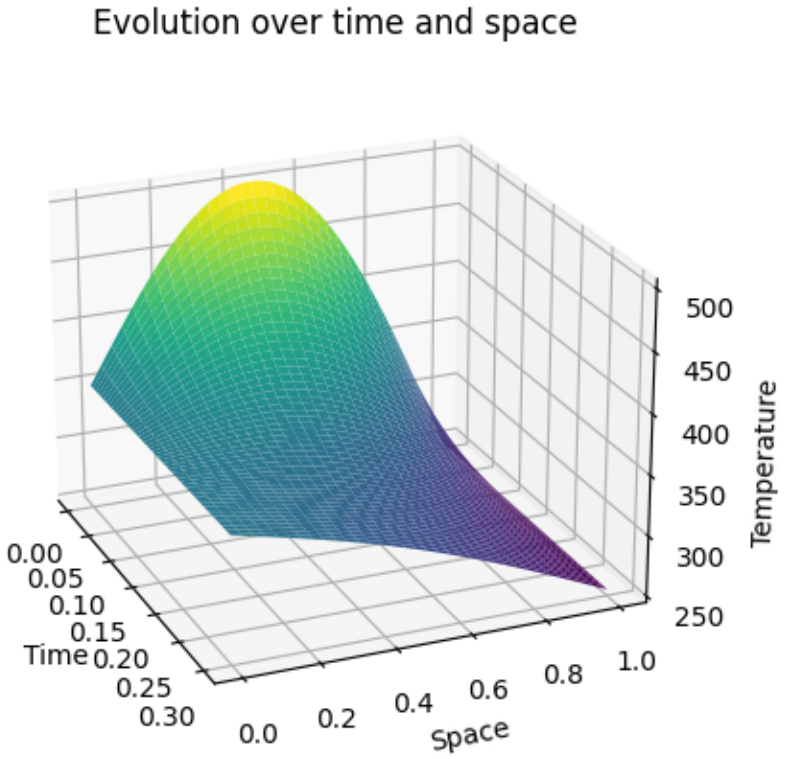
VU Scientific Computing

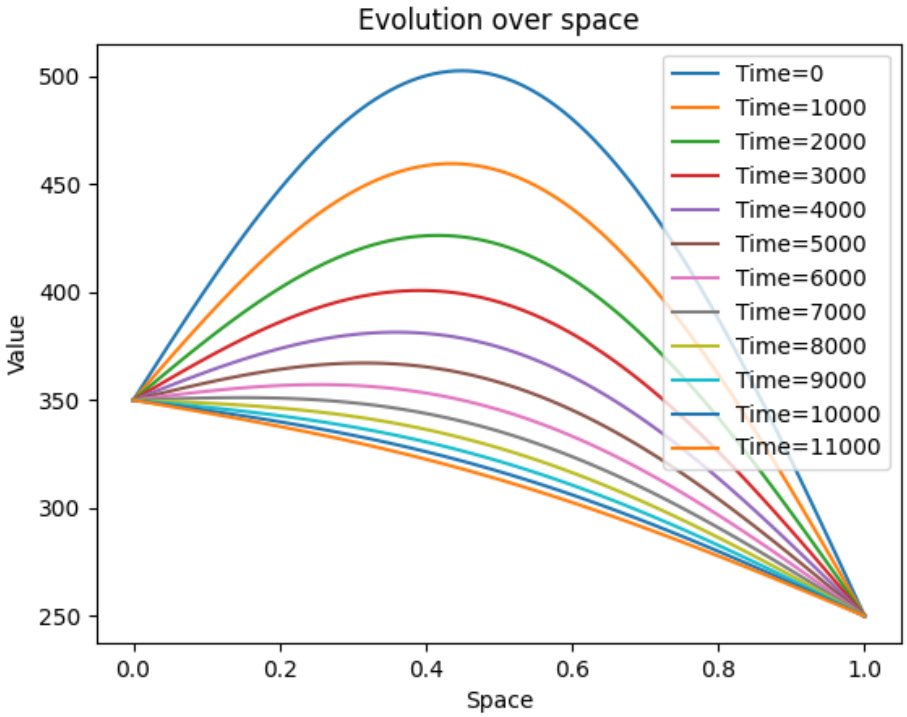
Gruppe:

Name	Matrikelnummer
David Gstir	12112059
Thomas Schwarz	12035825

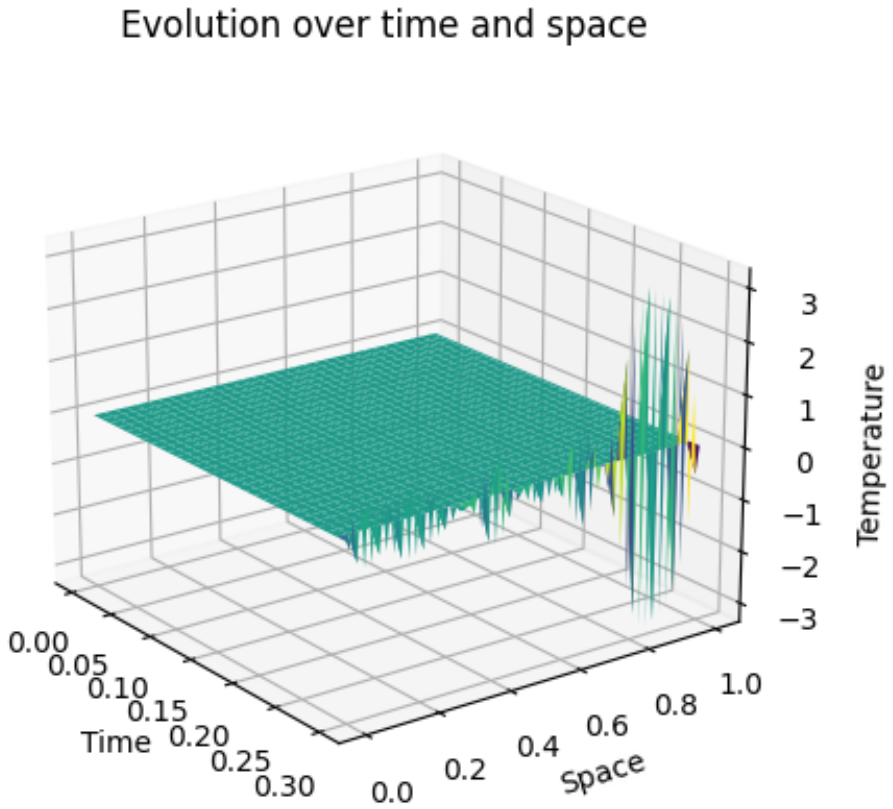
In this file there can be found all the plots and answers to questions for all the exercises. All the used parameters for the simulations and plots can be seen in the individual python files.

1.5


Evolution over time and space
Temperature
500
450
400
350
300
250
Time
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Space
0.0
0.2
0.4
0.6
0.8
1.0


Evolution over space
Value
Space
Time=0
Time=1000
Time=2000
Time=3000
Time=4000
Time=5000
Time=6000
Time=7000
Time=8000
Time=9000
Time=10000
Time=11000

The temperature always gets closer to the equilibrium. The speed of temperature change also decreases while getting closer to the equilibrium. If we violate the CFL condition, the plot looks like this:



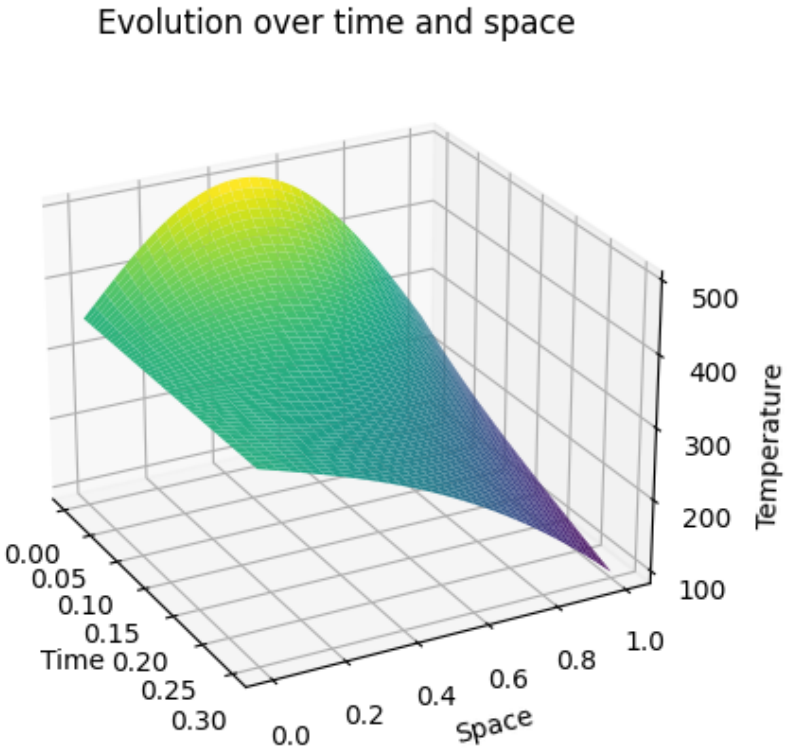
This happens because CFL imposes a limit on the size of the time step relative to the spatial discretization in order to maintain stability. When we violate this condition, it can lead to numerical instability.

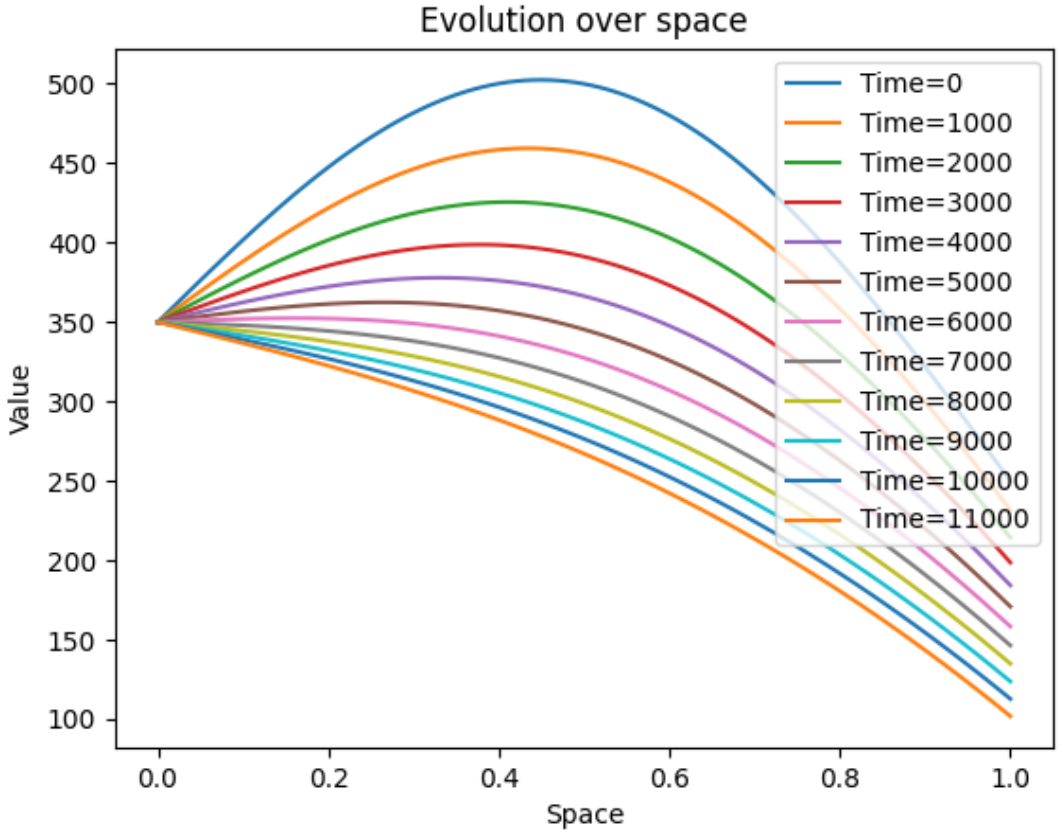
1.6

The decay rate depends on the initial temperature profiles because it is different for perhaps constant, linear, and also for oscillations in the initial temperature distribution. For example, a constant initial temperature profile does typically not have a decay rate because there are no spatial temperature differences. On the other side on linear initial temperature profiles, the decay rate depends on the slope of the temperature gradient.

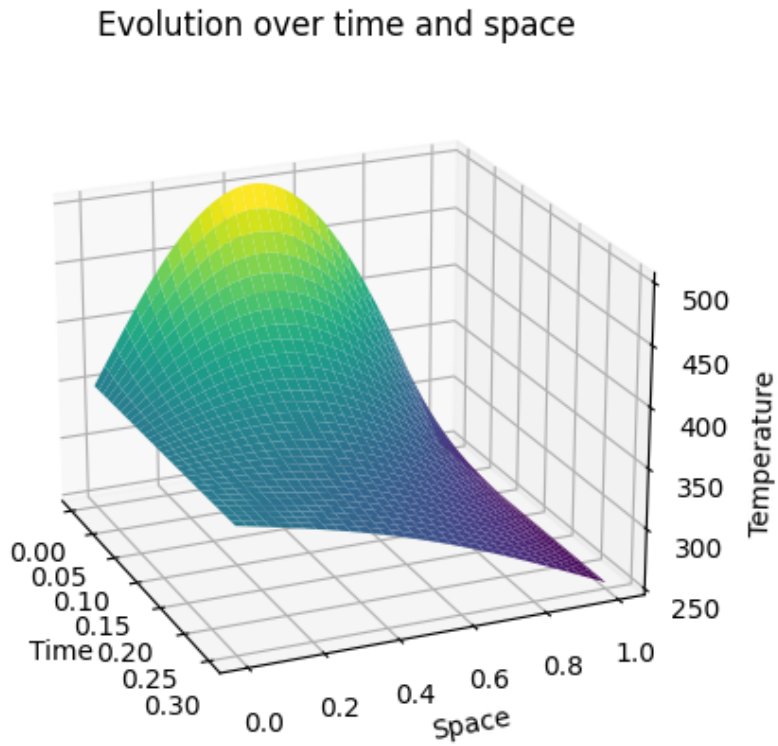
1.7

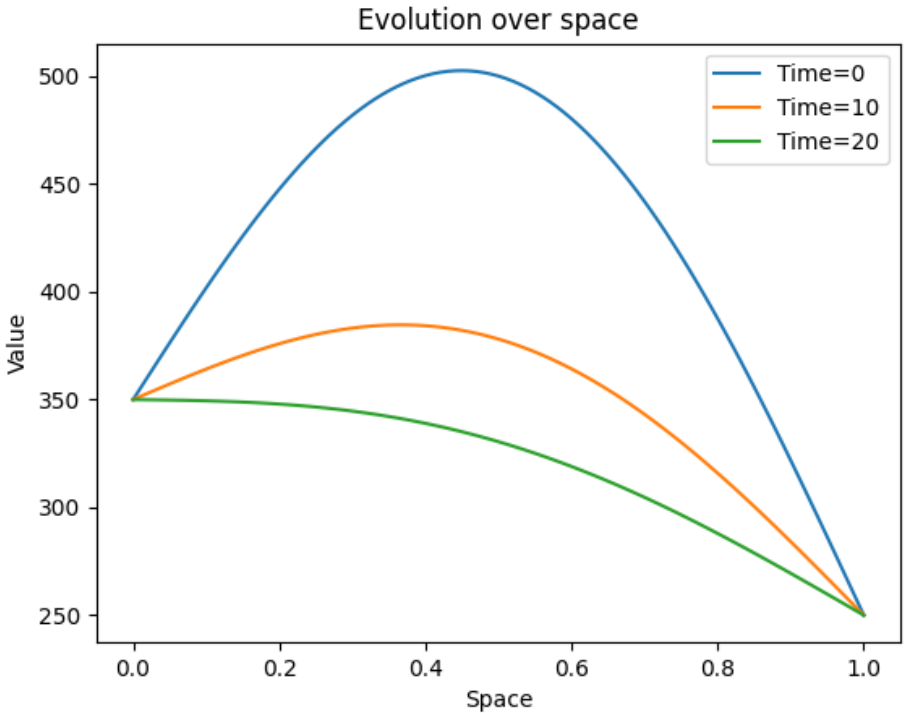
Implemented the same problem as in 1.6 but this time we used homogeneous Neumann boundary conditions on the right end of the rod. As we can see the right end of the rod gets colder and colder unlike in the previous exercise where the end of the rod temperature was constant. This makes sense because we defined the boundary condition like that.





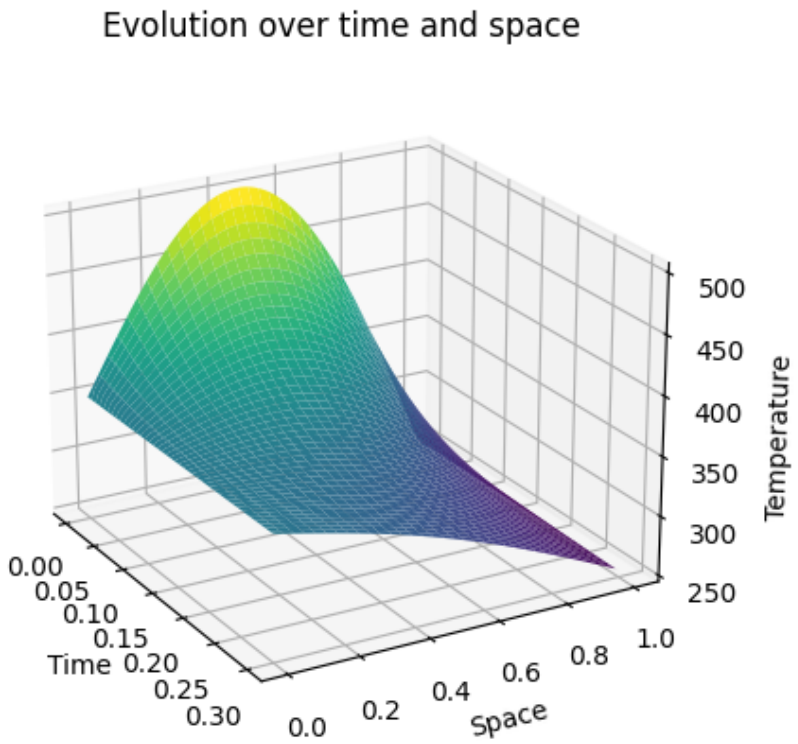
1.9

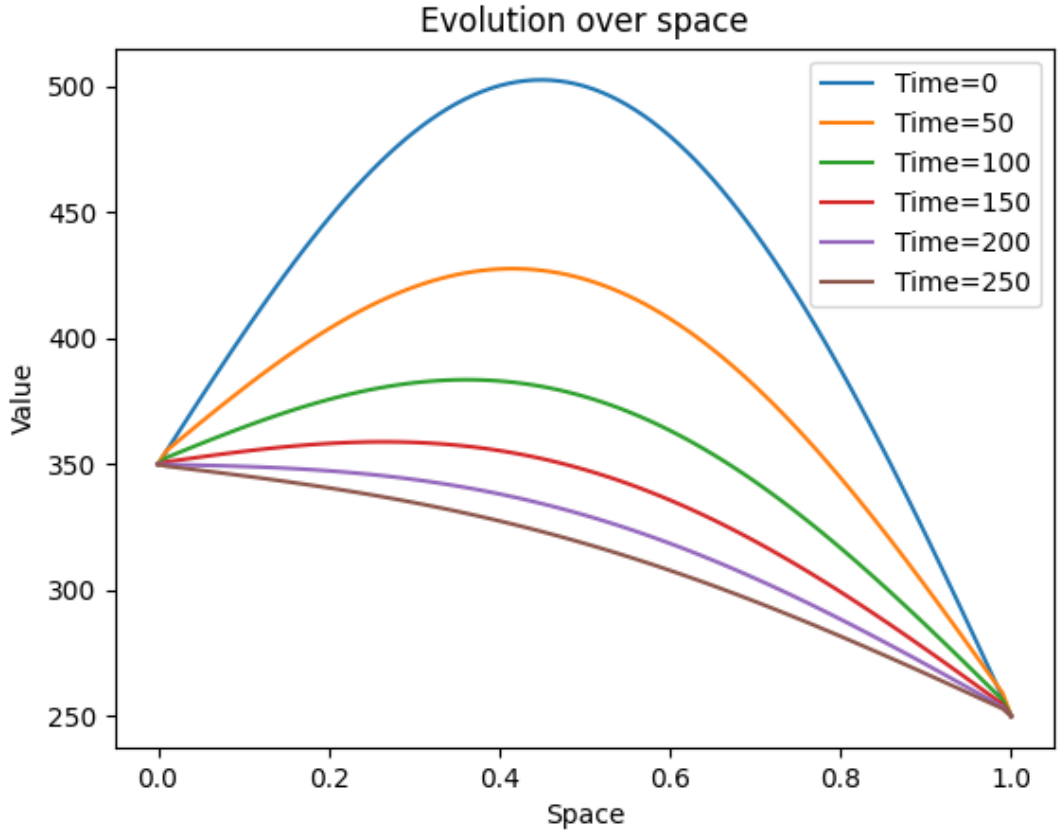




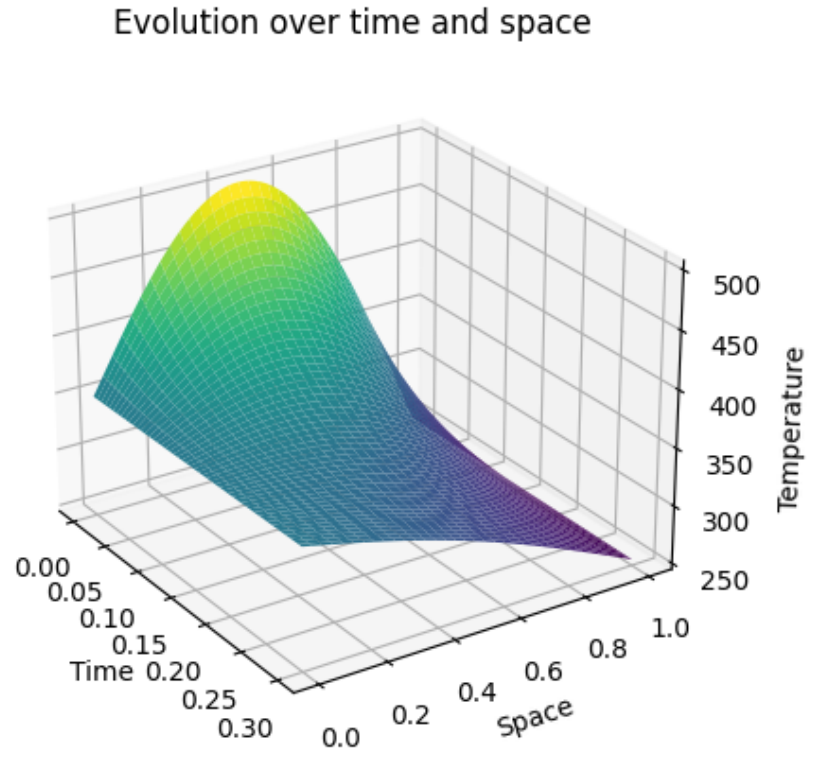
For the implicit Euler scheme, we used a much larger time step with the size of 0.01 in comparison to the time step 0.000025 with the explicit Euler scheme. Although the time step is so much larger with the implicit Euler scheme, we still get stable numerical solutions with it.

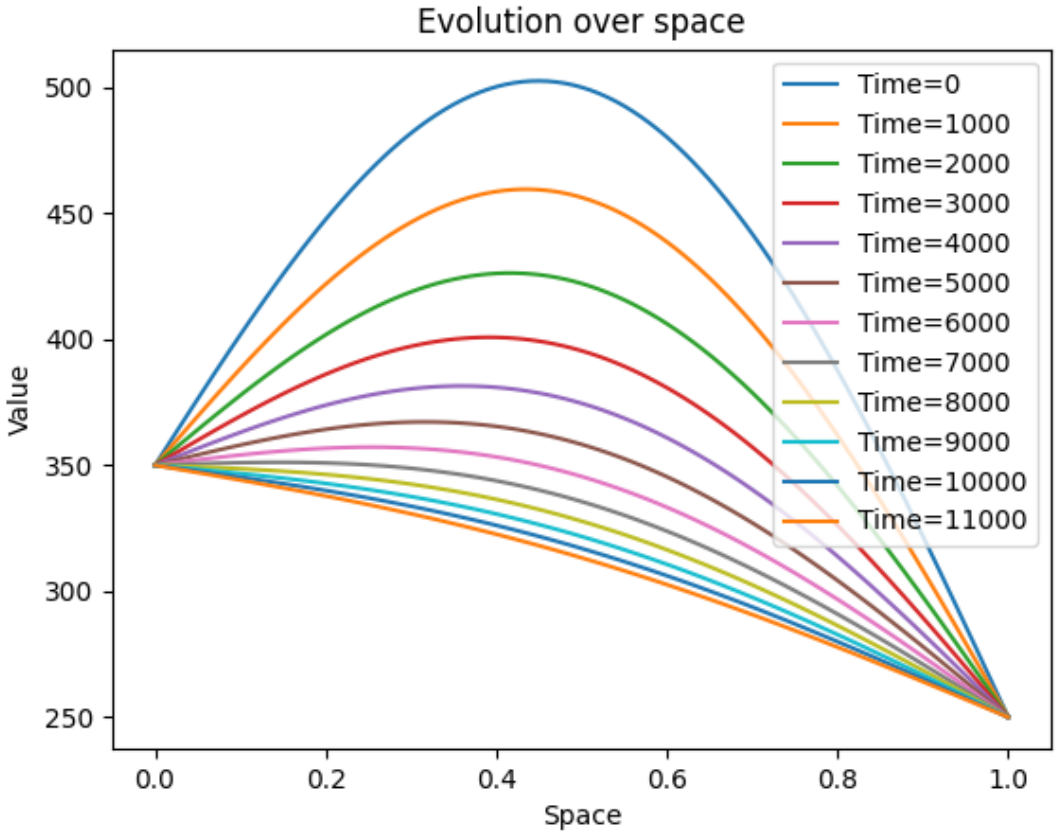
1.10


Evolution over time and space
Temperature
500
450
400
350
300
250
Time
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Space
0.0
0.2
0.4
0.6
0.8
1.0

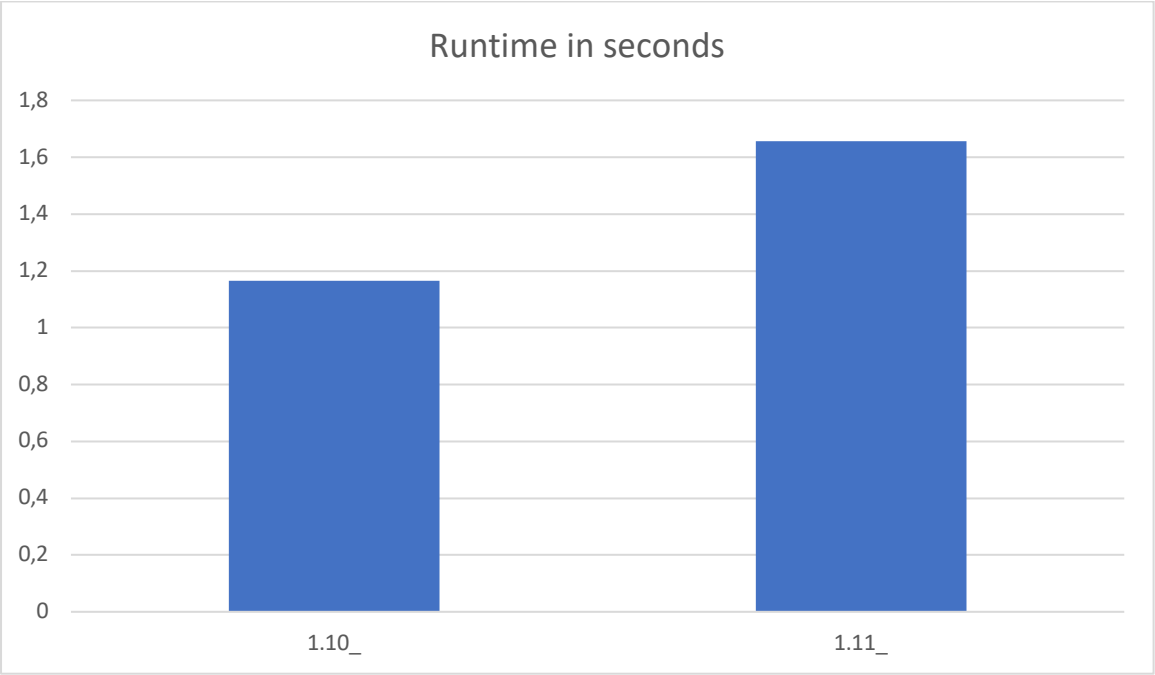

Evolution over space
Time=0
Time=50
Time=100
Time=150
Time=200
Time=250
Value
500
450
400
350
300
250
Space
0.0
0.2
0.4
0.6
0.8
1.0

1.11


Evolution over time and space
Temperature
500
450
400
350
300
250
Time
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Space
0.0
0.2
0.4
0.6
0.8
1.0


Evolution over space
Value
Space
Time=0
Time=1000
Time=2000
Time=3000
Time=4000
Time=5000
Time=6000
Time=7000
Time=8000
Time=9000
Time=10000
Time=11000

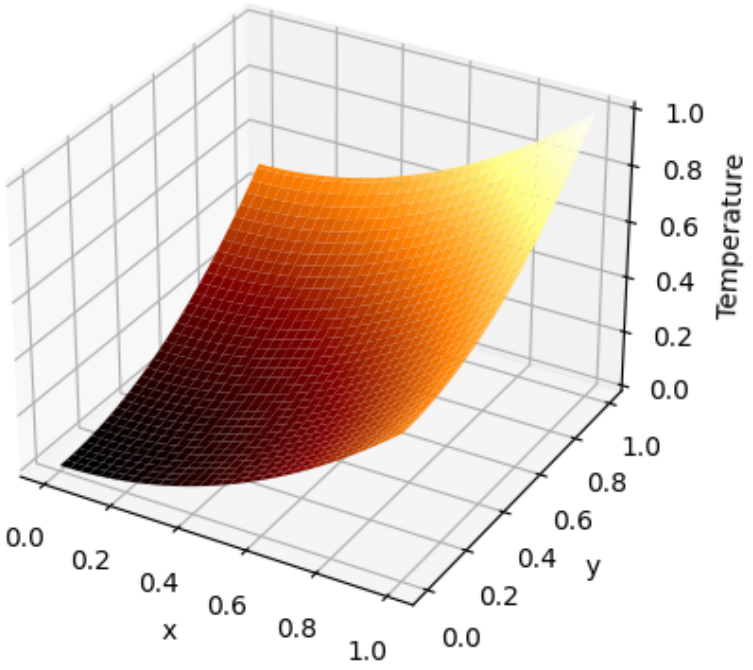
If we compare these two implementations with the explicit Euler scheme, we can see the following runtimes:



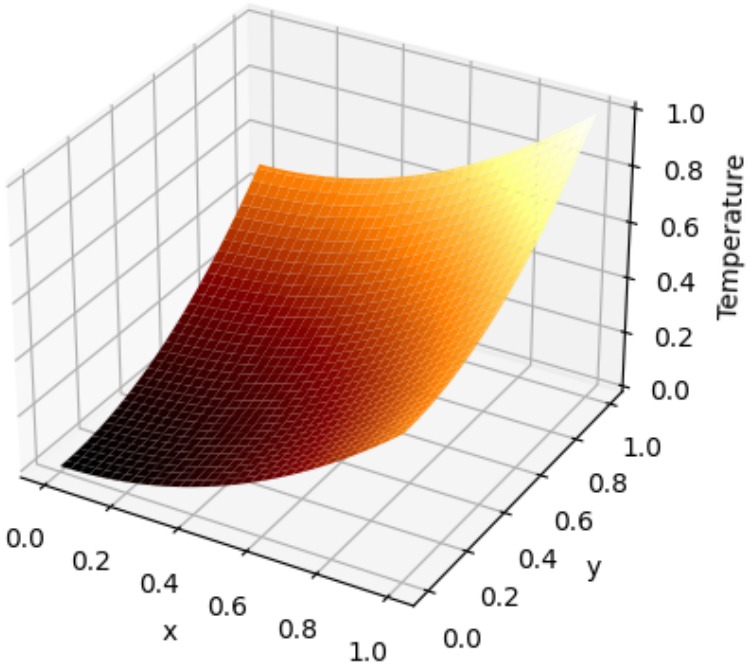
1.12/1.13

We plotted the results at three different times. First time directly at the beginning, second time after half the time steps and the third time at the end.

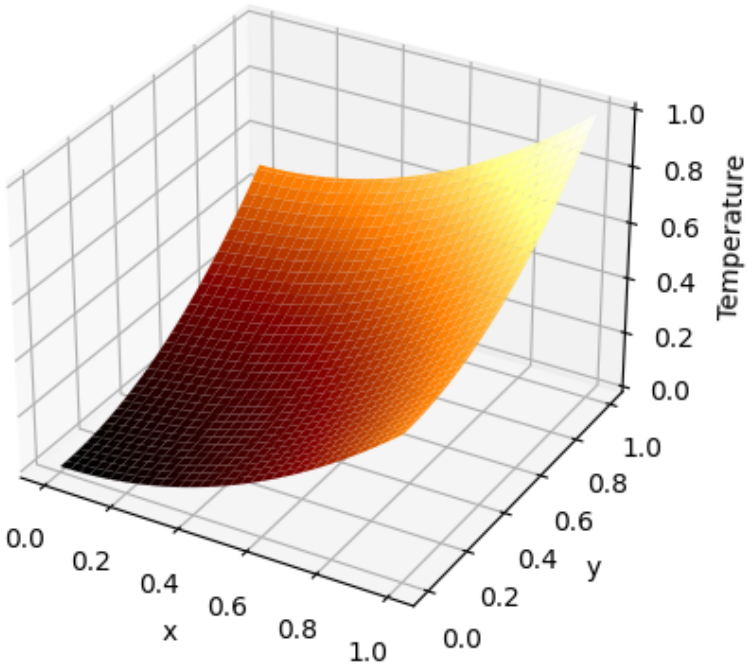
At the beginning:



Half of the time steps:



At the end:

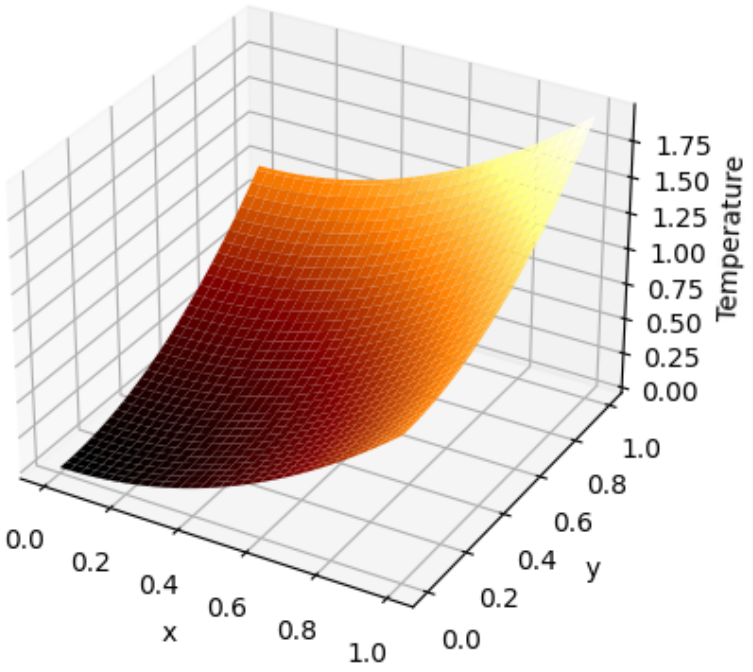


We can see in the upper plots, that the equilibrium is nearly a plane surface through the given boundary points.

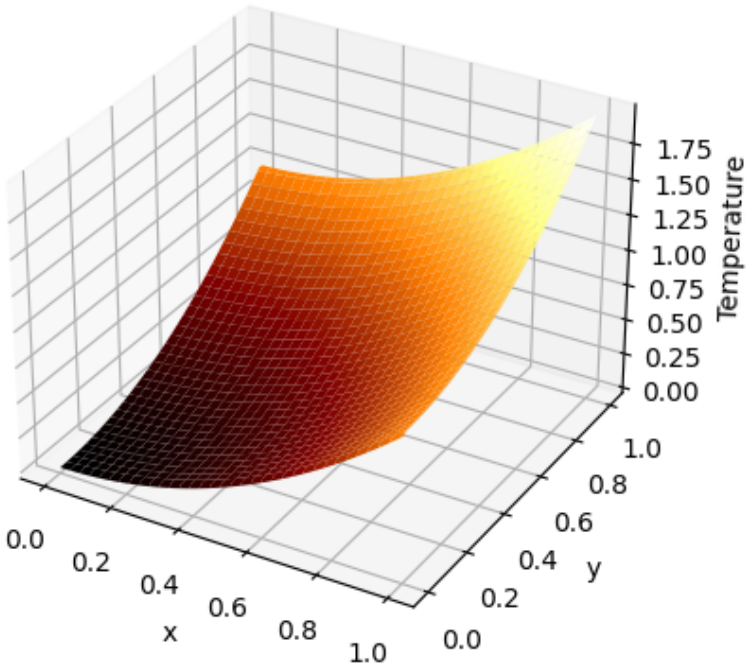
1.14

We added here another plot, a contour 2D plot after half the time steps, to the 3D plots at the beginning, after half the time steps and at the end.

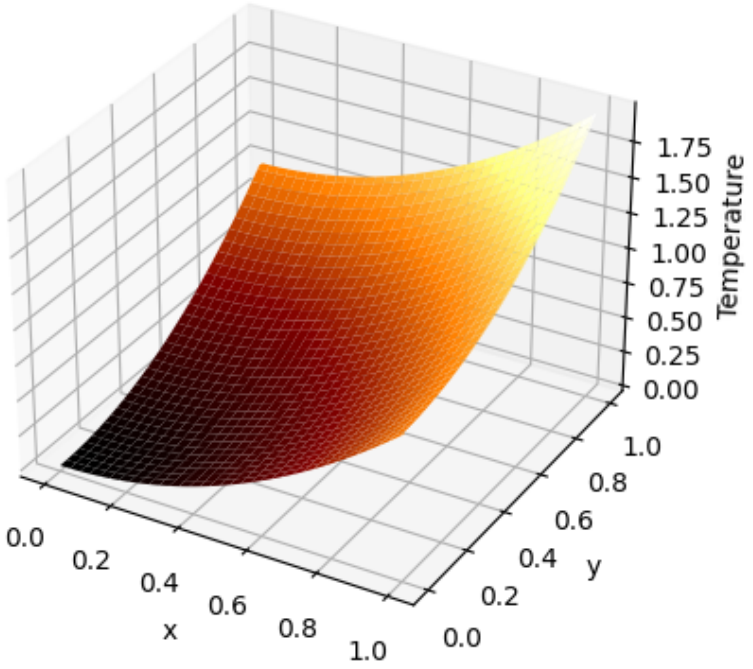
At the beginning:



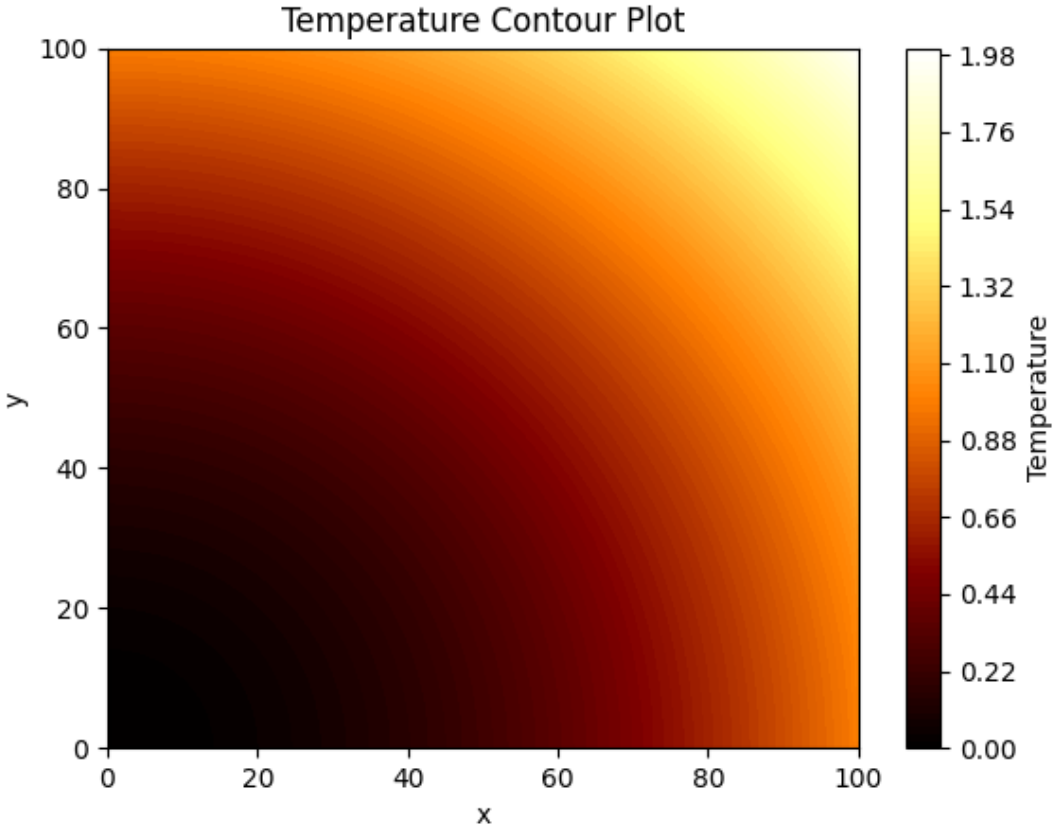
Half of the time steps:



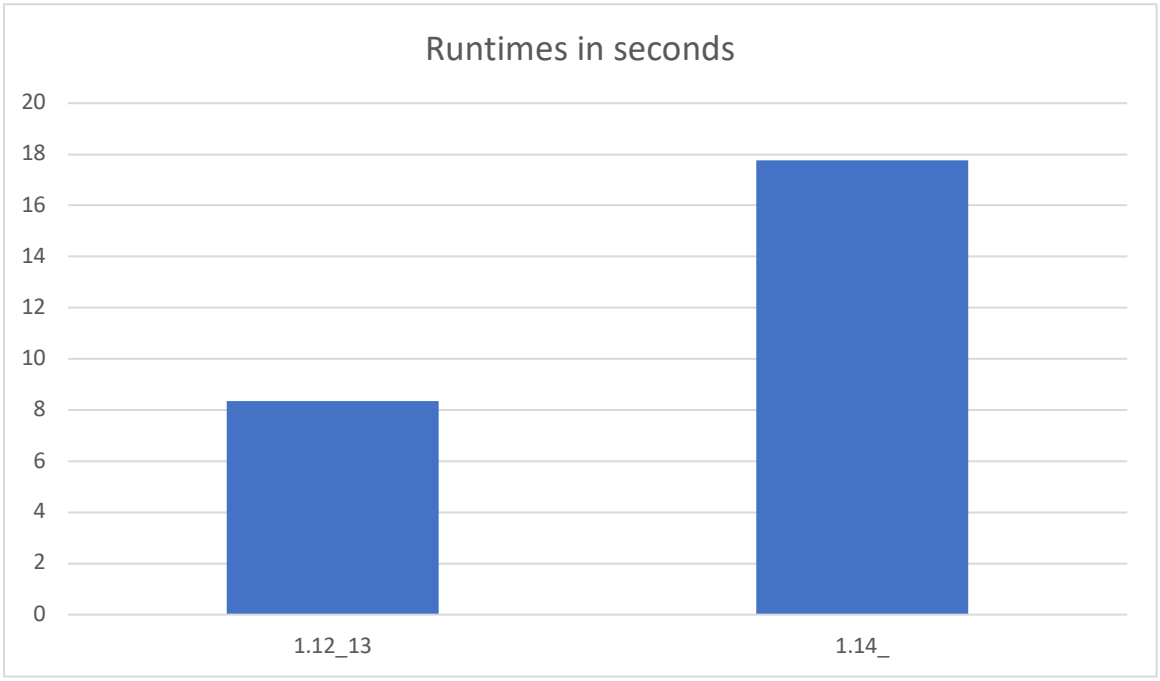
At the end:



Contour plot after half of the time steps:

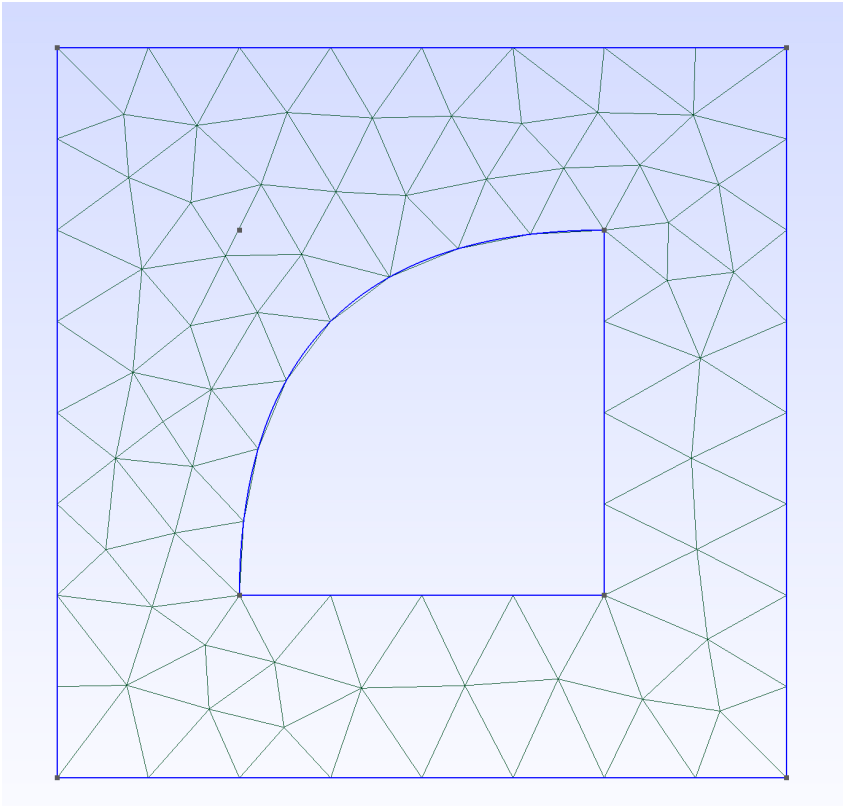


Performance comparison of the ADI scheme and the explicit Euler scheme for the two-dimensional heat equation:

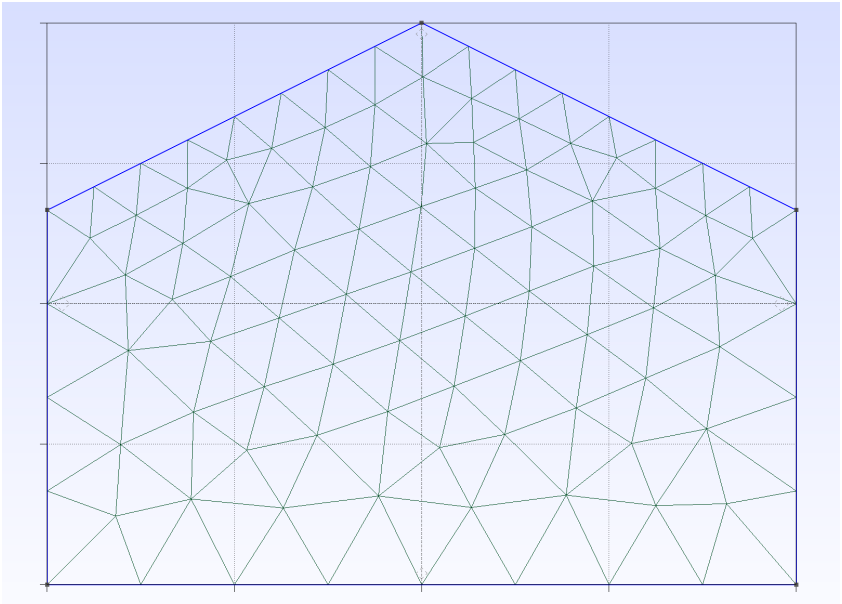


2.1

First example mesh:

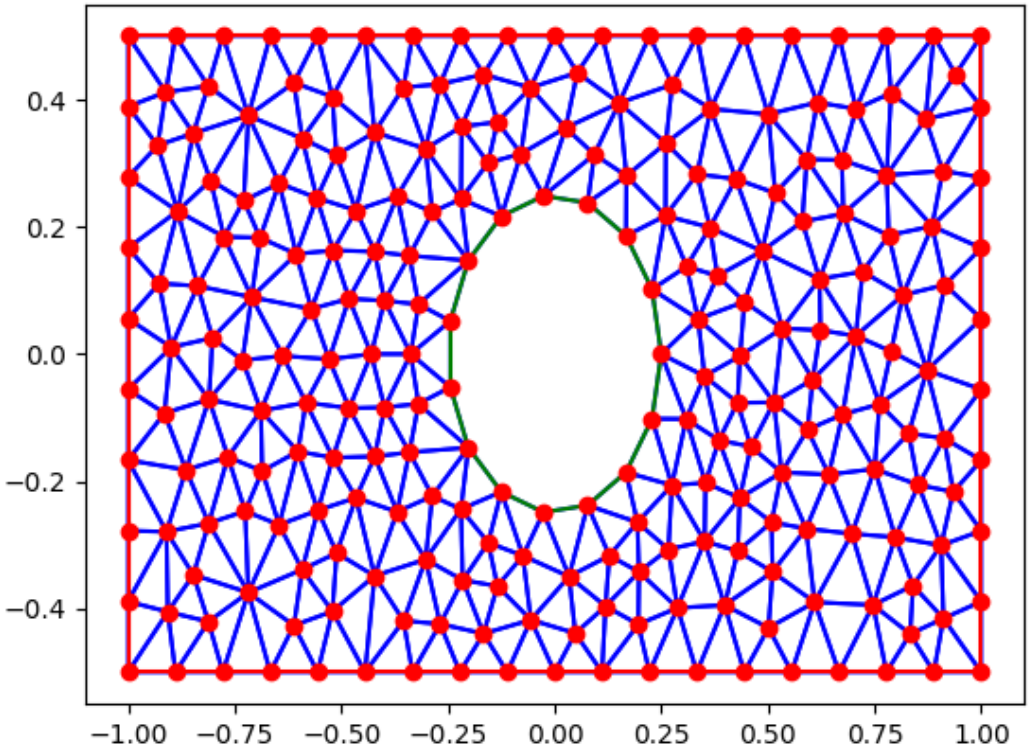


Second example mesh:

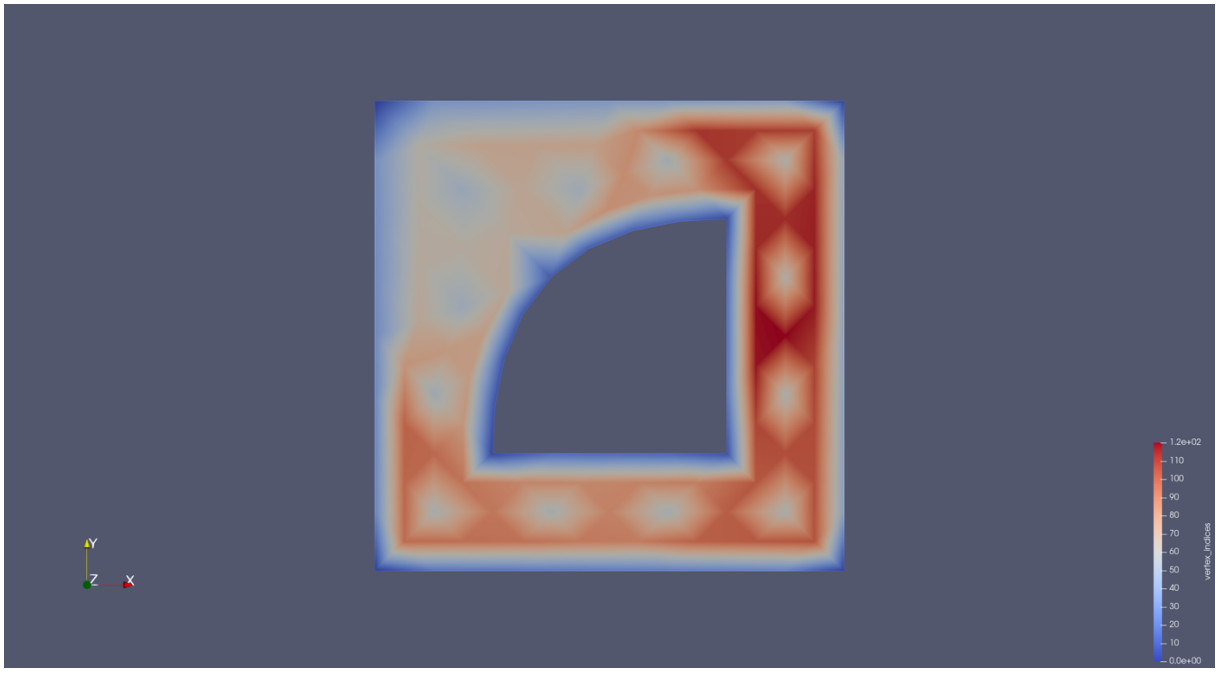


2.2

We used different colors for the outer and the inner boundary as well as for the mesh itself.

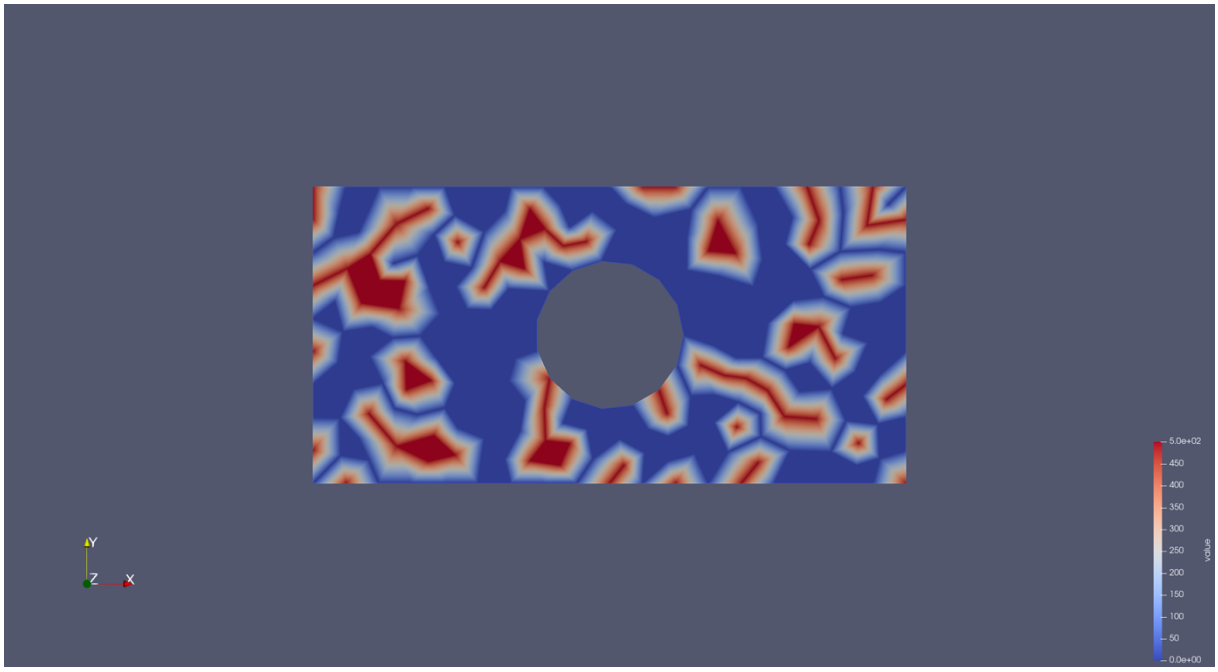


2.3

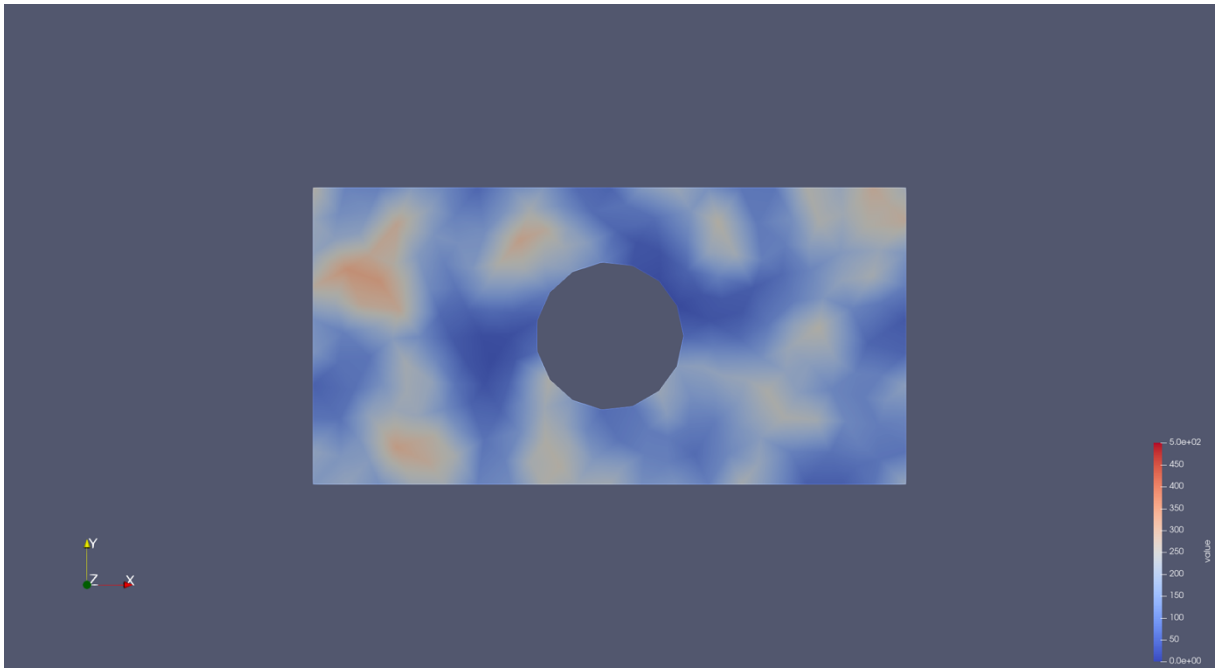


2.4

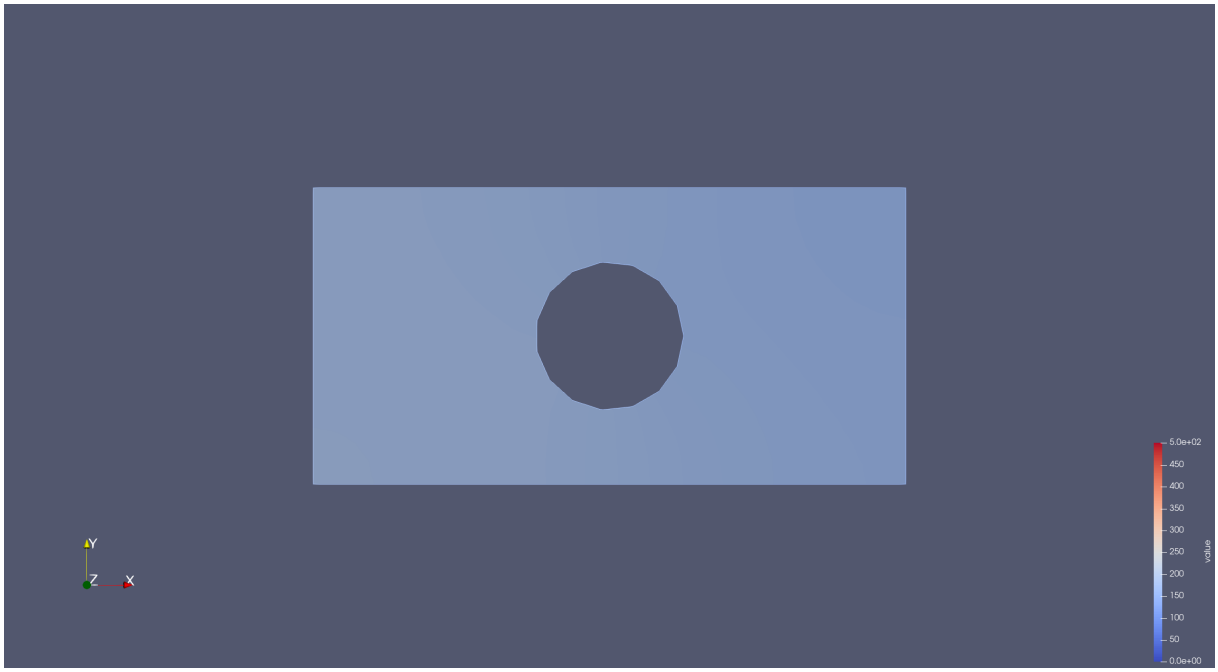
Visualization at the beginning:



Visualization after some iterations:



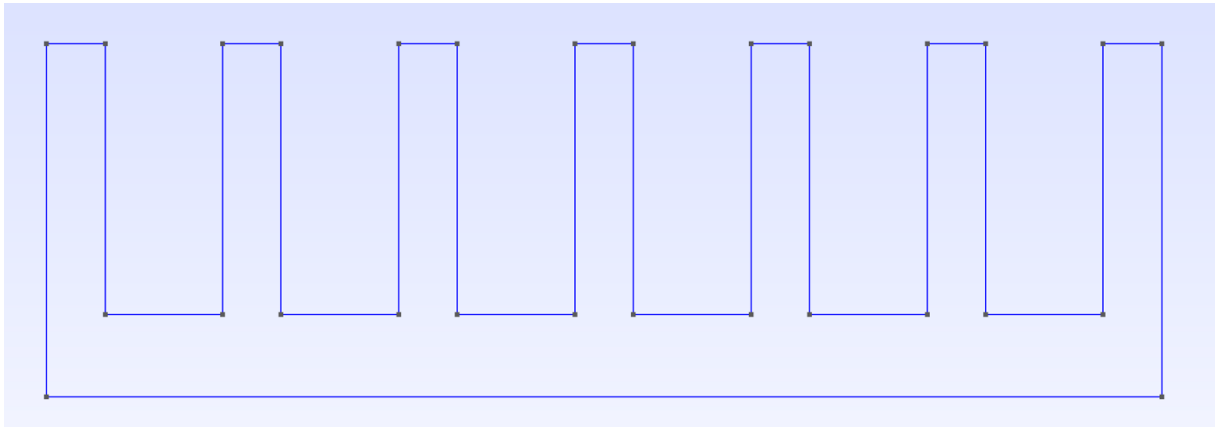
Visualization at the end:



When the python file for the simulation is executed, the implementation gets also checked by using the method of manufactured solutions. Therefore, the initial error and the final error is printed to the command line.

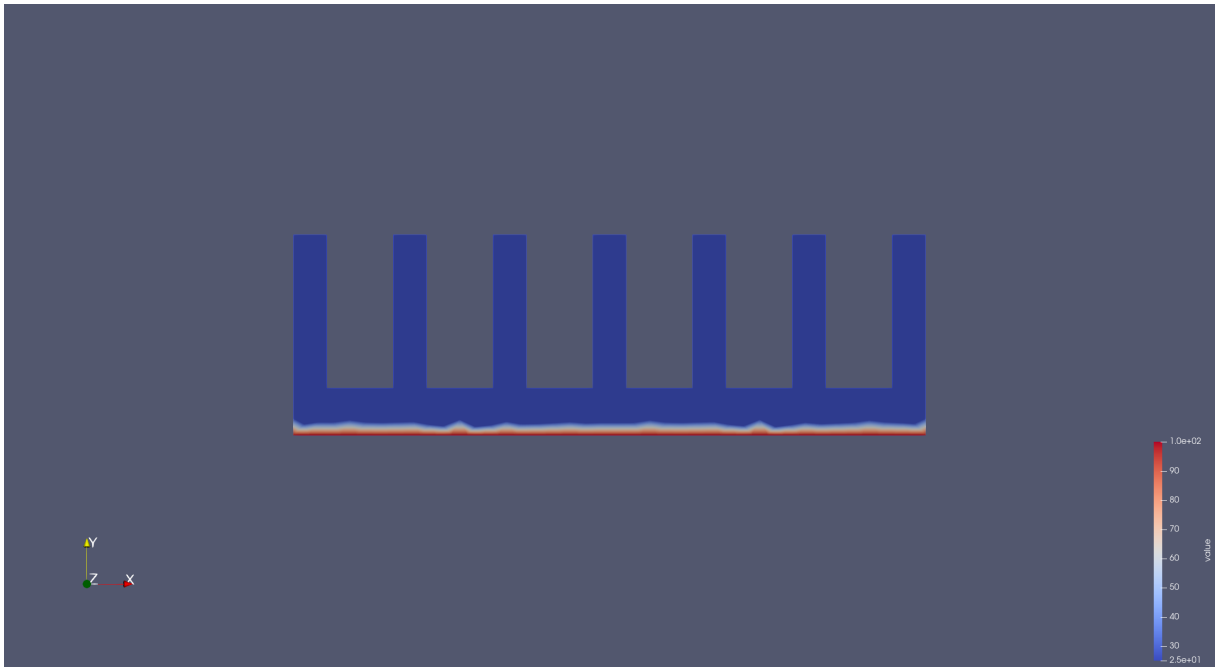
2.5

Used heat sink model:

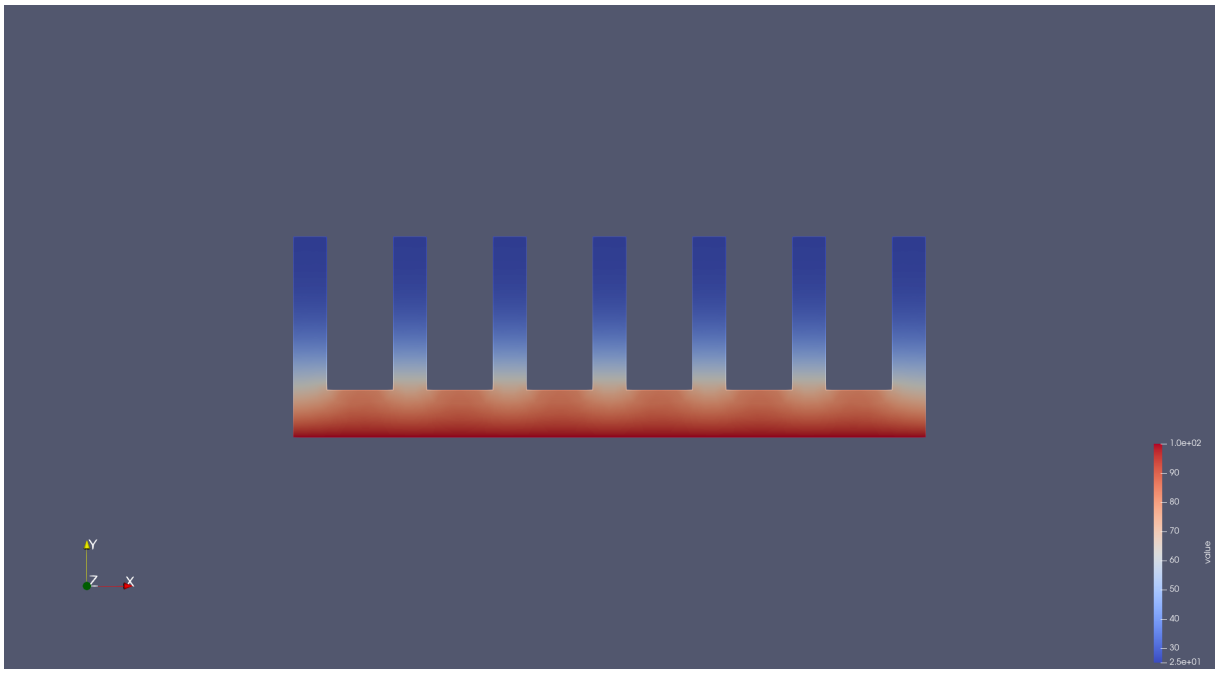


2.6/2.7

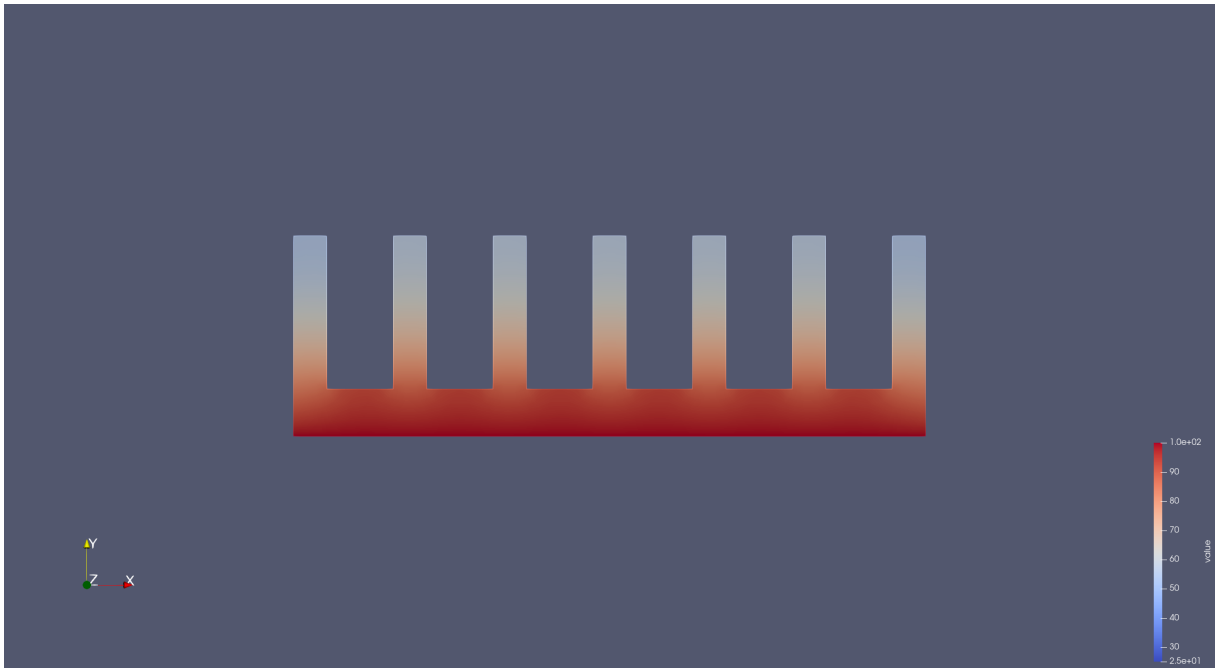
Visualization at the beginning:



Visualization after some iterations:



Visualization at the end:



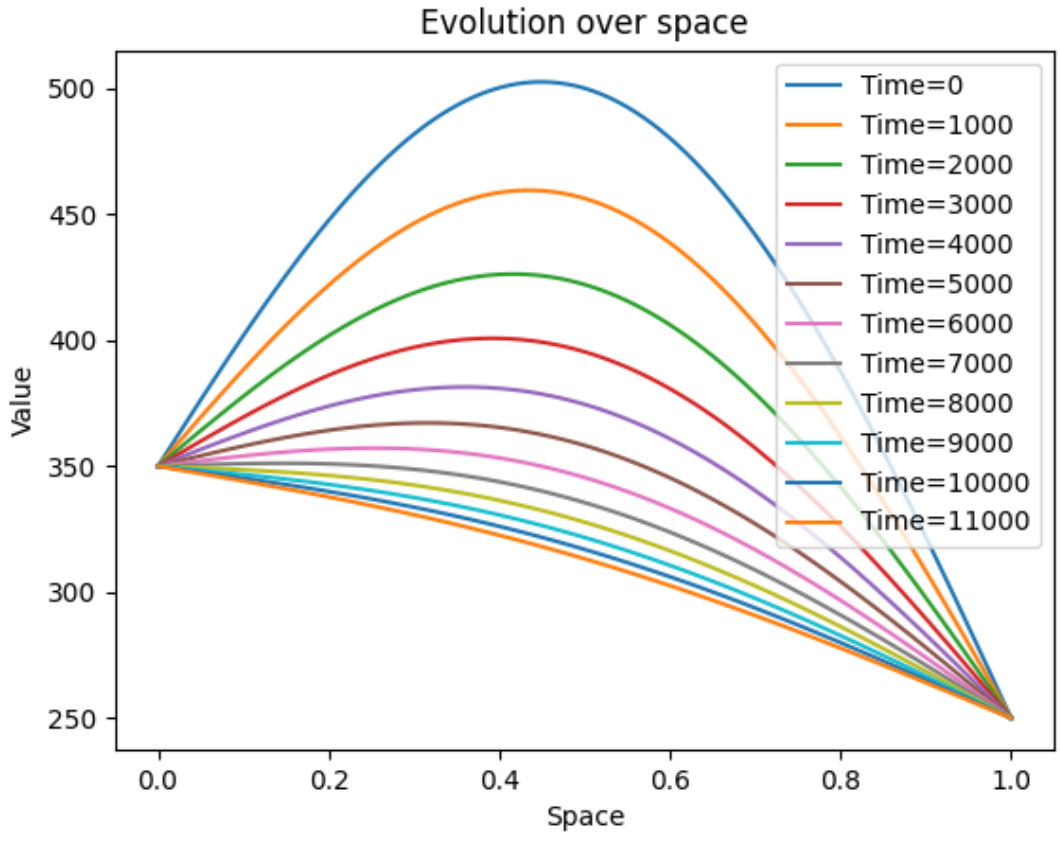
3.2/3.3

We implemented the explicit Euler centered difference scheme for the heat equation from exercise 1.5 in C++ and compared the performance of both implementations. The runtimes for the python and C++ implementation are:

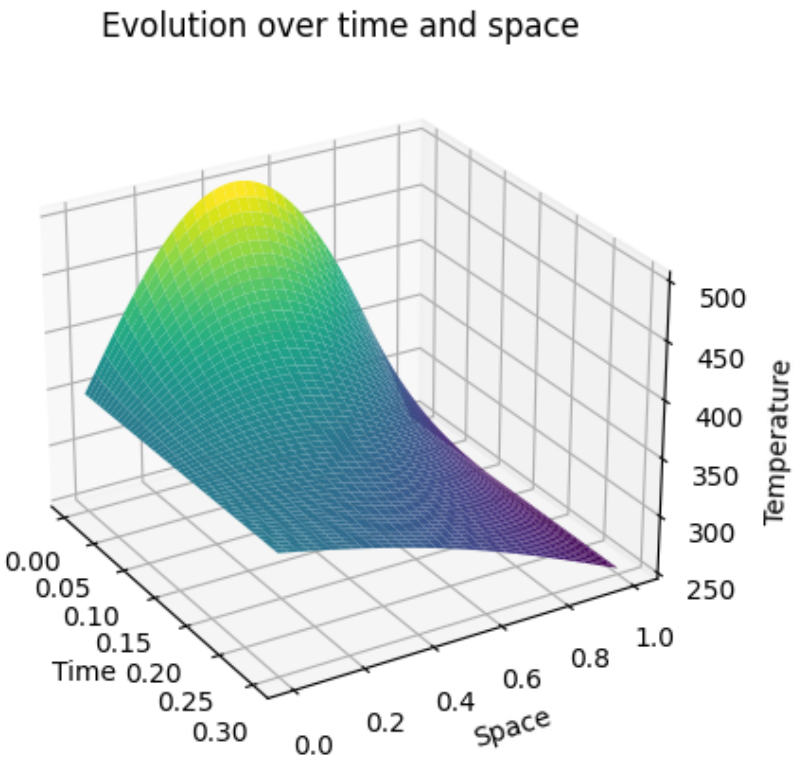
Python	C++
1322 ms	633 ms

We directly implemented the 3.2 in the way descripted in exercise 3.3, so 3.2 and 3.3 has the same implementation with no changes because of the already implemented periodical write to a text file. To plot the written data, there are two python implementations, one for plotting the data 2D and one for plotting it in 3D, which must be executed separately after the execution of the C++ code finished.

2D plot:



3D plot:



3.4

In this exercise we implemented the memory bound copy benchmark and the compute bound quadrature benchmark. We executed the benchmarks on the slightly different systems.

System 1:
MacBook Pro with M2 Pro chip with 16 GB RAM and 10 cores.

First the runtimes without optimization flag:

	Serial	1 Thread	2 Threads	4 Threads	8 Threads	10 Threads
Memory bound	2934 ms	2913 ms	1515 ms	885 ms	835 ms	785 ms
Compute bound	6494017 ms	6461 ms	3275 ms	1734 ms	1060 ms	985 ms

Runtimes with optimization flag -O3:

	Serial	1 Thread	2 Threads	4 Threads	8 Threads	10 Threads
Memory bound	993 ms	978 ms	846 ms	817 ms	763 ms	779 ms
Compute bound	0 ms	4850 ms	2475 ms	1305 ms	996 ms	760 ms

System 2:
MacBook Pro with M1 Pro chip with 16 GB RAM and 10 cores.

First the runtimes without optimization flag:

	Serial	1 Thread	2 Threads	4 Threads	8 Threads	10 Threads
Memory bound	3078 ms	3035 ms	1594 ms	927 ms	733 ms	716 ms
Compute bound	7323113 ms	7208 ms	3710 ms	1880 ms	1293 ms	1189 ms

Runtimes with optimization flag -O3:

	Serial	1 Thread	2 Threads	4 Threads	8 Threads	10 Threads
Memory bound	1037 ms	977 ms	766 ms	660 ms	604 ms	583 ms
Compute bound	0 ms	5794 ms	3108 ms	1544 ms	1187 ms	1329 ms

3.5

In this exercise we parallelized the explicit Euler scheme for the two-dimensional heat equation from exercise 1.12 using OpenMP.