

Project 2: Classification and Regression, from linear and logistic regression to neural networks

Course: FYS-STK4155

Semester: Autumn 2020

Name: Sander Losnedahl

Abstract

a

Introduction

a

Preliminaries

a

Exercise 1a): a

Defining the heat equation

In this exercise we want to mathematically describe how heat can transfer within a rod of length L over a given time interval t . The rod will initially be heated at time $t = 0$ and the heat will decay after this time. How the heat changes as function of time and space is given by the heat equation as seen below.

$$\frac{\partial u(x, t)}{\partial t} = \frac{K_0}{c\rho} \frac{\partial^2 u(x, t)}{\partial x^2} \quad (1)$$

where $u(x, t)$ is the temperature at a specific time t and a position x . The term $\frac{K_0}{c\rho}$ is the thermal diffusivity which consists of thermal conductivity K_0 divided by the specific heat capacity c times the density of the material ρ . This quantity is a measure of the rate of heat transfer from the heated part of the rod to the cold part of the rod.

Equation 1 is a partial differential equation (PDE) which means that we need an initial condition and two boundary conditions in order to solve the equation. The initial condition can be interpreted as how much the rod is heated at $t = 0$ while the boundary conditions can be interpreted as how the heat reacts to the ends of the rod at $x = 0$ and $x = L$. The boundary conditions in this case is given by equations 2 and 3.

$$u(0, t) = 0 \quad (2)$$

$$u(L, t) = 0 \quad (3)$$

We can set the initial condition such that the rod is heated the most at the middle of the rod and least at the end of the road. Such a condition can be described by a sine function as given in equation 4.

$$u(x, 0) = \sin(\pi x) \quad (4)$$

One can observe from equation 4 that a rod of length 1 is heated the most at 0.5 and is not heated at all at the ends of the rod.

Analytical solution to the heat equation

We can now split equation 1 into its spatial and temporal components such that

$$u(x, t) = X(x)T(t) \quad (5)$$

Equation 5 can then be solved for x and t respectively. We start by taking the partial derivatives of equation 5 with respect to both t and x .

$$\begin{aligned} u_t &= X(x)T'(t) \\ u_{xx} &= X''(x)T(t) \end{aligned} \quad (6)$$

Equation 1 then takes the form

$$\begin{aligned} u_t &= u_{xx} \\ X(x)T'(t) &= X''(x)T(t) \\ \frac{X''(x)}{X(x)} &= \frac{T'(t)}{T(t)} = \lambda \end{aligned} \quad (7)$$

where λ is some constant. We can first solve equation 7 for its spatial part X such that

$$\begin{aligned} \frac{X''(x)}{X(x)} &= \lambda \\ X''(x) - \lambda X(x) &= 0 \end{aligned} \quad (8)$$

Equation 8 is recognized as a Sturm-Liouville problem when λ represents the eigenvalues of equation 8 (Hancock, 2006). This means that the solutions to equation 8 is given by equation 9.

$$X_n(x) = b_n \sin(n\pi x) \quad (9)$$

where n is an eigenfunction given by $n = \sqrt{\lambda}/\pi$ where λ are the eigenvalues of the Sturm-Liouville problem. Solutions then arise at $\lambda = \pi^2 n^2$ and from equation 7 the solution with respect to $T'(t)$ is found.

$$\begin{aligned} \frac{T'(t)}{T(t)} &= \lambda \\ T'(t) &= \lambda T(t) \end{aligned} \quad (10)$$

which has the solution

$$T'(t) = e^{\pi^2 n^2 t} \quad (11)$$

We can then find the solution for $u(x, t)$ by combining equation 9 and equation 11 such that

$$u_n(x, t) = b_n \sin(n\pi x) e^{\pi^2 n^2 t} \quad (12)$$

The subscript n denotes the individual eigenfunctions that solve the heat equation, however, most of them do not satisfy the initial condition in equation 4. The sum of these functions may satisfy the initial condition and thus, equation 12 becomes

$$u_n(x, t) = \sum_{n=1}^{\infty} b_n \sin(n\pi x) e^{\pi^2 n^2 t} \quad (13)$$

One can observe from equation 13 that the exponential term decreases as n increases. Therefore, a reasonable approximation may be to only utilize the first term of the sum such that equation 13 can be written as

$$u_n(x, t) = \sin(\pi x)e^{\pi^2 t} \tag{14}$$

where the constant b_n has been set to one. This is the final solution of the heat equation that will be implemented in the code.

Exercise 1b: A numerical approach to solving the heat equation

The forward Euler approach

The heat equation is a partial differential equation that can be solved numerically using the forward Euler algorithm (sometimes referred to as the explicit Euler method). This algorithm aims to incrementally approximate the gradient of spatial and temporal parts of the heat equation by utilizing the definition of the derivative as seen in equations 15 and 16.

$$u_t \approx \frac{u(x, t + \Delta t) - u(x, t)}{\Delta t} \quad (15)$$

$$u_{xx} \approx \frac{u(x + \Delta x, t) - 2u(x, t) + u(x - \Delta x, t)}{\Delta x^2} \quad (16)$$

where Δ represents the incremental step size in space or time. Since the initial conditions are known, the forward Euler algorithm can incrementally approximate the change in heat from the middle of the rod and out towards the boundary. Equations 15 and 16 can be combined using the heat equation from equation 1 such that

$$\begin{aligned} \frac{u(x, t + \Delta t) - u(x, t)}{\Delta t} &= \frac{u(x + \Delta x, t) - 2u(x, t) + u(x - \Delta x, t)}{\Delta x^2} \quad (17) \\ \frac{u_j^{n+1} - u_j^n}{\Delta t} &= \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} \end{aligned}$$

where the subscript j denotes the step in space and the subscript n denotes the step in time. We can solve equation 17 for u_j^{n+1} such that we can calculate the heat incrementally forward in time. Thus, equation 17 becomes

$$\begin{aligned} u_j^{n+1} &= \Delta t \left(\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} \right) + u_j^n \\ u_j^{n+1} &= \left(1 - 2\frac{\Delta t}{\Delta x^2} \right) u_j^n + \frac{\Delta t}{\Delta x^2} (u_{j+1}^n + u_{j-1}^n) \end{aligned} \quad (18)$$

Equation 18 is only stable for $\frac{\Delta t}{\Delta x^2} \leq 0.5$, so the implementation needs to adjust Δt according to Δx . The implementation of equation 18 allows us to solve the heat equation incrementally and the results are then compared to the analytical solution in the following section.

Comparing the analytical and numerical solutions of the heat equation

We now compare how the heat translates throughout the rod at a given time for both the analytical implementation and for the forward Euler implementation. It was found

that the heat decayed quickly. Therefore, good time intervals of observing the heat was found to be at $t = 0$ (initial heat), $t = 0.5$ and $t = 1$. Figures 1, 2 and 3 shows the heat distribution in the rod at these given times for $\Delta t = 0.005$ and $\Delta x = 0.1$ for both the analytical and forward Euler implementations (and their difference).

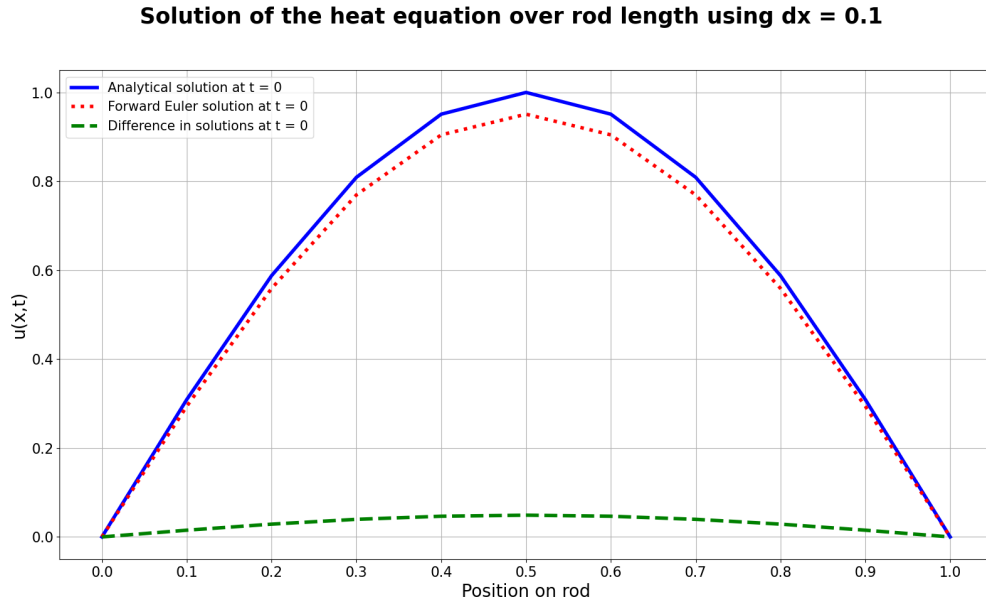


Figure 1: Heat distribution within the rod for both analytical (blue) and forward Euler implementations (red). The heat is measured at $t = 0$ using step sizes of $\Delta t = 0.005$ and $\Delta x = 0.1$.

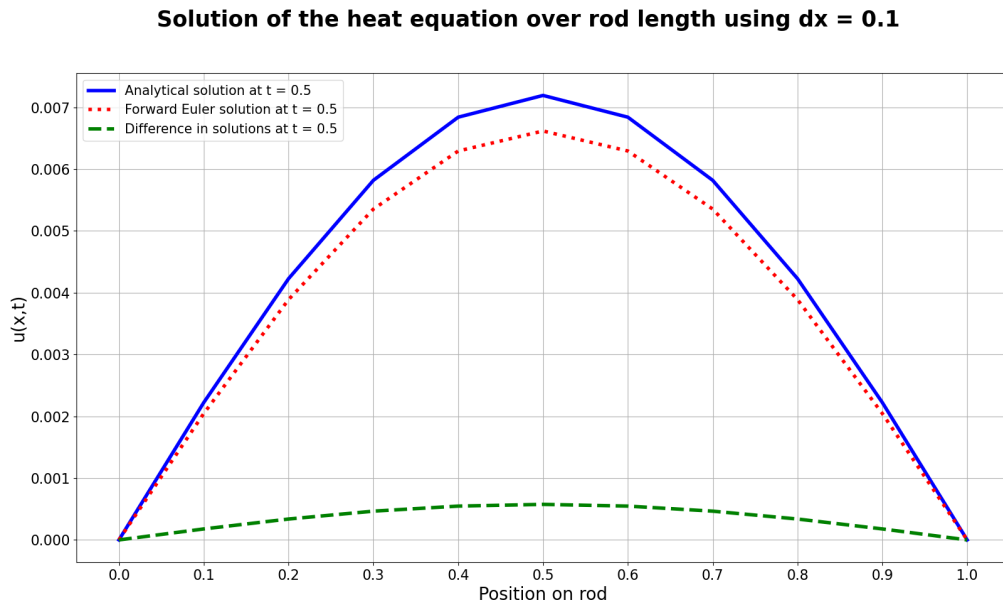


Figure 2: Heat distribution within the rod for both analytical (blue) and forward Euler implementations (red). The heat is measured at $t = 0.5$ using step sizes of $\Delta t = 0.005$ and $\Delta x = 0.1$.

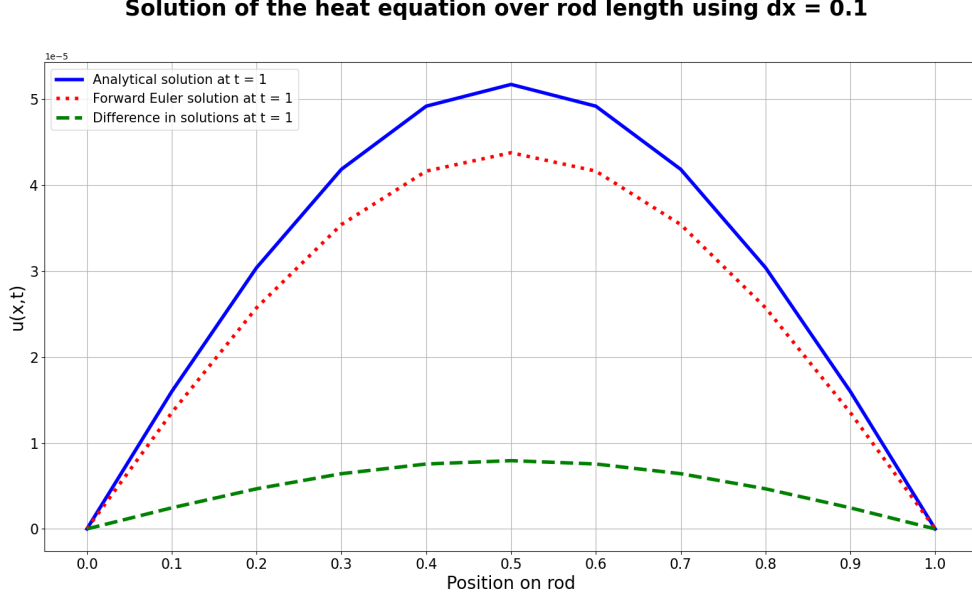


Figure 3: Heat distribution within the rod for both analytical (blue) and forward Euler implementations (red). The heat is measured at $t = 1$ using step sizes of $\Delta t = 0.005$ and $\Delta x = 0.1$.

One first observed that the rod is heated the most at the middle $x = 0.5$, which is according to the initial condition of equation 4. One can also observe that the forward Euler lies pretty close to the analytical solution. The difference between the analytical solution and the forward Euler solution decreases as time increases (the y-axis becomes smaller, so don't get fooled). Additionally, one can observe that the heat decays fairly quickly as most of the heat is gone within 1 second.

We now decrease the step size in space from $\Delta x = 0.1$ to $\Delta x = 0.01$, and this means further we need to adjust the step size in time according to $\frac{\Delta t}{\Delta x^2} \leq 0.5$. Therefore, the new step size in time becomes $\Delta t = 0.5 * 0.01^2 = 0.00005$. Figures 4, 5 and 6 shows the same as the above figures, but using the new step sizes.

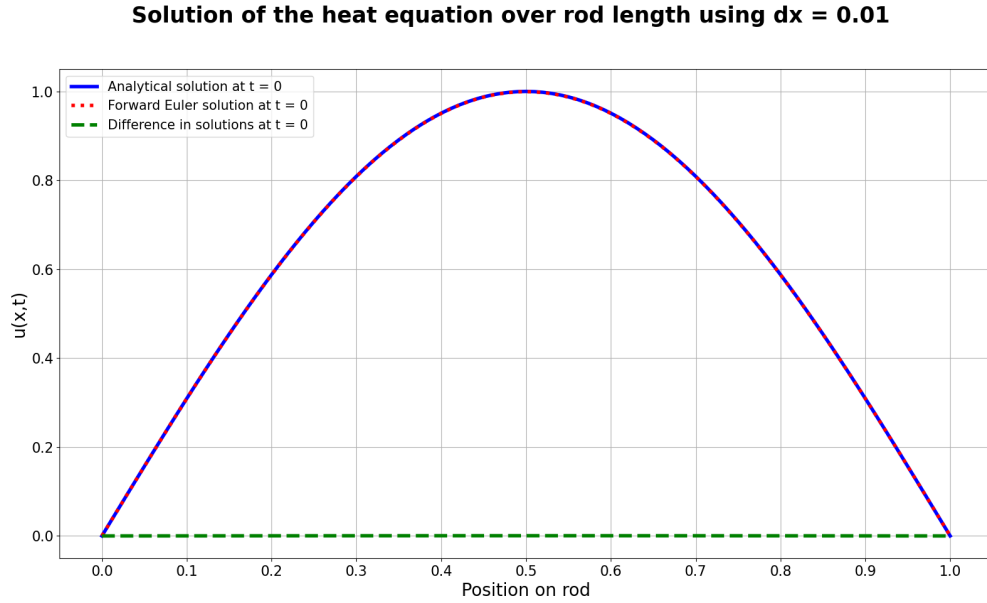


Figure 4: Heat distribution within the rod for both analytical (blue) and forward Euler implementations (red). The heat is measured at $t = 0$ using step sizes of $\Delta t = 0.00005$ and $\Delta x = 0.01$.

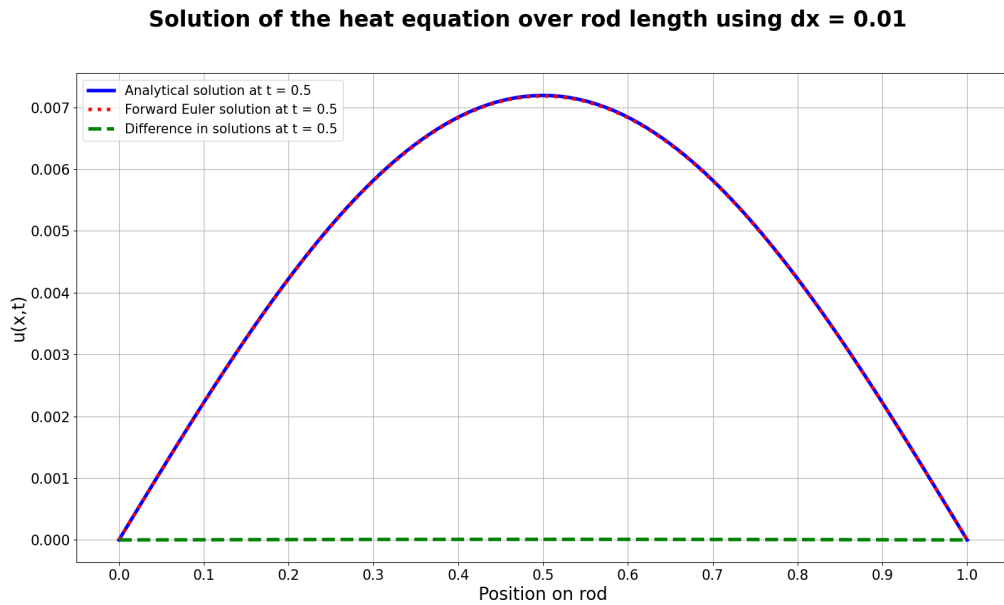


Figure 5: Heat distribution within the rod for both analytical (blue) and forward Euler implementations (red). The heat is measured at $t = 0.5$ using step sizes of $\Delta t = 0.00005$ and $\Delta x = 0.01$.

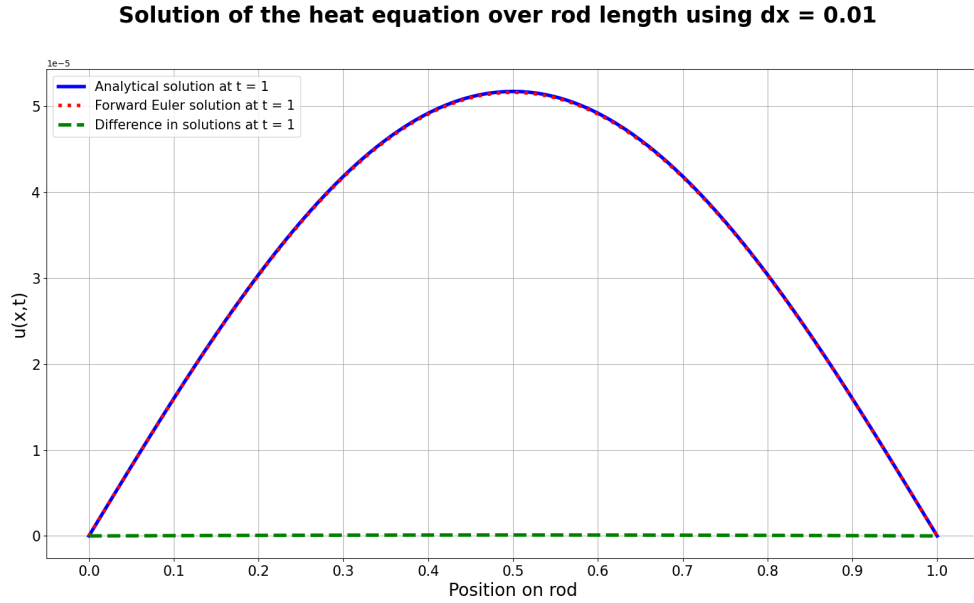


Figure 6: Heat distribution within the rod for both analytical (blue) and forward Euler implementations (red). The heat is measured at $t = 1$ using step sizes of $\Delta t = 0.00005$ and $\Delta x = 0.01$.

It is now observed that the forward Euler implementation matches the analytical implementation perfectly at all times. The reason for this is because the definition of the derivative in which the forward Euler relies on is a better approximation when the step sizes are small. This means further that the forward Euler implementation can accurately represent the heat distribution within a rod at all times and places.

Exercise c): Solving PDEs with deep neural networks

Neural network implementation

The concepts and details of a neural network was discussed in Project 2 which can be found at <https://github.com/sanderwl/FYS-STK4155/tree/master/Project2/Project2>. Much of the concepts are the same, that is, we pass an input through a network of hidden layers and neurons and an output is the same. However, the implementation is slightly different.

The neural network still aims to minimize some cost function which in this case is the MSE given by equation 19.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\text{prediction error})^2 \quad (19)$$

Since $\frac{\partial u(x,t)}{\partial t} - \frac{\partial^2 u(x,t)}{\partial x^2} = 0$ solves the heat equation, it can be utilized as the prediction error. Equation 19 then becomes

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n \left(\frac{\partial u(x,t)}{\partial t} - \frac{\partial^2 u(x,t)}{\partial x^2} \right)^2 \quad (20)$$

Since we now pretend not to know any solution to the heat equation, we can implement a trial solution. A trial solution must meet the boundary conditions (with already known initial condition). This solution is not guaranteed to minimize the cost function in equation 20, in fact, it is rare that such a solution minimize the cost function. Therefore, we can tune the weights of the neural network through gradient descent and update the weights to find different trial solutions. For each update, the cost function should minimize and a more accurate solution to the heat equation is found. The trial solution g used in this project is given by equation QQQ.

$$g = (1 - t) * \sin(\pi x) + x(1 - x)t * \text{output} \quad (21)$$

where $\sin(\pi x)$ is the initial condition and output is the output from the neural network in which the gradient descent must handle.

The neural network is then trained by sending the two variables x and t through the network followed by a weight update through gradient descent on the cost function. This process happens many times and the final trial solution should reflect the analytical solution.

My own implementation vs. Tensorflow

A neural network algorithm was implemented from scratch using what I have learned in this course. However, this algorithm proved way too slow and was thus deemed unfit for this exercise as time is limited. Instead, Tensorflow algorithms were implemented as these yielded fast, stable and good results. Additionally, Tensorflow is easy to handle and has a lot of nice features to play around with. In particular, the Tensorflow version of

the Adam optimization algorithm proved to yield great results. The Adam optimization algorithm is really a combinations between stochastic gradient descent and RMSprop (Kingma and Ba, 2014). The algorithm is essentially stochastic gradient descent with momentum where a moving average of the gradient is used instead of the entirety of the gradient. Additionally, the Adam algorithm uses adaptive learning rates just like RMSprop.

Comparing the analytical and neural network solutions of the heat equation

After implementing the above discussed neural network algorithm, we can take the final trial solution and use it to solve the heat equation and compare it with the analytical and forward Euler implementations. First we have to set some parameters for the neural network to work with. These parameters are defined in table 1 below.

Table 1: Parameters of the neural network.

Δt	0.1
Δx	0.1
Number of layers	10
Number of neurons in each layer	10
Number of iterations	100000
Initial learning rate	0.001

Since we are doing a stochastic gradient descent like algorithm (Adam optimizer) we have to set some initial learning rate. The number of iterations denotes how many times we do the feed forward/back propagation loop. The other parameters were found through trial and error. These parameters could have been optimized like in Project 2, but the results have shown to be good without exact optimization.

We can first plot the entirety of the 3D heat equation solution as done in figures 7 and 8.

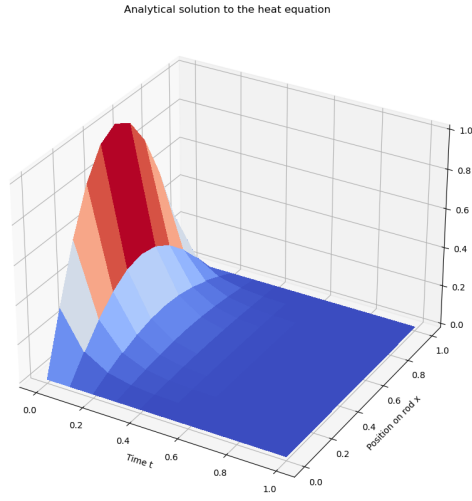


Figure 7: Heat distribution for the rod as function of space and time for the analytical implementation.

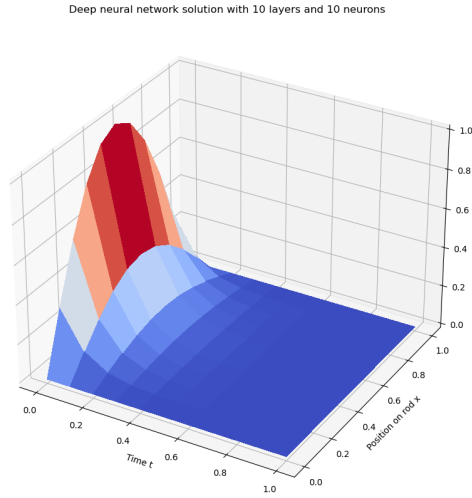


Figure 8: Heat distribution for the rod as function of space and time for the neural network implementation.

One can observe that the two implementations in figures 7 and 8 are pretty similar. In order to get a better comparison, we can plot the heat distribution at constant times $t = 0$ (initial heat), $t = 0.5$ and $t = 1$ just like before.

Solution of the heat equation over rod length using the Tensorflow neural network

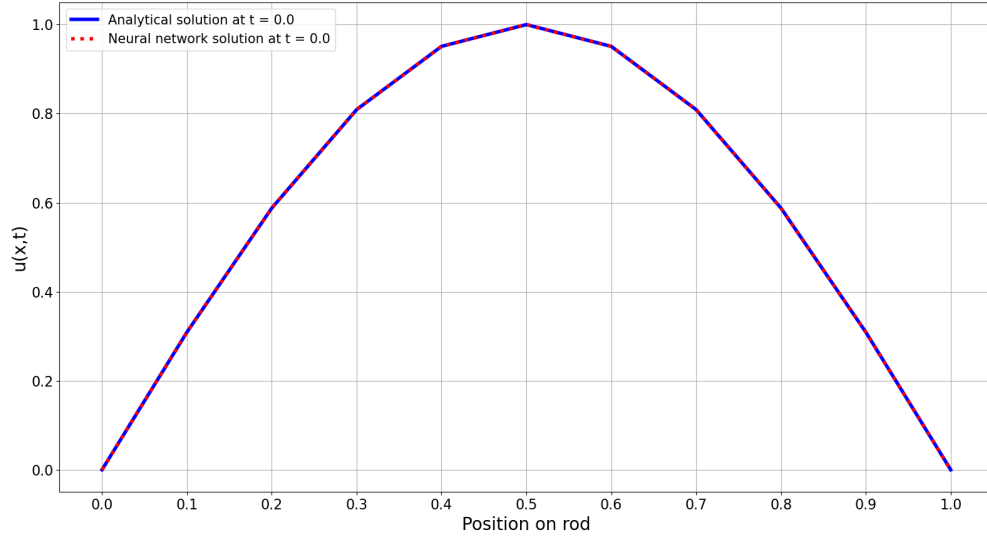


Figure 9: Heat distribution within the rod for both analytical (blue) and neural network implementations (red). The heat is measured at $t = 0$ using the parameters of table 1.

Solution of the heat equation over rod length using the Tensorflow neural network

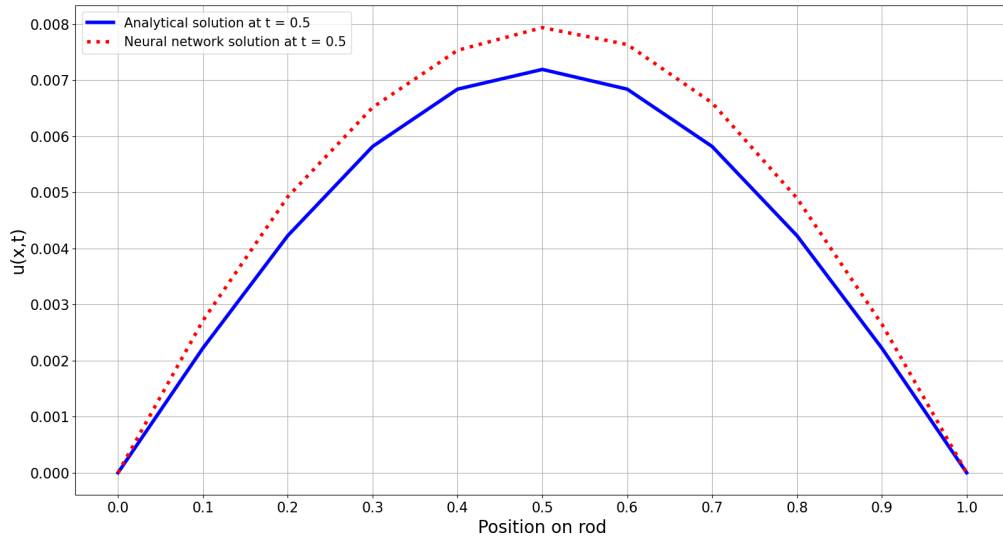


Figure 10: Heat distribution within the rod for both analytical (blue) and neural network implementations (red). The heat is measured at $t = 0.5$ using the parameters of table 1.

Solution of the heat equation over rod length using the Tensorflow neural network

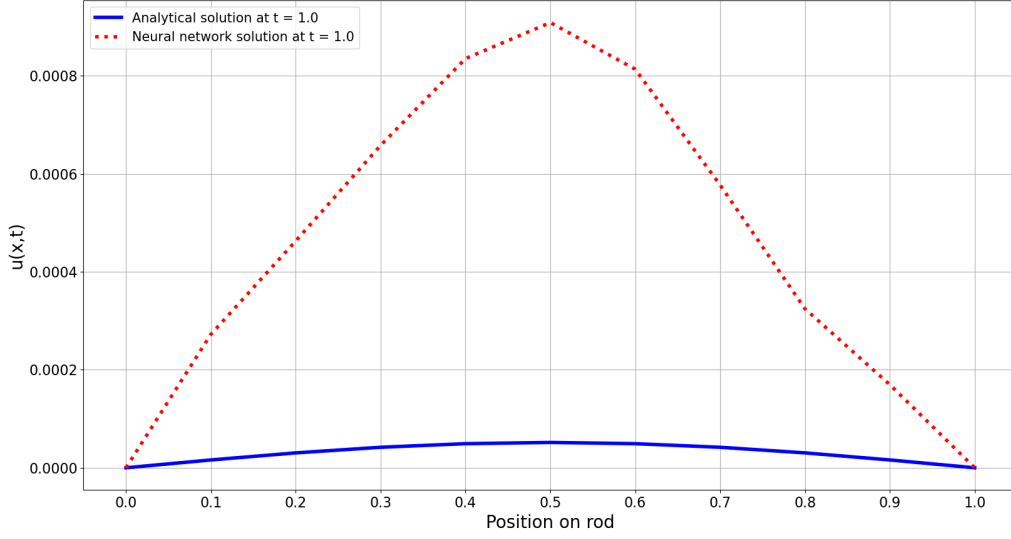


Figure 11: Heat distribution within the rod for both analytical (blue) and neural network implementations (red). The heat is measured at $t = 1$ using the parameters of table 1.

One can observe that the initial heat distribution in figure 9 is equal for the analytical and neural network implementations. After 0.5 seconds however, the two implementations starts to deviate slightly. At $t = 1$ the two implementations are essentially zero, but the neural network still maintain the shape of the previous heat distributions. This was a recurring problem with the neural network implementation regardless of input parameters. However, since the error is very small, the result is deemed acceptable.

Exercise e): a

a

Conclusion

a

Future work

a

References

- Hancock, M. J., 2006, *The 1-D Heat Equation*, *Linear Partial Differential Equations*, page 5, Available from <https://ocw.mit.edu/courses/mathematics/18-303-linear-partial-differential-equations-fall-2006/lecture-notes/heateqni.pdf>
- Kingma, D. P., Ba, J. L., 2014 *Adam: A Method for Stochastic Optimization*, pp 1, arXiv:1412.6980v9