Gliwice, 13.01.2025

# Mathematical modeling and computer simulations in theory and practice

Documentation of laboratory task no 8

Title: **Distribution of temperature in a bar**

*Author: Aleksandra Stachecka*
Field of studies: Informatics sem.V

## 1 Project Objective

The goal of the project is to solve and visualize the heat equation, which describes how temperature changes in one-dimensional bar over time. The project simulates different scenarios with various starting temperatures and boundary conditions.

## 2 Description

### 2.1 Program

The program consists of a function `heat` taking the following inputs:

- **xstart_**: The starting point of the spatial domain ($x = a$), where the bar begins. For example 0 means the bar starts at position $x = 0$.
- **xend_**: The end point of the spatial domain ($x = b$), where the bar ends.
- **tmax_**: The maximum time ($t^*$) for which the simulation runs.
- **init_**: The initial temperature distribution $u(x, 0)$ across the bar at time $t = 0$.
- **boundLeft_**: The time-dependent boundary condition $u(a, t)$ at the left end of the bar.
- **boundRight_**: The time-dependent boundary condition $u(b, t)$ at the right end of the bar.

The first step is to define the partial differential equation that models the heat distribution in the bar. This equation describes how the temperature at each point in space $x$ evolves over time $t$. The heat equation is written as:

(1)

where $u(x, t)$ represents the temperature at position of $x$ and time $t$, and $k$ is a constant that depends on the material properties of the bar, in our case $k$ is always 1.

Then we specify the initial condition, which defines the temperature distribution at the start of the process (at $t = 0$). *init* represents a function or expression that provides the initial temperature at each position $x$ along the bar.

The next step is to define the boundary conditions. These conditions specify the temperature at the left and right ends of the bar at any time $t$. At the end we need to specify the spatial and time domains over which the solution will be computed. The result is stored in the variable *sol*, which contains the numerical solution for the temperature distribution $u(x, t)$ at all points in space and over time within the defined domains.

```
sol = NDSolve[
   {D[u[x, t], t] == D[u[x, t], {x, 2}],
    u[x, 0] == init,
    u[xa, t] == boundLeft, u[xb, t] == boundRight},
   u, {x, xa, xb}, {t, 0, tb}];
```

After that, we create a table of frames that display two visualizations for the temperature profile at various time points. These visualizations consist of:

1. A **plot of the temperature distribution** as a function of space $x$ at a particular time $t$.

2. A **density plot** that shows the temperature profile with color gradients.

The variable $tVal$ iterates over the time interval from 0 to $tb$ with a step size of $tb/100$ meaning the table will contain 101 frames (including $t = 0$ and $t = tb$).

Each frame contains two elements arranged in a grid: a plot of the temperature distribution $u(x, t)$ at the given time $tVal$ and a density plot that uses color gradients to show the temperature distribution.

```
frames = Table[
   Grid[{
     {Plot[Evaluate[u[x, t] /. sol /. t -> tVal],
       {x, xa, xb},
       PlotRange -> All,
       PlotStyle -> {Thick, Blue},
       AxesLabel -> {"x", "Temp."},
       PlotLabel ->
        StringJoin["Temperature Profile at t = ", ToString[N[tVal]]]
       ],
      DensityPlot[Evaluate[u[x, t] /. sol /. t -> tVal],
       {x, xa, xb}, {y, 0, 0.1},
       PlotRange -> All,
       AspectRatio -> 0.1,
       Frame -> False,
       ColorFunction -> "TemperatureMap"]}
     }],
   {tVal, 0, tb, tb/100}];
```

Let's have a closer look at the `Plot` part.
`Evaluate` replaces the solution *sol* at the specific time $tVal$. It evaluates the temperature distribution $u(x,t)$ for all $x$ at the given time $tVal$. We also use `PlotLabel` to add a label showing the specific time $tVal$ at which temperature profile is displayed.
Looking closer at the `DensityPlot` part, we create a density plot that represents the temperature profile at the given time $tVal$, again using `Evaluate` to replace *sol* with the evaluated solution at time $tVal$. We use `ColorFunction -> "TemperatureMap"` which applies a color gradient to represent the temperature values the best out of all color maps (from blue for low temperatures to red for high temperatures).

## 2.2 Example of usage, visualization

```
Quiet[heat[0, 2, 3, x, t, 1 / 2 t]]
```

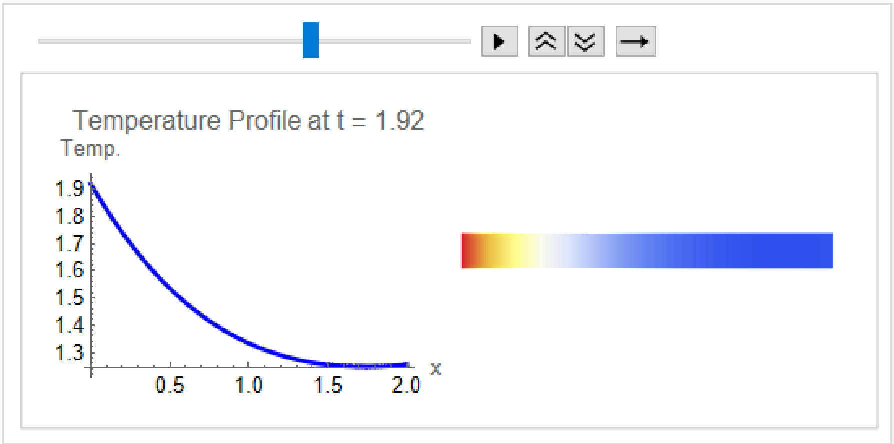


Figure 1: Example from laboratory

```
Quiet[heat[0, 1, 5, 2, 0, 0]]
```

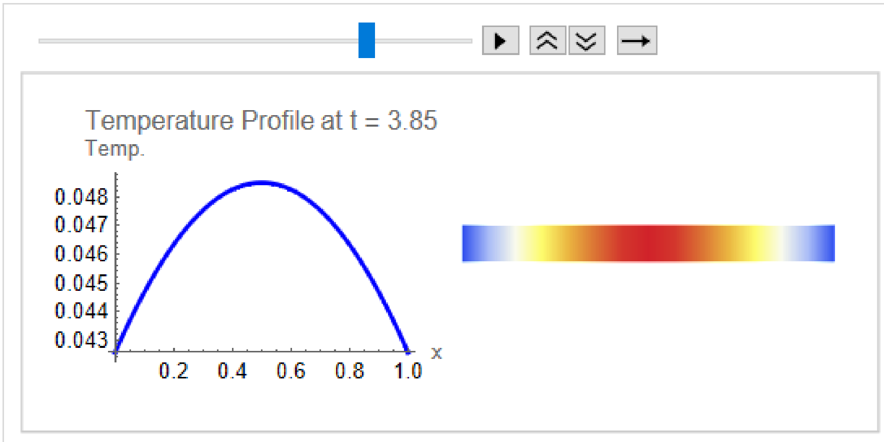


Figure 2: Uniform cooling of the bar

```
Quiet[heat[0, 1, 5, 0, 2, 2]]
```

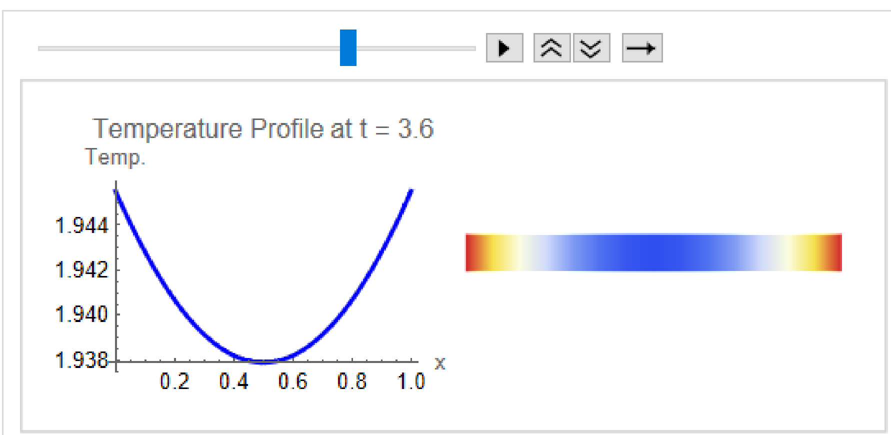


Figure 3: Uniform warming of the bar

```
Quiet[heat[0, 1, 5, 0, 0, 2]]
```

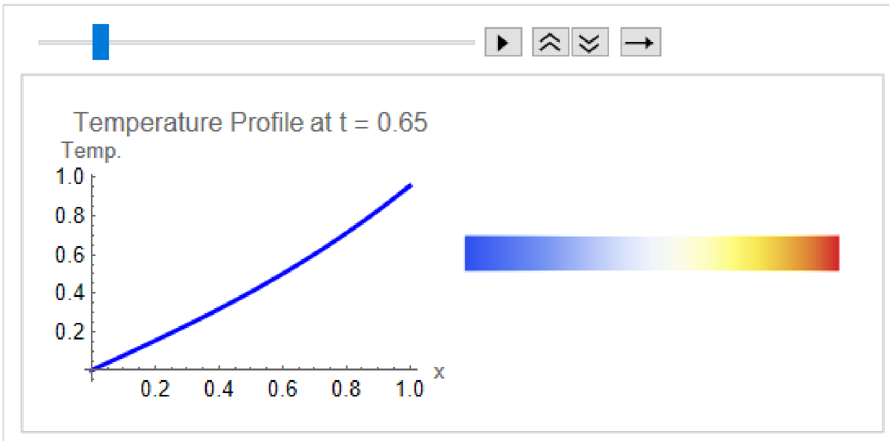


Figure 4: Warming from the right-hand side

```
Quiet[heat[0, 1, 5, Piecewise[{{x, 0 < x ≤ 1/2}, {1 - x, 1/2 < x ≤ 1}}], 1, 2]]
```

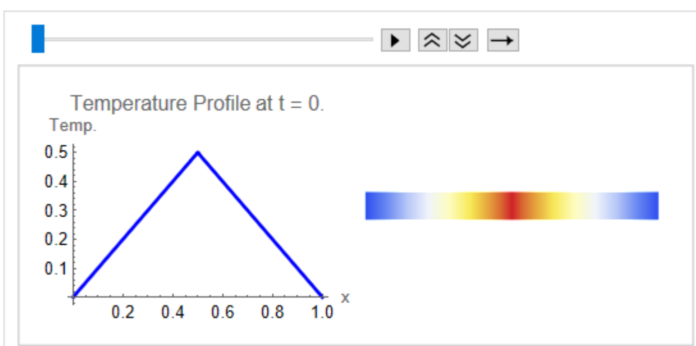


Figure 5: Non uniform warming - part 1

```
Quiet[heat[0, 1, 5, Piecewise[{{x, 0 < x ≤ 1 / 2}, {1 - x, 1 / 2 < x ≤ 1}}], 1, 2]]
```

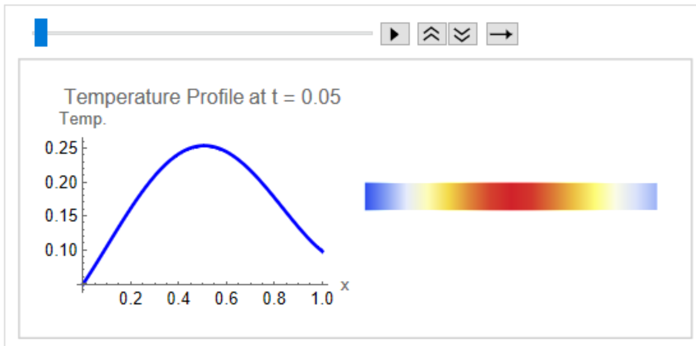


Figure 6: Non uniform warming - part 2

```
Quiet[heat[0, 1, 5, Piecewise[{{x, 0 < x ≤ 1 / 2}, {1 - x, 1 / 2 < x ≤ 1}}], 1, 2]]
```

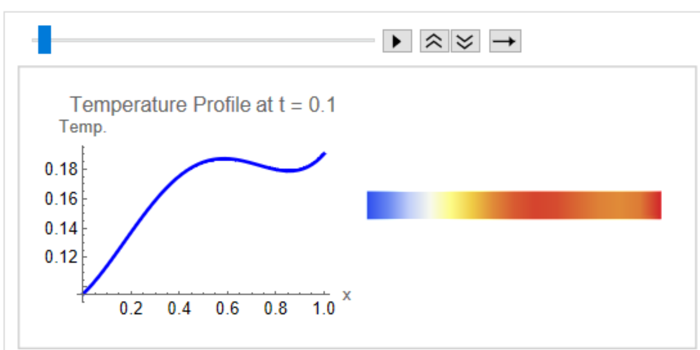


Figure 7: Non uniform warming - part 3

```
Quiet[heat[0, 1, 5, Piecewise[{{x, 0 < x ≤ 1 / 2}, {1 - x, 1 / 2 < x ≤ 1}}], 1, 2]]
```

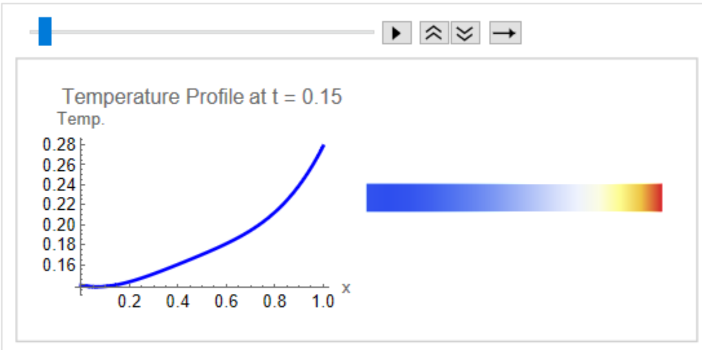


Figure 8: Non uniform warming - part 4

```
Quiet[heat[0, 1, 5, Piecewise[{{x, 0 < x ≤ 1 / 2}, {1 - x, 1 / 2 < x ≤ 1}}], 1, 2]]
```

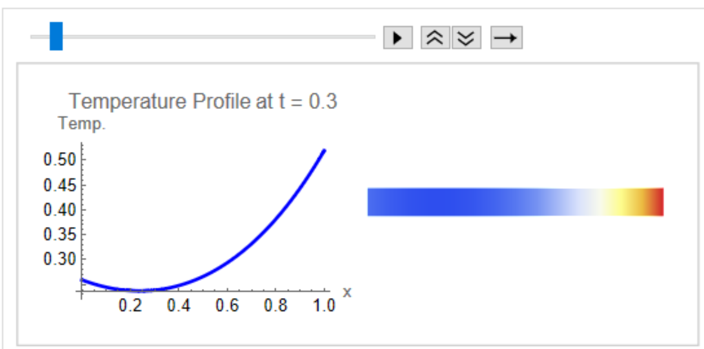


Figure 9: Non uniform warming - part 5

```
Quiet[heat[0, 1, 5, 0, Sin[t], Sin[2 t]]]
```

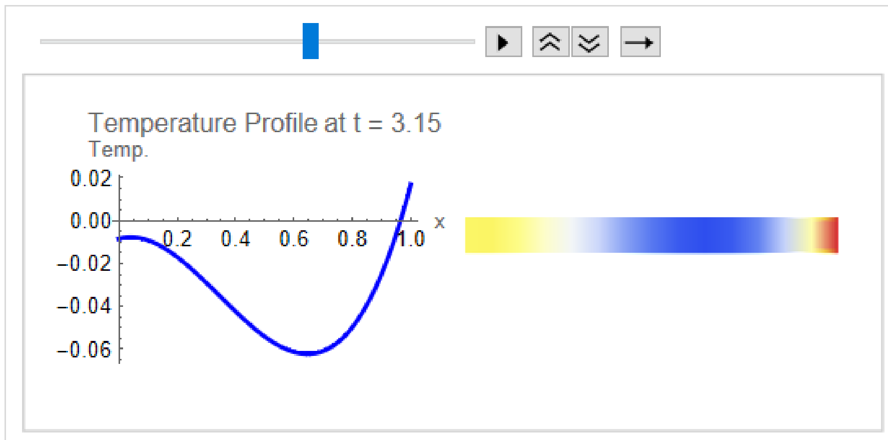


Figure 10: Sinusoidal boundaries - part 1

```
Quiet[heat[0, 1, 5, 0, Sin[t], Sin[2 t]]]
```

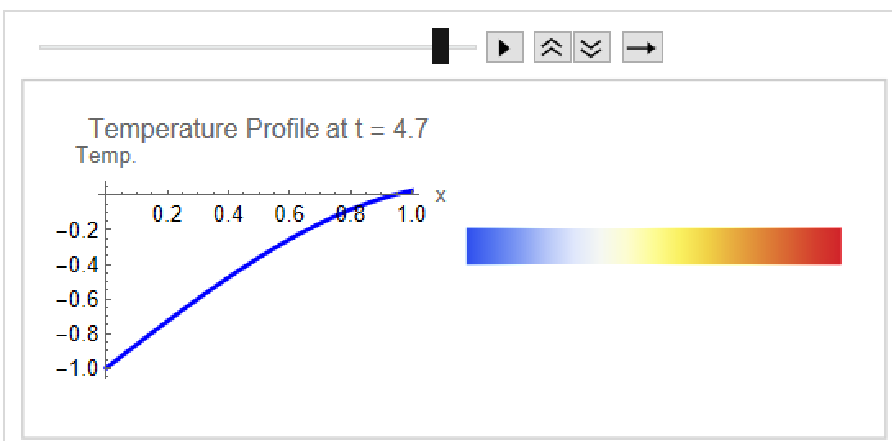


Figure 11: Sinusoidal boundaries - part 2

```
Quiet[heat[0, 2, 3, x^2, t, 0]]
```

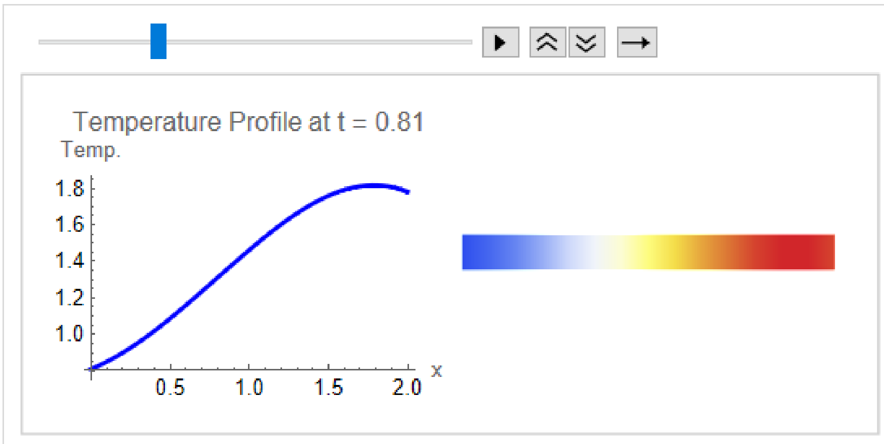


Figure 12: Parabolic initial temperature distribution - part 1

```
Quiet[heat[0, 2, 3, x^2, t, 0]]
```

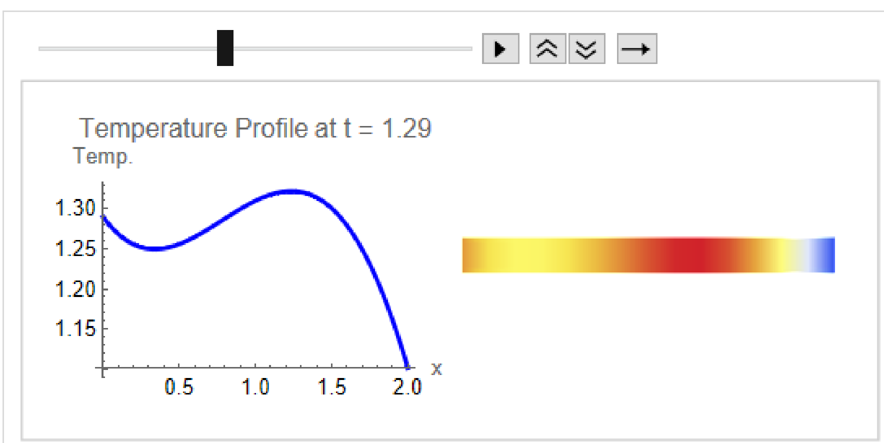


Figure 13: Parabolic initial temperature distribution - part 2

## 3 Enclosures

"STACHECKA Aleksandra Project 6.nb"