Теплопроводность и детерминированное горение. Этап 3

Групповой проект

Студенты:

Тагиев Б.А. *Чекалова Л.Р.* *Сергеев Т.С.*

Саттарова В.В. *Прокошев Н.Е.* *Тарусов А.С.*

Группа: НФИбд-02-20

2023

Задачи

- Решить одномерное уравнение теплопроводности с помощью явной разностной схемы
- Решить одномерное уравнение теплопроводности с помощью неявной разностной схемы
- Визуализировать теоретическое решение уравнения: $T(x,t) = T_0 + \frac{Q_0}{Q}\frac{1}{\sqrt{4\pi\chi t}}e^{-(x-x_0)^2/4\chi t}$
- Добавить химическую реакцию в уравнение теплопроводности
- Построить график скорости горения

Явная разностная схема:

```
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")
```

Рис.1: Программа на Julia

Явная разностная схема: визуализация

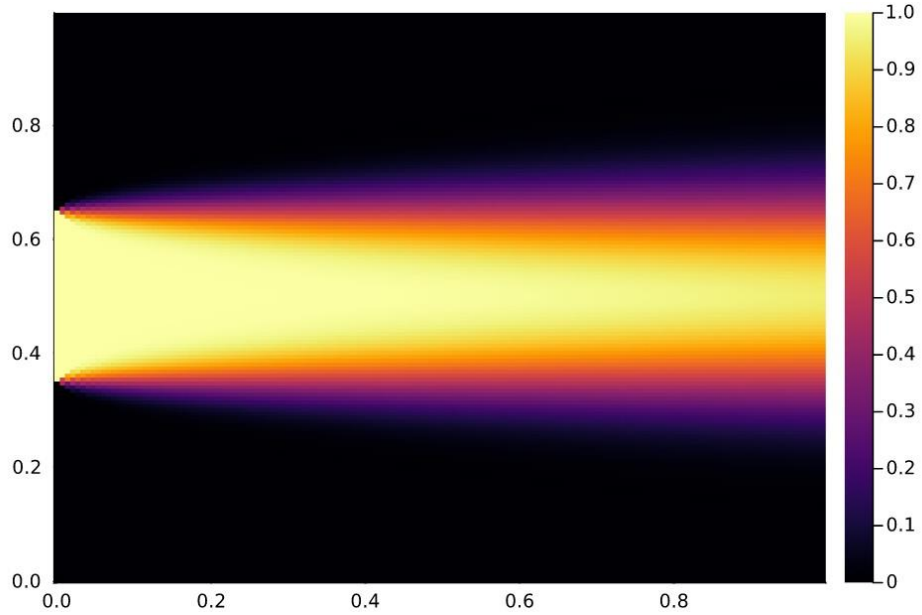

Рис.2: Тепловая карта распространения температуры

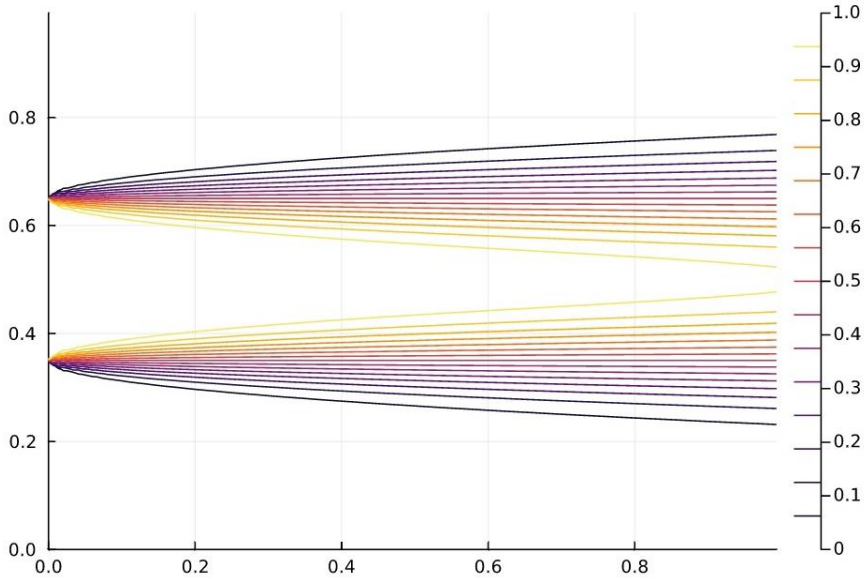

Рис.3: График распространения температуры

Неявная разностная схема: программная реализация

```
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")
```

Рис.4: Программа на Julia

Неявная разностная схема: визуализация

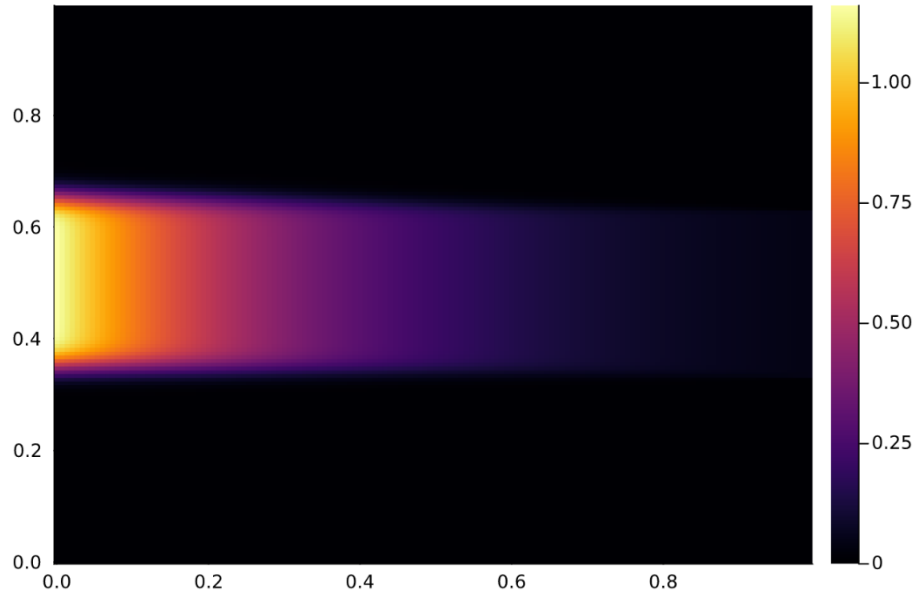

Рис.5: Тепловая карта распространения температуры

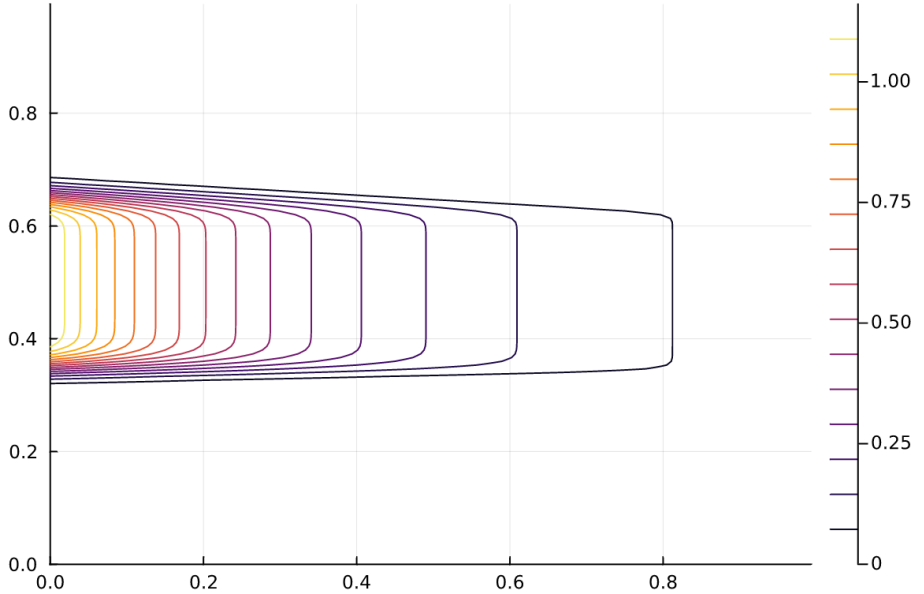

Рис.6: График распространения температуры

Теоретическое решение: программная реализация (Julia)

```julia
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")
```

Рис.7: Программа на Julia

Теоретическое решение: визуализация (Julia)

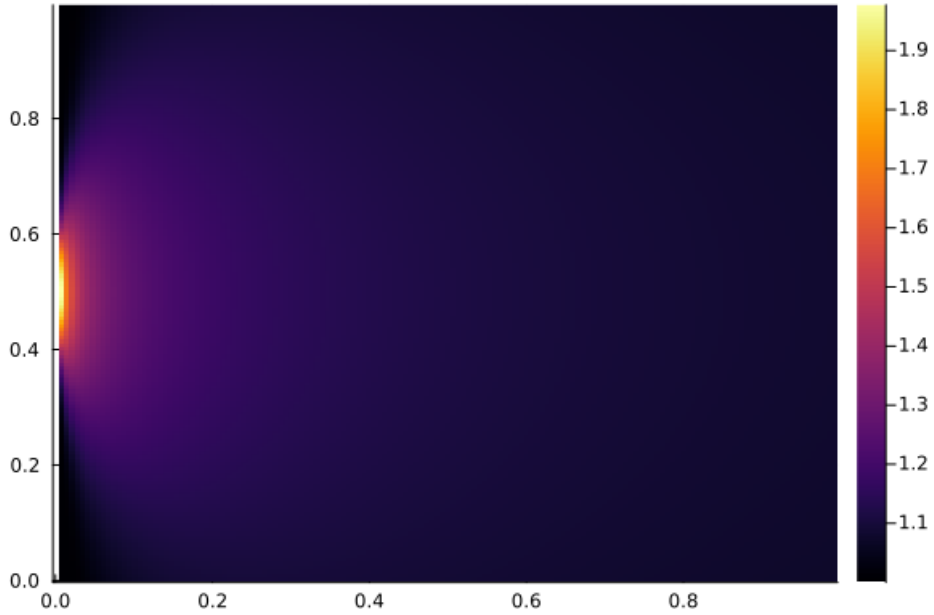

Рис.8: Тепловая карта распространения температуры

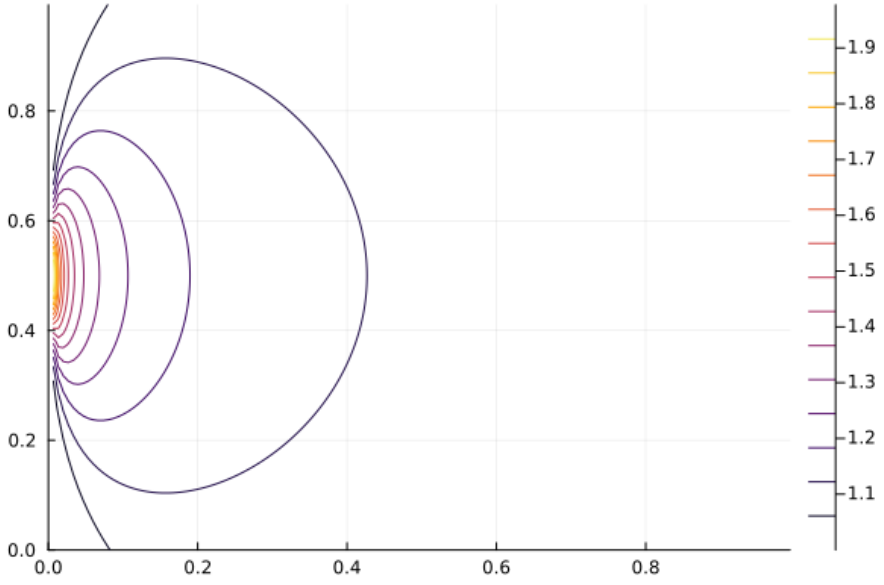

Рис.9: График распространения температуры

Химическая реакция: программная реализация

```
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")
```

Рис.10-Рис.11: Программа на Julia

Химическая реакция: визуализация

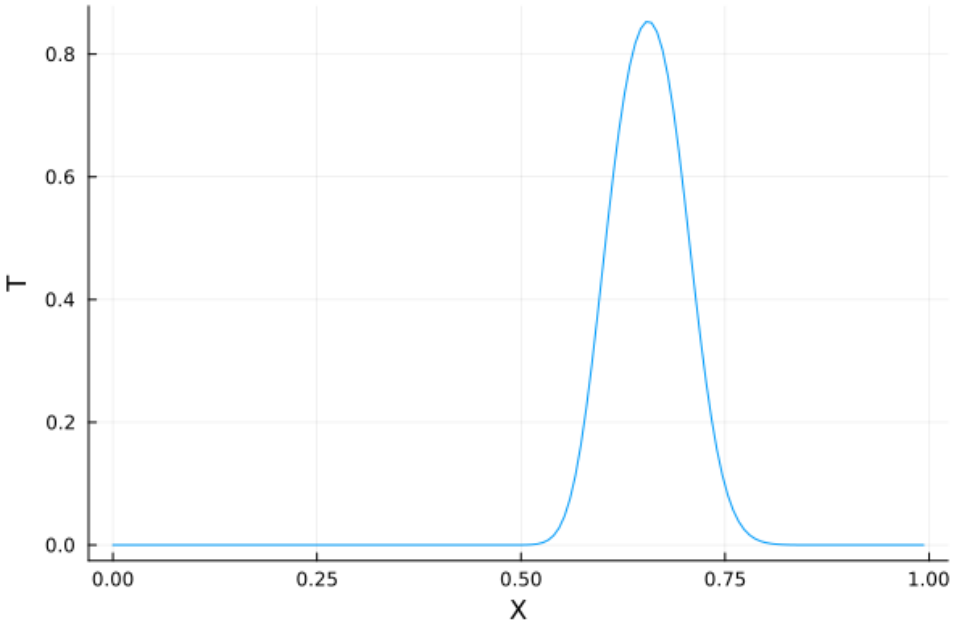

Рис.12: График изменения температуры

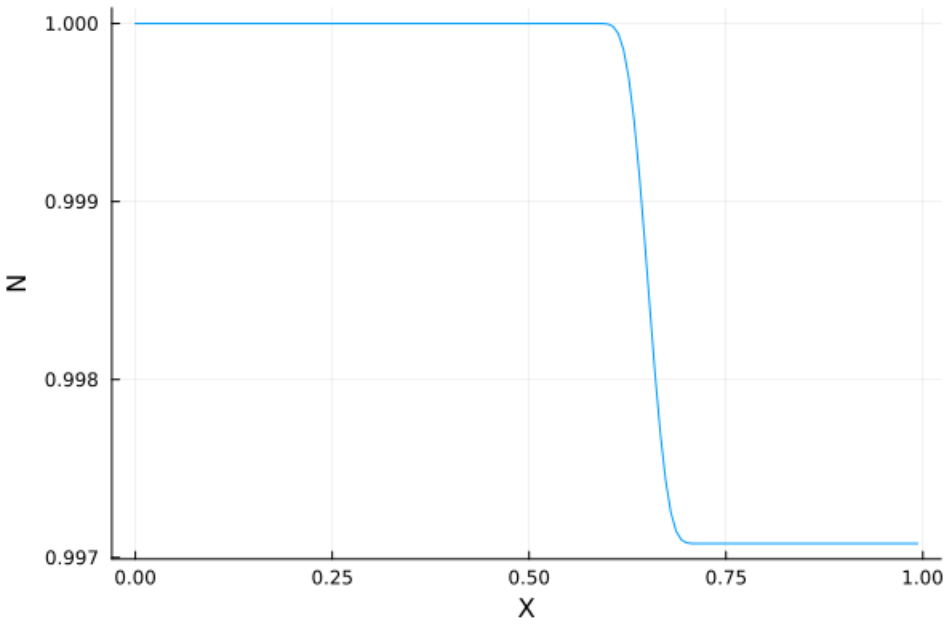

Рис.13: График изменения количества вещества

Скорость горения: программная реализация

```julia
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
```

```julia
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")
```

Рис.14-Рис.15: Программа на Julia

Скорость горения: визуализация

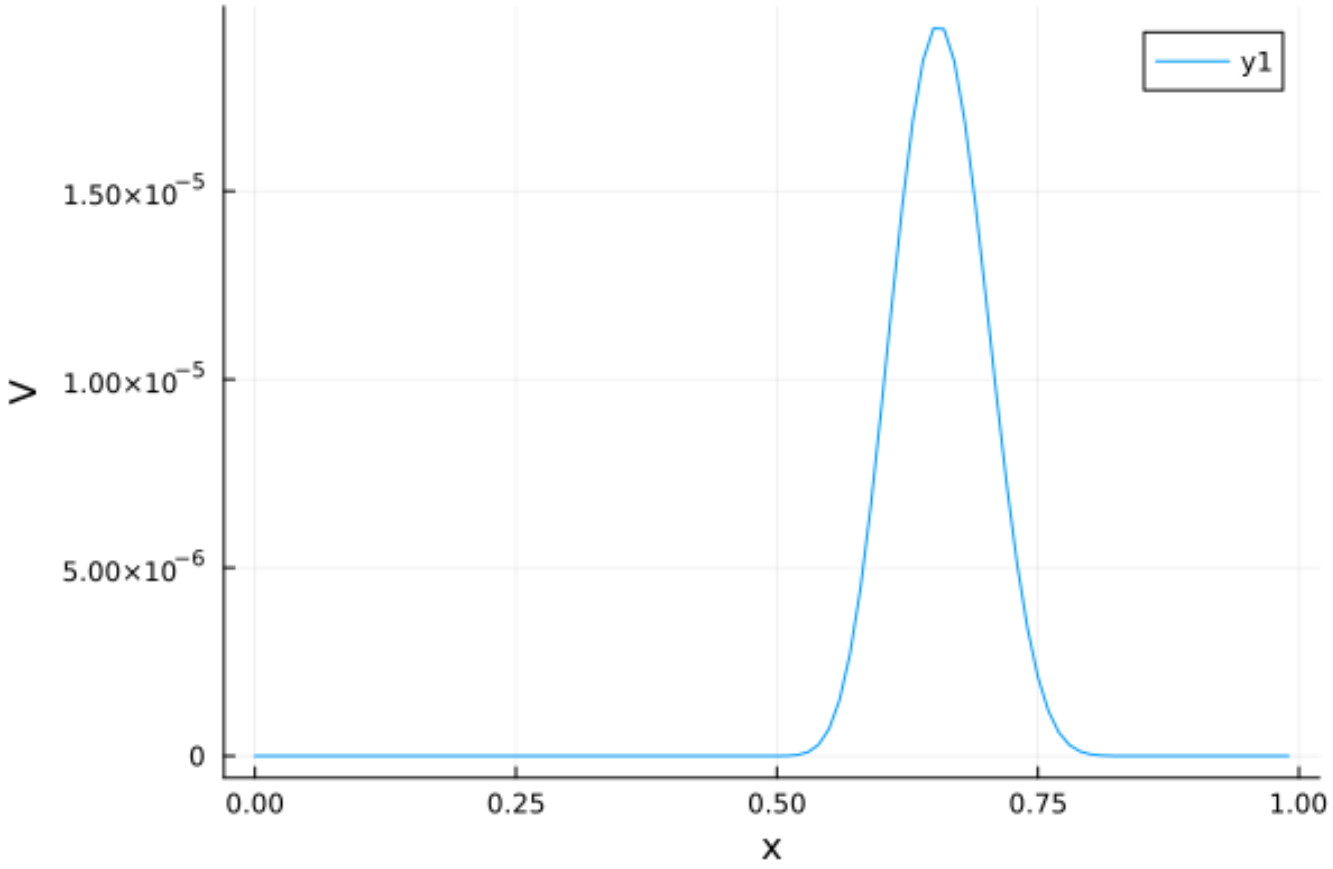

Рис.16: График изменения скорости горения

Теоретическое решение: программная реализация (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;
```

Рис.17: Программа на OpenModelica

Теоретическое решение: визуализация (OpenModelica)

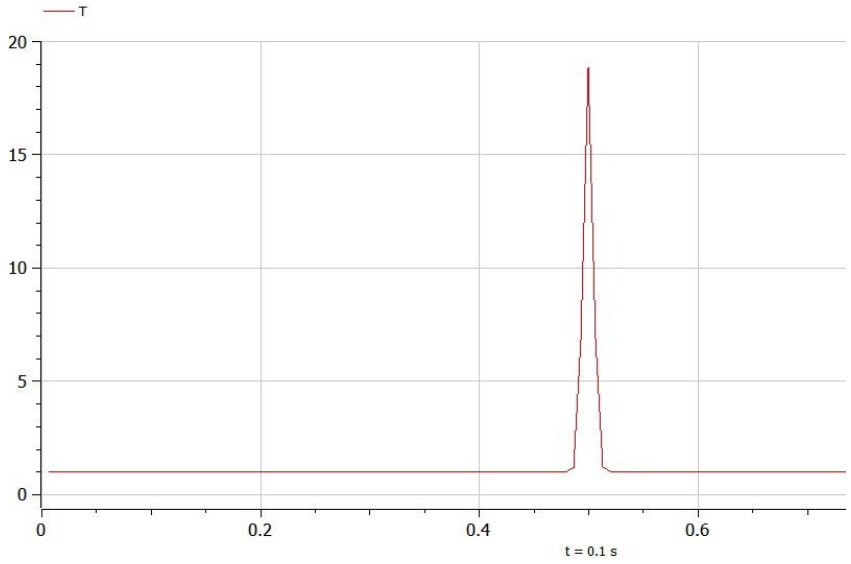

Рис.18: График температуры при $t = 0.1$

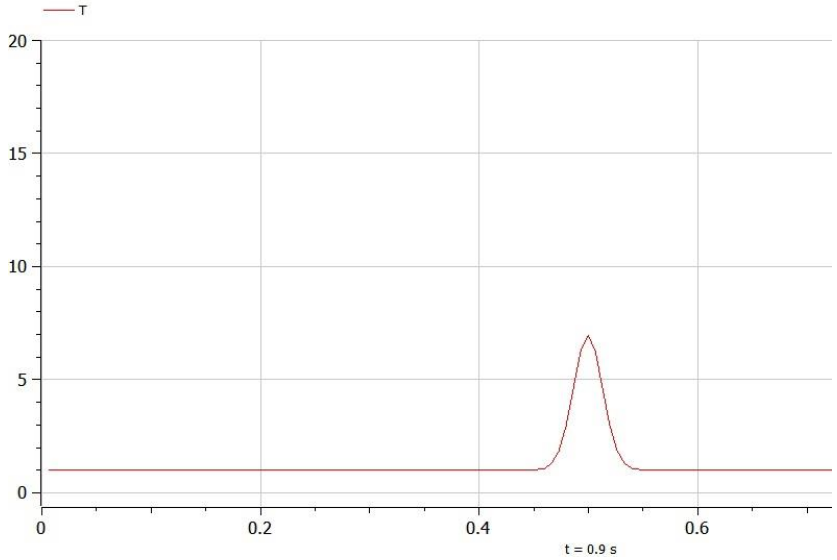

Рис.19: График температуры при $t = 0.9$