РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Теплопроводность и детерминированное горение. Этап 3

групповой проект

дисциплина: Математическое моделирование

Студенты: Тагиев Б.А.
Чекалова Л.Р.
Сергеев Т.С.
Саттарова В.В.
Прокошев Н.Е.
Тарусов А.С.
Группа: НФИбд-02-20

МОСКВА

2023 г.

Оглавление

1. Задачи

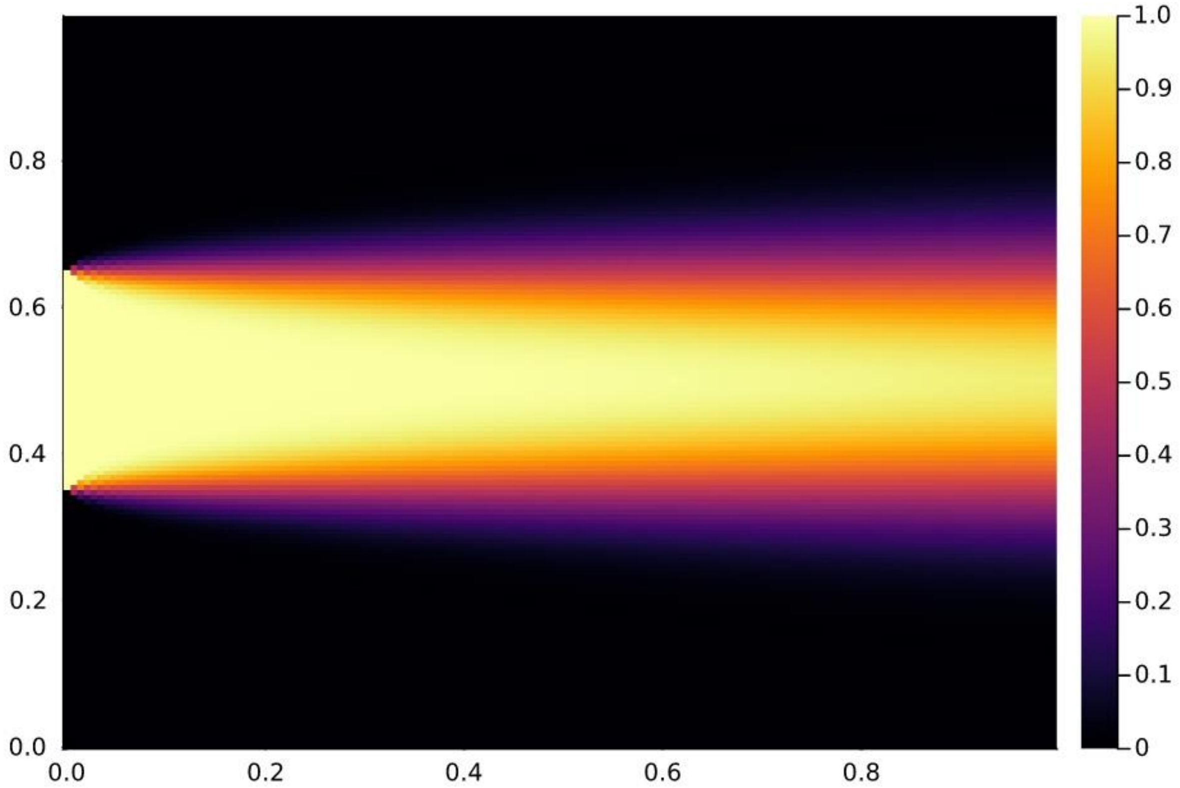
- Решение одномерного уравнения теплопроводности с использованием явной разностной схемы
- Решение одномерного уравнения теплопроводности с использованием неявной разностной схемы Кранка-Николсона
- Визуализация теоретического решения уравнения теплопроводности на неограниченной прямой в случае, если в начальный момент в точке $x = x_0$ мгновенно выделяется количество тепла Q_0
- Добавление химической реакции в уравнение теплопроводности
- Построение графика скорости горения

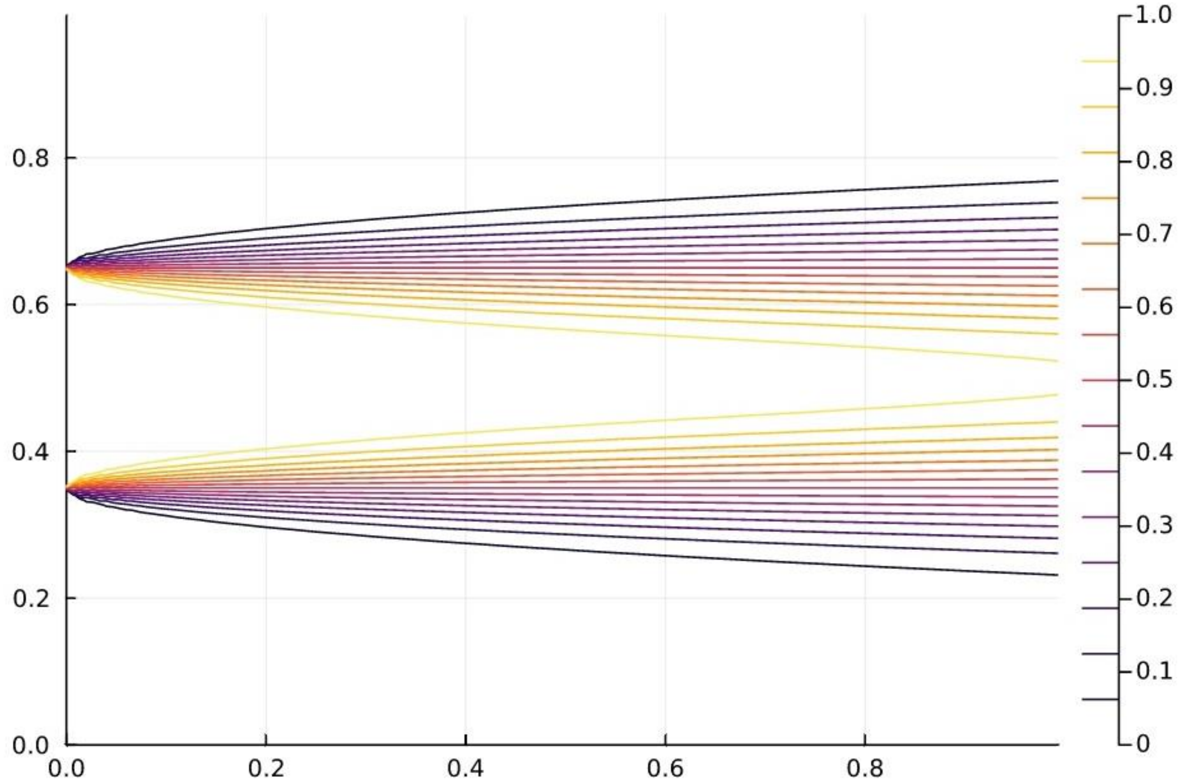
2. Явная разностная схема

Пишем программу на Julia, которая решит уравнение теплопроводности с помощью явной разностной схемы. Воспользуемся дельта-функцией, чтобы задать начальные условия, а в цикле будем применять явную разностную схему для нахождения температуры на новом слое по времени на основе температуры в узлах текущего временного слоя.

```
using Plots
using DifferentialEquations

δ(x) = x == 0 ? 0.5 : x > 0 ? 1 : 0 # дельта-функция с использованием тернарного оператора
startcond = x-> δ(x - 0.35) - δ(x - 0.65) # начальное условие
bordrcond = x-> 0. # условие на границе

Nx = 150
Nt = 150
tlmt = 1.0

dx = 1 / Nx
dt = tlmt / Nt

x = [i for i in range(0, length = Nx, step = dx)] # один из способов задать массив с помощью цикла
t = [i for i in range(0, length = Nt, step = dt)]

U = zeros(Nx, Nt)
U[: , 1] = startcond.(x)
U[1 , :] = U[Nt, :] = bordrcond.(t)

for j = 1:Nt-1, i = 2:Nx-1
    χ = 0.003
    h = dx
    U[i, j + 1] = U[i, j] + (χ*dt / h ^ 2)*(U[i - 1, j] - 2 * U[i, j] + U[i + 1, j])
end

p11 = plot(heatmap(t, x, U))
p12 = plot(t, x, U)

savefig(p11, "out/project/task_1_1.png")
savefig(p12, "out/project/task_1_2.png")
```

Выводим результаты в виде тепловой карты и графика распространения температуры.





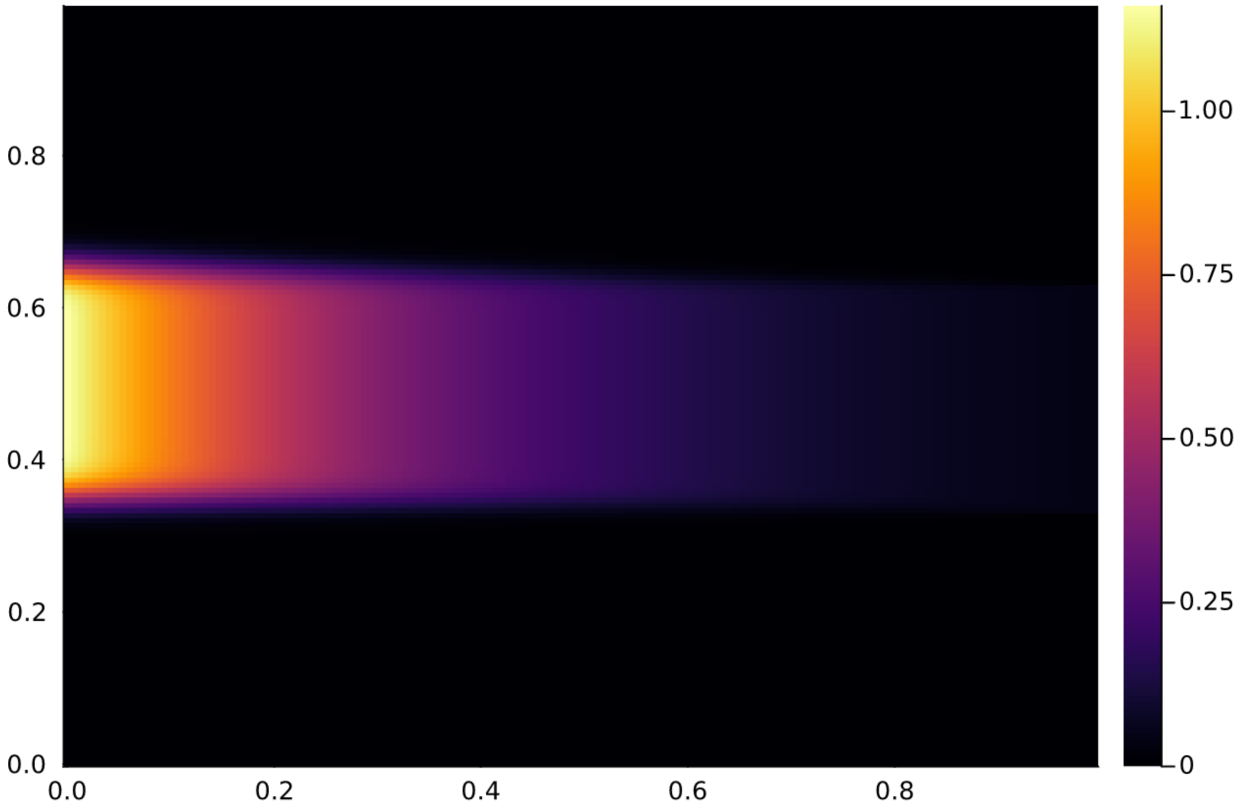
Мы видим, что температура сначала высокая, а потом постепенно снижается.

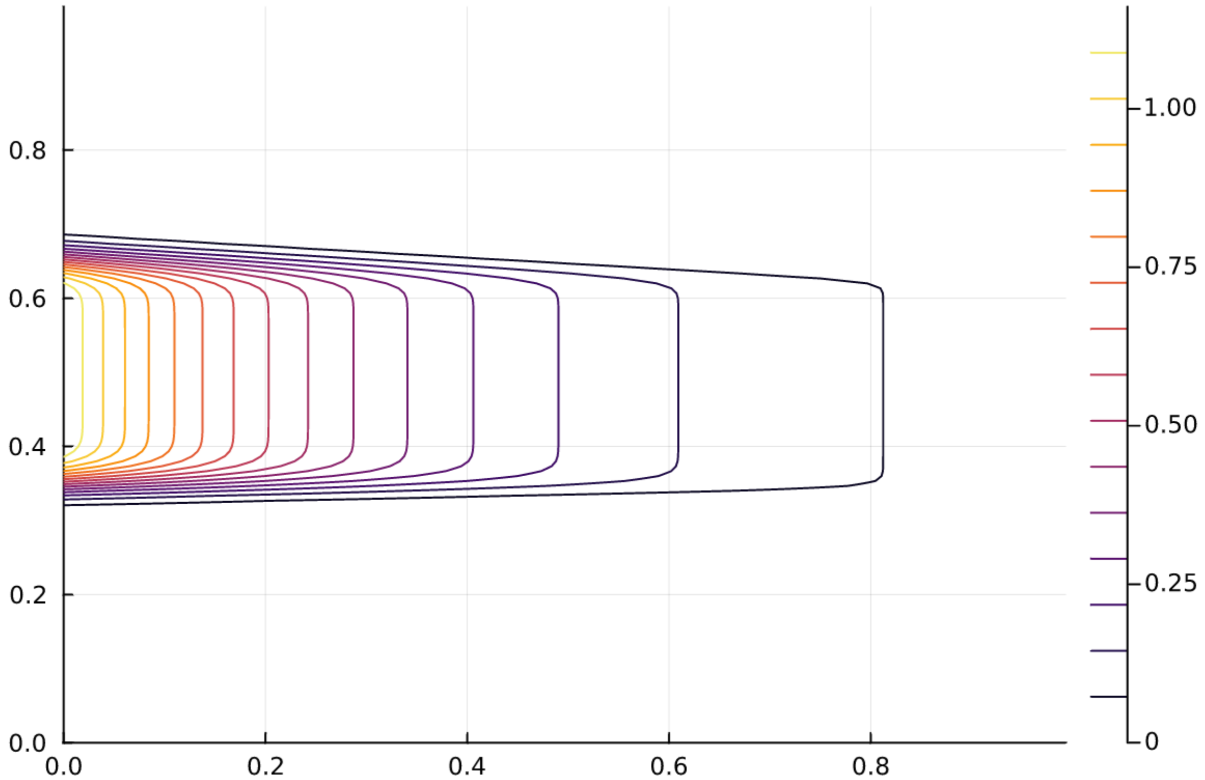
3. Неявная разностная схема

Далее модифицируем программу, чтобы решать уравнение с помощью неявной разностной схемы Кранка-Николсона, то есть температура на новом временном слое вычисляется как на основе температуры в узлах нового временного слоя, так и на основе температуры в узлах текущего временного слоя.

```
using Plots
using DifferentialEquations

δ(x) = x == 0 ? 0.5 : x > 0 ? 1 : 0 # дельта-функция с использованием тернарного оператора
startcond = x-> δ(x - 0.35) - δ(x - 0.65) # начальное условие
bordrcond = x-> 0. # условие на границе

Nx = 150
Nt = 150
tlmt = 1.0

dx = 1 / Nx
dt = tlmt / Nt

x = [i for i in range(0, length = Nx, step = dx)] # один из способов задать массив с помощью цикла
t = [i for i in range(0, length = Nt, step = dt)]

U = zeros(Nx, Nt)
U[: , 1] = startcond.(x)
U[1 , :] = U[Nt, :] = bordrcond.(t)

for j = 1:Nt - 1, i = 2:Nx - 1
    χ = 0.0001
    h = dx
    U[i, j + 1] = U[i, j] + (χ*dt / h ^ 2 / 2) * (U[i - 1, j + 1] - 2 * U[i, j + 1] + U[i + 1, j + 1])
    U[i, j + 1] += (χ*dt / h ^ 2 / 2) * (U[i - 1, j] - 2 * U[i, j] + U[i + 1, j])
end

print(U)
p = plot(heatmap(t, x, U))

savefig(p, "out/project/task_2.png")
```

Получаем тепловую карту и график распределения температуры.





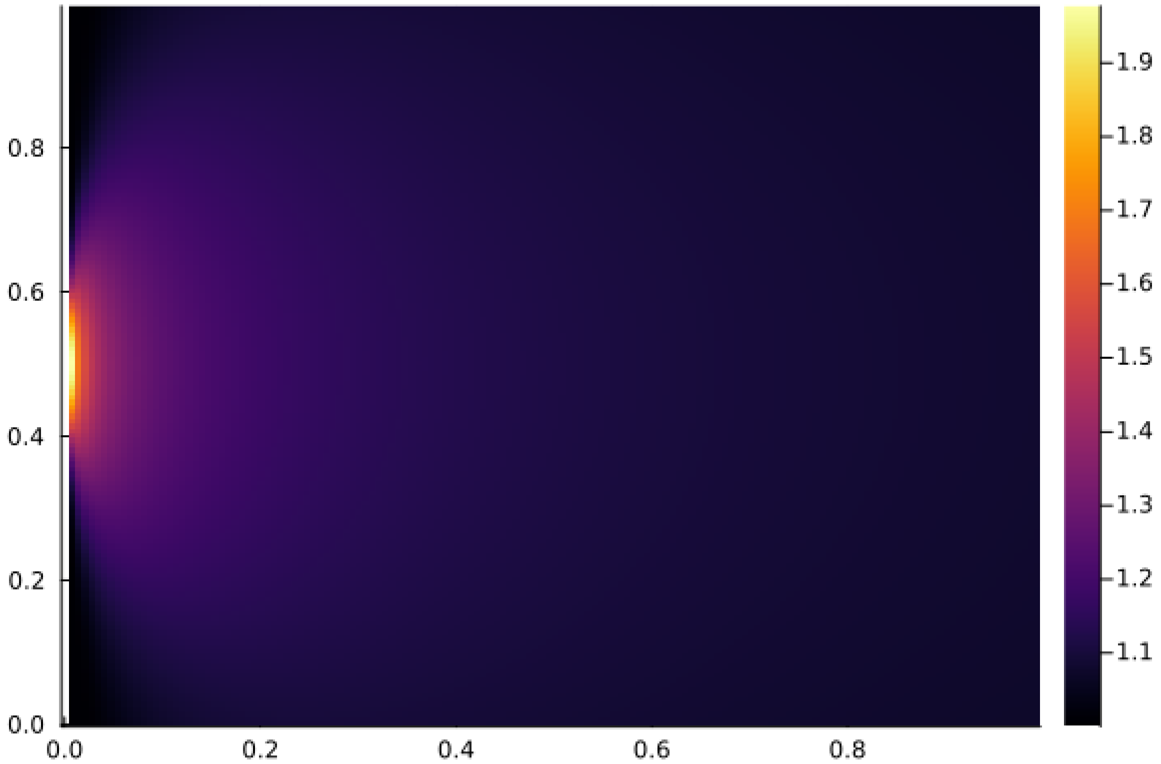
Здесь мы наблюдаем постепенное понижение температуры так же, как и в явной схеме.

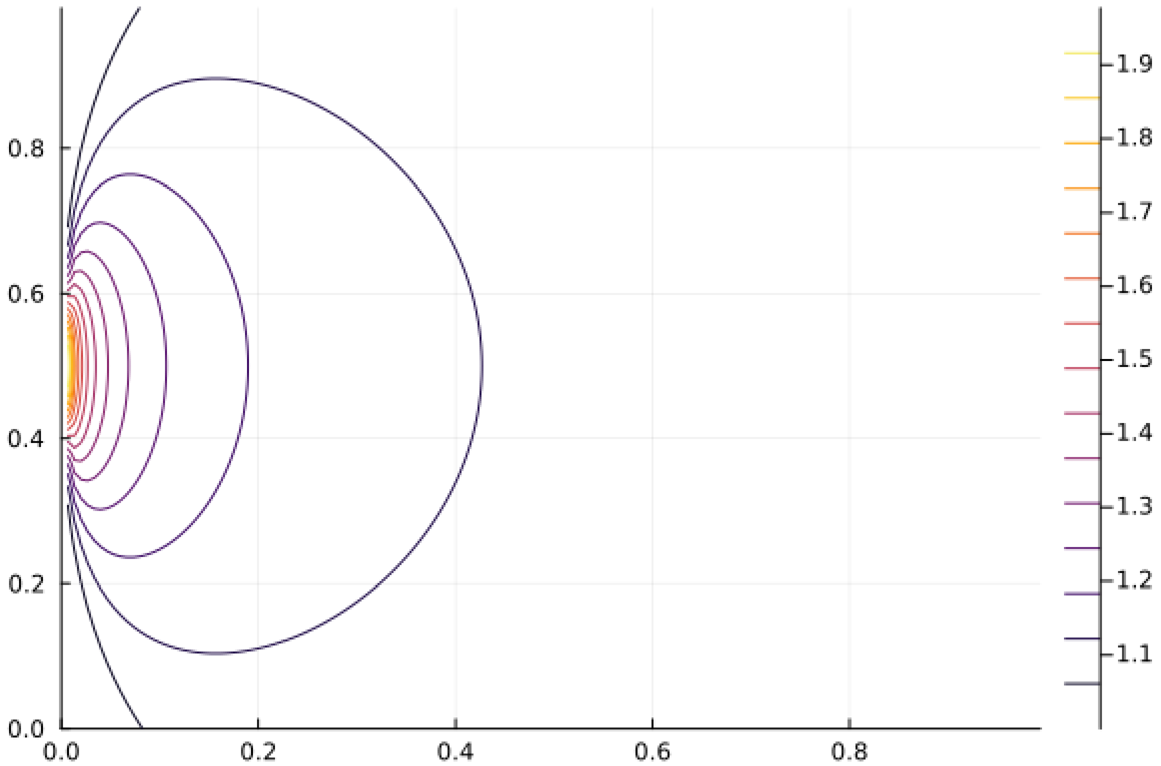
4. Теоретическое решение

Напишем программу, описывающую теоретическое решение уравнения теплопроводности на неограниченной прямой для случая, когда в начальный момент в точке $x = x_0$ мгновенно выделяется количество тепла Q_0. Такое решение имеет вид $T(x,t) = T_0 + \frac{Q_0}{Q}\frac{1}{\sqrt{4\pi\chi t}}e^{-(x-x_0)^2/4\chi t}$.

```
using Plots
using DifferentialEquations

Nx = 150
Nt = 150
tlmt = 1.0

dx = 1 / Nx
dt = tlmt / Nt

x = [i for i in range(0, length = Nx, step = dx)] # один из способов задать массив с помощью цикла
t = [i for i in range(0, length = Nt, step = dt)]

Q = 5
Q0 = 1
x0 = 0.5
χ = 0.5
T = zeros(Nx, Nt)
for x_i = 1:length(x)
    for t_i = 1:length(t)
        T[x_i, t_i] = 1 + Q0/Q * 1/sqrt(4*pi*χ*t[t_i])*exp(-(x[x_i]-x0)^2/(4*χ*t[t_i]))
    end
end

p31 = plot(heatmap(t, x, T))
p32 = plot(t, x, T)

savefig(p31, "out/project/task_3_1.png")
savefig(p32, "out/project/task_3_2.png")
```

Результат также представляем в виде тепловой карты и графика распределения температуры.





Из-за того, что в точке x_0 мгновенно выделяется некоторое количество тепла, в начальный момент времени мы наблюдаем высокие температуры, которые затем начинают стремительно снижаться.

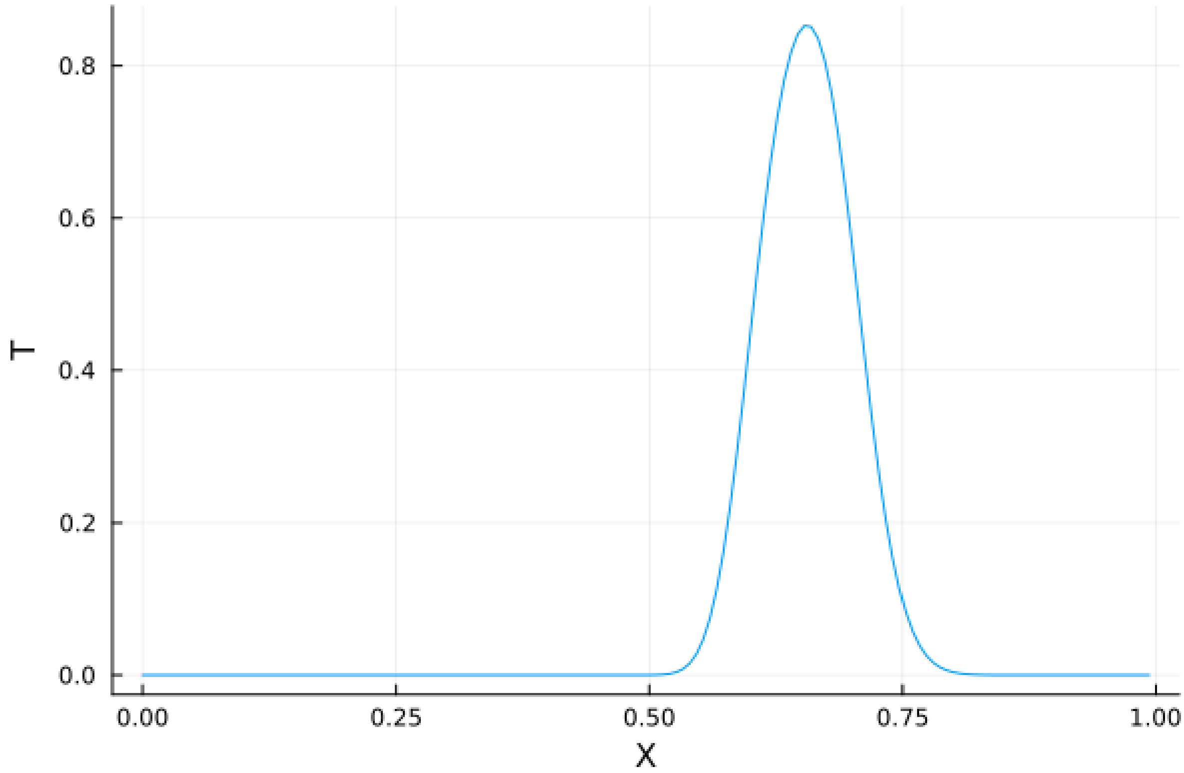
5. Химическая реакция

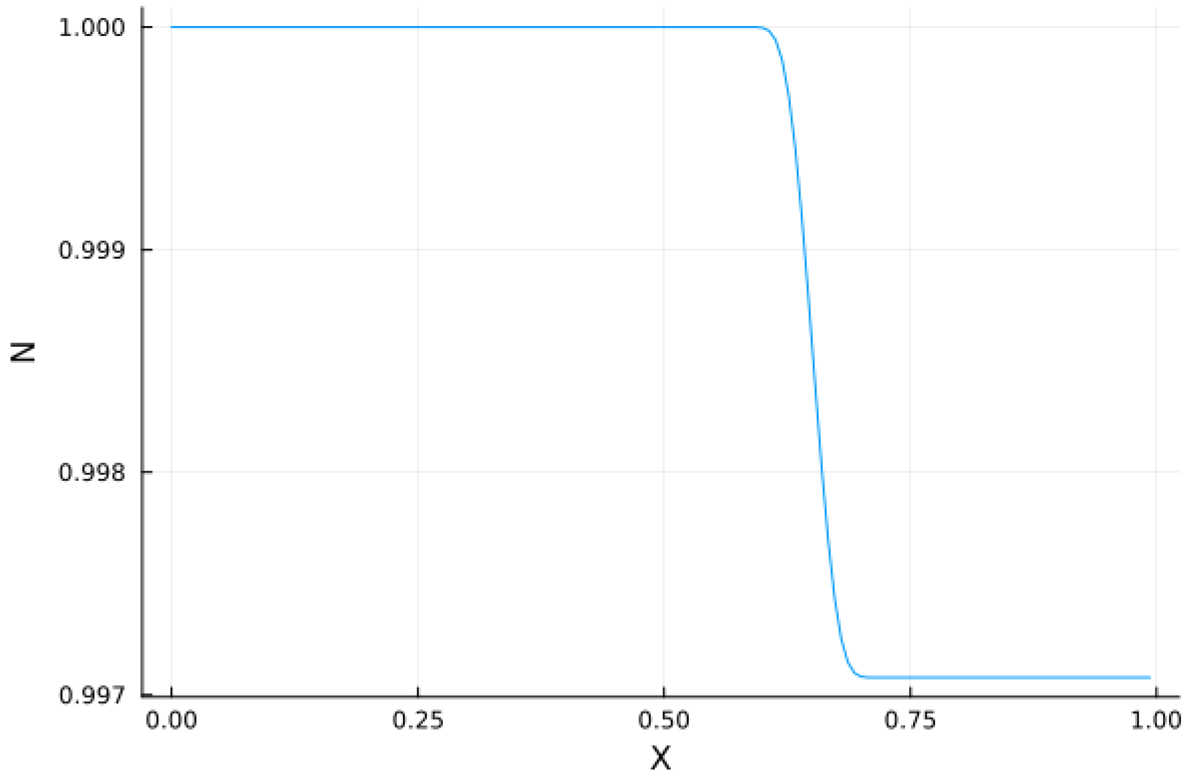
Напишем программу, решающую уравнение теплопроводности с химической реакцией. Для этого мы добавим изменение температуры за счет энерговыделения в химических реакциях за шаг по времени (ΔN_i, которое выводится из закона Аррениуса для реакции первого порядка).

```
using Plots
using DifferentialEquations

δ(x) = x == 0 ? 0.5 : x > 0 ? 1 : 0 # дельта-функция с использованием тернарного оператора
startcond = x-> δ(x - 0.45) - δ(x - 0.55) # начальное условие
bordrcond = x-> 0. # условие на границе

Nx = 150
Nt = 150
tlmt = 2.0 #20

dx = 1 / Nx
dt = tlmt / Nt

x = [i for i in range(0, length = Nx, step = dx)] # один из способов задать массив с помощью цикла
t = [i for i in range(0, length = Nt, step = dt)]

N = zeros(150)
U = zeros(Nx, Nt)
U[: , 1] = startcond.(x)
U[1 , :] = U[Nt, :] = bordrcond.(t)
```

```
χ = 0.0005
h = dx
tau = 0.1
E = 5
N[1] = 1
U_ = zeros(150)
for i = 2:149
    for j = 1:149
    dN = -N[i - 1] / tau*dt*exp(-E / U[i, j])
    U[i, j + 1] = U[i, j] + (χ*dt / h ^ 2 / 2) * (U[i - 1, j + 1] - 2 * U[i, j + 1] + U[i + 1, j + 1]) + dN
    U[i, j + 1] += (χ*dt / h ^ 2 / 2) * (U[i - 1, j] - 2 * U[i, j] + U[i + 1, j])
    if j == 145
        N[i] = N[i - 1] + dN # доля прореагировавшего вещества
        U_[i] = U[i, j + 1]
    end
    end
end
N[150] = N[149]
p41 = plot(x, U_, xlabel = "X", ylabel = "T", legend=false)
p42 = plot(x, N, xlabel = "X", ylabel = "N", legend=false)
p43 = plot(heatmap(t, x, U))

savefig(p41, "out/project/task_4_U.png")
savefig(p42, "out/project/task_4_N.png")
```

Получим график изменения температуры и график изменения количества вещества.





Мы видим, как возрастает температура, одновременно с этим происходит уменьшение количества вещества. Потом температура снижается до нуля и вещество прекращает горение, его количество прекращает уменьшаться.

В уравнении присутствует энергия активации, от нее зависит режим горения. Если при $T_0 \ll 1$ энергия активации меньше 6.56, то происходит стационарное горение, если больше – пульсирующее.

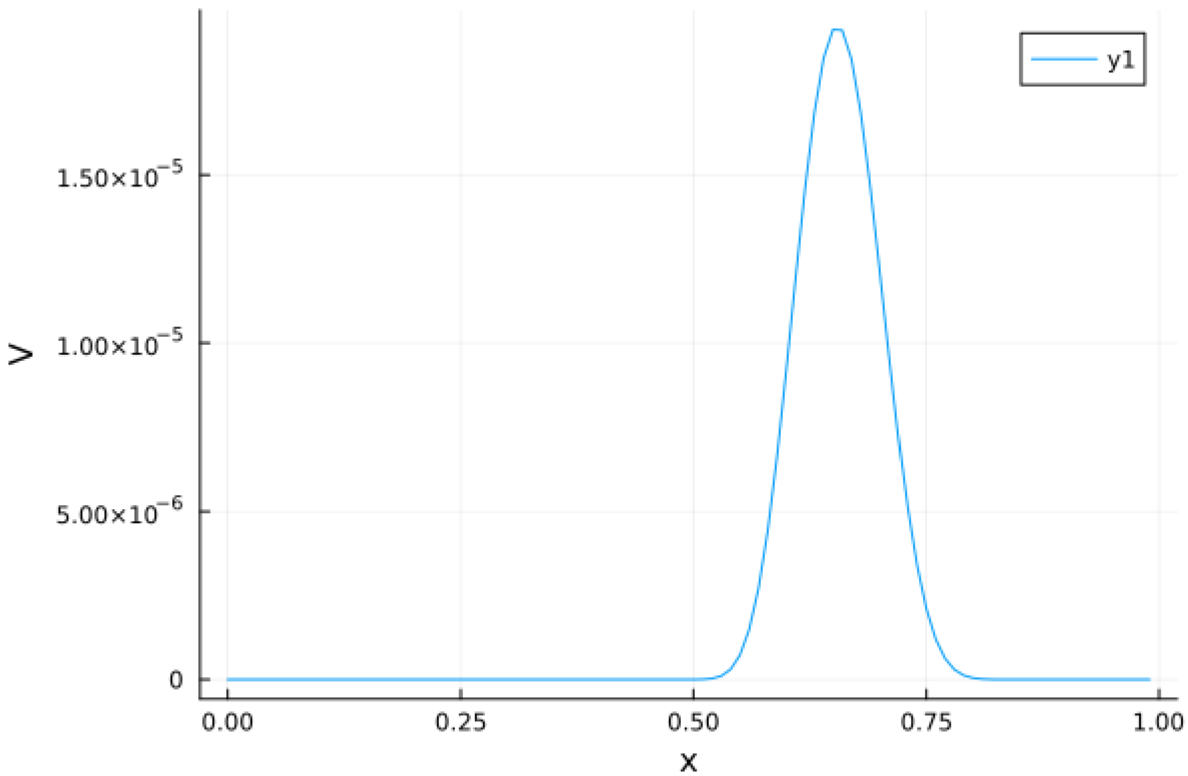
6. Скорость горения

Напишем программу, чтобы построить график скорости горения от координаты фронта. Найдем координату, где $N = 0.5$. Для этого будем пользоваться интерполяцией.

```
using Plots
using DifferentialEquations

δ(x) = x == 0 ? 0.5 : x > 0 ? 1 : 0 # дельта-функция с использованием тернарного оператора

Nx = 100
Nt = 100
tlmt = 1.0

dx = 1 / Nx
dt = tlmt / Nt

x = [i for i in range(0, length = Nx, step = dx)] # один из способов задать массив с помощью цикла
t = [i for i in range(0, length = Nt, step = dt)]

startcond = x-> δ(x - 0.0) - δ(x - 0.2) # начальное условие
bordrcond = x-> 0. # условие на границе

U = zeros(Nx, Nt)
DNN = zeros(Nx)
U[: , 1] = startcond.(x)
U[1 , :] = bordrcond.(t)

N_ = zeros(Nx, Nt)
vt = zeros(Nt)
N = [i for i in range(1, stop=0, step = -dx)]

for i=1:Nx
N_[1, i] = 1
end

tau = 1.8
E = 10.8
```

```
tau = 1.8
E = 10.8

for i = 2:Nx-1, j = 1:Nt-1
    χ = 0.005
    h = dx
    dN = -N[i]/tau*dt*exp(-E/U[i, j])
    U[i, j + 1] = U[i, j] + (χ*dt / h ^ 2 / 2) * (U[i - 1, j + 1] - 2 * U[i, j + 1] + U[i + 1, j + 1])
    U[i, j + 1] += (χ*dt / h ^ 2 / 2) * (U[i - 1, j] - 2 * U[i, j] + U[i + 1, j]) + dN
    N_[i, j + 1] = N[i] + dN
    if i == 55
    vt[j] = -dN / dt
    end
end

I = 0
XX = 0
for i = 1:Nx-1
    if round((N[i]+N[i+1])/2, digits=1) == 0.5
        global XX = x[i]
        global I = i
    end
end

for j = 1:Nt-1
    dN = -N[I]/tau*dt*exp(-E/U[I, j])
    vt[j] = -dN / dt
end

p51 = plot(x, vt, xlabel="x", ylabel="V")

savefig(p51, "temp_1.png")
```

Получаем график зависимости скорости горения от координаты фронта.



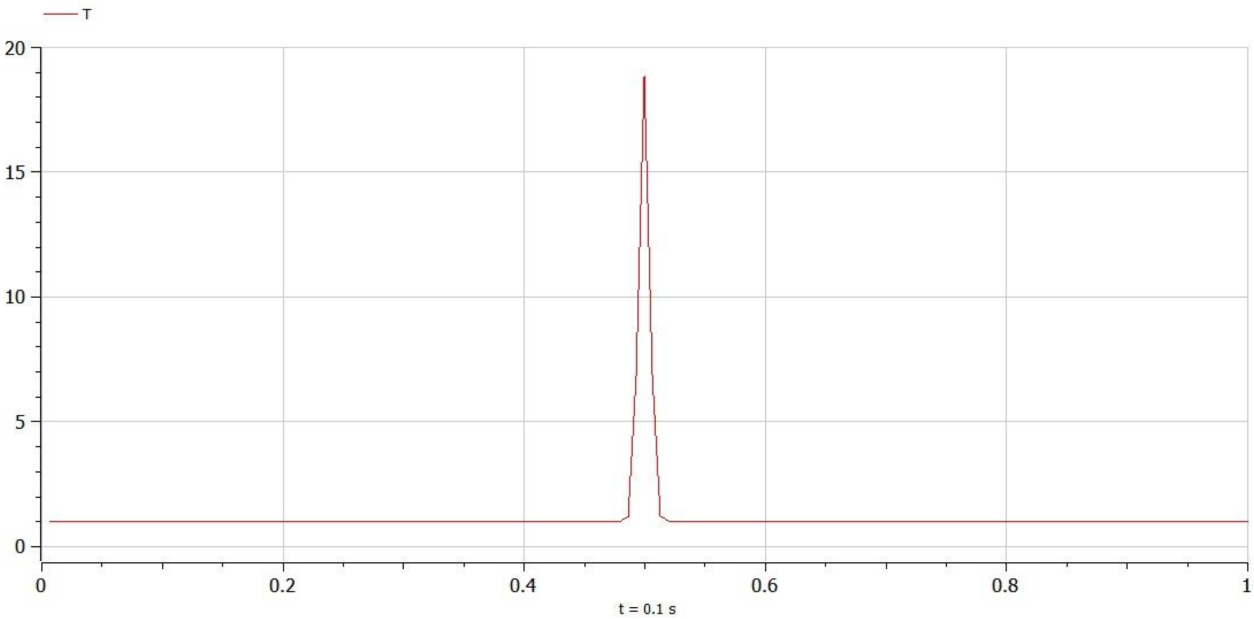
Мы видим, как скорость горения резко возрастает, а затем так же стремительно падает.

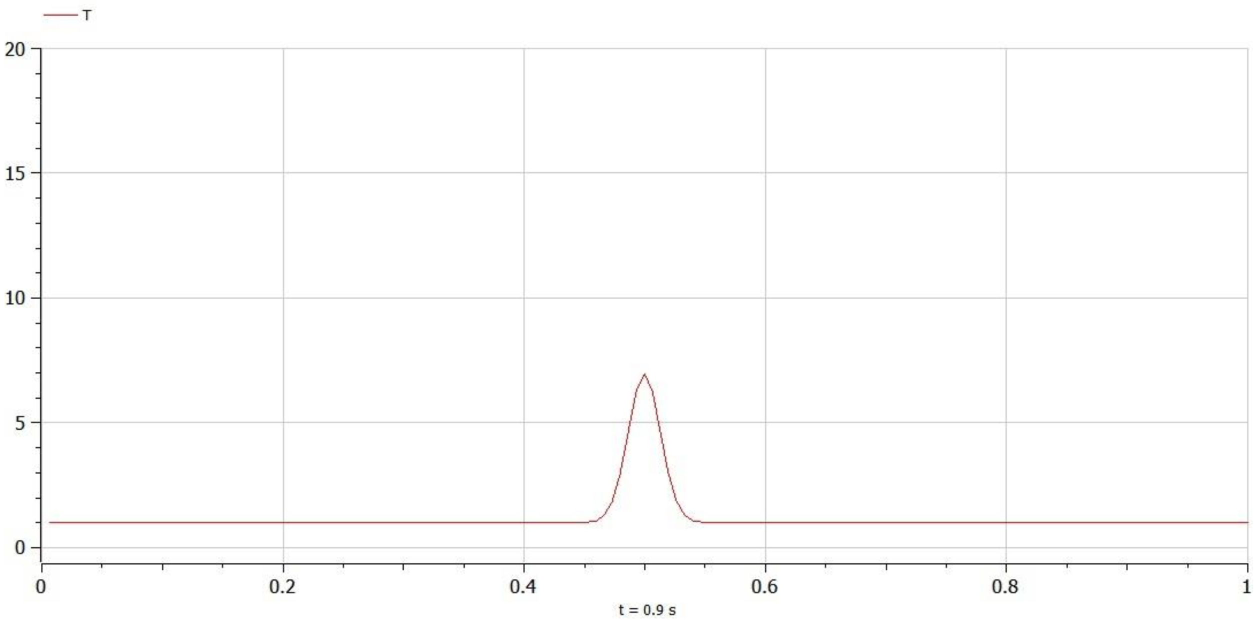
7. Реализация на OpenModelica

Запрограммируем теоретическое решение уравнения теплопроводности на неограниченной прямой для случая, когда в начальный момент в точке $x = x_0$ мгновенно выделяется количество тепла Q_0, на OpenModelica.

```
model task
Real x[150] = {i/150 for i in 1:150};
Real x0 = 0.5;
Real Chi = 0.0001;
Real pi = 3.1416;
Real Q0 = 1;
Real Q = 5;
Real T[150];
equation
algorithm
  for i in 1:size(x, 1) loop
    if time > 0 then
    T[i] := 1 + Q0/Q*1/sqrt(4*pi*Chi*time)*Modelica.Math.exp(-(x[i]-x0)^2/(4*Chi*time));
    end if;
  end for;
end task;
```

Результат представим в виде двух графиков: температуры при $t = 0.1$ и температуры при $t = 0.9$.





Мы видим, что в момент времени, близкий к начальному, температура высокая, а далее она начинает падать.

8. Вывод

Мы выполнили все поставленные задачи, решили уравнение теплопроводности разными способами и визуализировали полученные результаты, отдельно рассмотрев изменения некоторых параметров, таких как температура, количество вещества, скорость.

9. Список литературы

1. Медведев Д. А., Куперштох А. Л., Прууэл Э. Р., Сатонкина Н. П., Карпов Д. И. Моделирование физических процессов и явлений на ПК: Учеб. пособие / Новосибирск: Новосиб. гос. ун-т., 2010. / ISBN 978-5-94356-933-3
2. Хакимзянов Г. С., Черный С. Г. Методы вычислений: В 4 ч.: Учеб. пособие / Новосиб. гос. ун-т. Новосибирск, 2007. Ч. 3: Численные методы решения задач для уравнений параболического и эллиптического типов. 160 с. / ISBN 978-5-94356-612-7