i**-**1**]/(**Ci**[**i**-**1**]+**Ai**[**i**-**1**]\***alpha**[**i**-**1**])**

beta**[**i**]=** **(**Fi**[**i**-**1**]-**Ai**[**i**-**1**]\***beta**[**i**-**1**])/(**Ci**[**i**-**1**]+**Ai**[**i**-**1**]\***alpha**[**i**-**1**])**

**}**

#print(beta)

#print(alpha)

#B(tj[j])

U**[**j,N**+**1**]=(**h**\***B**(**tj**[**j**])+**k2**\***beta**[**N**])/(**1**-**k2**\***alpha**[**N**])**

**for(**i **in** N**:**1**)** #Считаем Uj

U**[**j,i**]=** alpha**[**i**]** **\*** U**[**j,i**+**1**]** **+** beta**[**i**]**

#U[j,1]=k1\*U[j,2]-h\*A(tj[j])

return**(**U**[**j,**])**

**}**

**for(**j **in** 2**:**n**)**

U**[**j,**]** **=** IterF**(**j**)**

Uacc **=** matrix**(**data**=NA**,nrow**=**n,ncol**=**N**+**1**)**

colnames**(**Uacc**)=**x

row.names**(**Uacc**)=** c**(**0**:(**n**-**1**))**

summm**=function(**x,t**)**

**{**

summ**=**0

**for(**k **in** 1**:**10000**)**

**{**

#summ= summ + 1/(2\*k-1)^3\*sin((2\*k-1)\*pi\*x/L) \* exp(-(2\*k-1)^2\*pi^2\*a^2\*t/L^2)

#if(k==10) print(summ)

summ**=** summ **+** **(-**1**)^(**k**+**1**)/**k**^**2**\***exp**(-**k**^**2**\***pi**^**2**\***a**^**2**\***t**/**L**^**2**)\***cos**(**k**\***pi**\***x**/**L**)**

**}**

return**(**summ**)**

**}**

**for(**ix **in** 1**:(**N**+**1**))**

**for(**it **in** 1**:**n**)** **{**

#Uacc[it,ix]=A(tj[it]) + (B(tj[it]) - A(tj[it])) \* x[ix]/L - 8\*(B(tj[it])-A(tj[it]))/pi^3 \* summm(x[ix], tj[it])

#Uacc[it,ix]=B(tj[it])\*(a^2\*tj[it]/L+(3\*x[ix]^2-L^2)/(6\*L))+2\*L/pi^2\*summm(x[ix], tj[it])

**}**

library**(**animation**)**

oopt **=** ani.options**(**interval **=** 0.3**)**

ani.record**(**reset **=** **TRUE)**

**for(**j **in** 1**:**n**)**

**{**

plot**(**U**[**j,**]**, xaxt**=**"n", xlab **=** 'X values', ylab **=** 'U[x,t] values'**)**

lines**(**U**[**j,**]**,col**=**"red"**)**

#lines(Uacc[j,],col="blue")

axis**(**1, at **=** c**(**1**:(**N**+**1**))**, labels **=** x**)**

ani.pause**()**

ani.record**()**

**}**

print**(**"Численное решение"**)**

print**(**U**)**

#print("Точное решение")

#print(Uacc)

**while(TRUE)**

**{**

ani.replay**()**

ani.options**(**oopt, loop **=** 100**)**

**}**

# Список использованной литературы

1. Тихонов А.Н., Самарский А.А. - Уравнения математической физики , Москва, 1977