1 Task 1

The first task is a rather straightforward one. One can directly adapt the functions given in tutorial four. The main difference lies in the given boundary conditions since they are mixed boundary conditions for both fluid/solid temperature and that one should solve for two quantities.

The chosen neural network architecture is a simple feed-forward network with five hidden layers and 100 neurons per layer. The network has two outputs where by arbitrary choice the first one is the fluid temperature and the second one is the solid temperature.

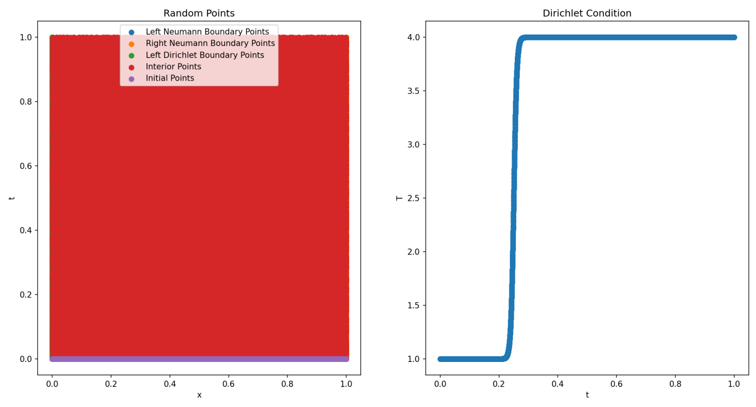

Figure 1: Sampled training data and the dirichlet boundary condition.

The loss function is constructed in the same way as in exercise four. The internal loss, the boundary loss and the initial loss are summed. The difference is that the boundary loss contains four sub-losses for the four boundary conditions. Therefore, one needs four training data sets for the boundary conditions.

The second order LBFGS method was used for optimization without deeper considerations. The amount of randomly sampled data was chosen to fully utilize the GPU memory. The amount of data is quiet excessive and is shown in Figure 1. In Figure 2 the loss change during training is presented. Here the training duration was chosen too long, as the loss goes into saturation.

In Figure 3 one can see the predicted fluid and solid temperature. One can see that the training of the neural network was not perfect since there are aberrations in both temperatures. On the left boundary, there is the region before the jump of the dirichlet condition where both temperatures should be at T_0 since there are no other heat sources, but temperature above T_0 is predicted. This error could not be fixed with hyperparameter tuning.

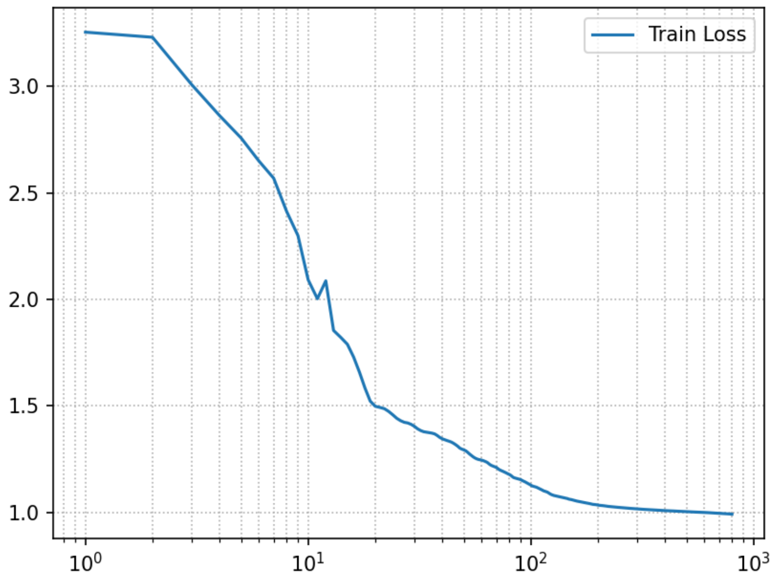

Figure 2: The training loss change.

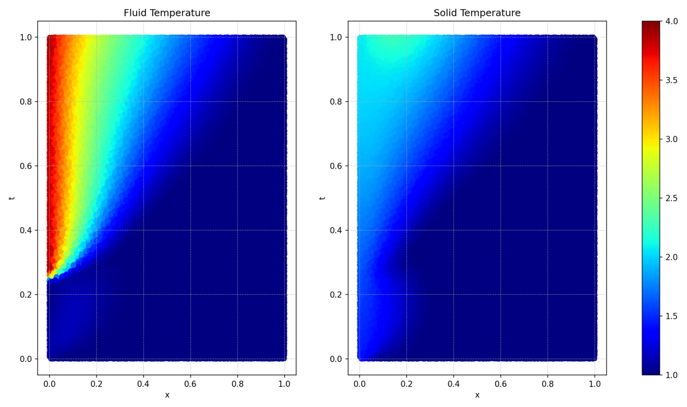

Figure 3: Preditections for fluid and solid temperature.

2 Task 2

In the second task, an inverse problem was solved. The solid temperature was treated as an unknown parameter and only the equation for the liquid temperature was solved. The solution presented is based on exercise five, in which the same PDE as in exercise four was solved, but as an inverse problem. The neural network architecture is the same as in exercise one, a fully connected feedforward network with two outputs for the temperatures. The main difference is that for the inverse problem, not only the unsupervised PDE/boundary loss is used, but a loss from the given samples is added. Also, the amount of unsupervised data had to be reduced so as not to overshadow the supervised data.

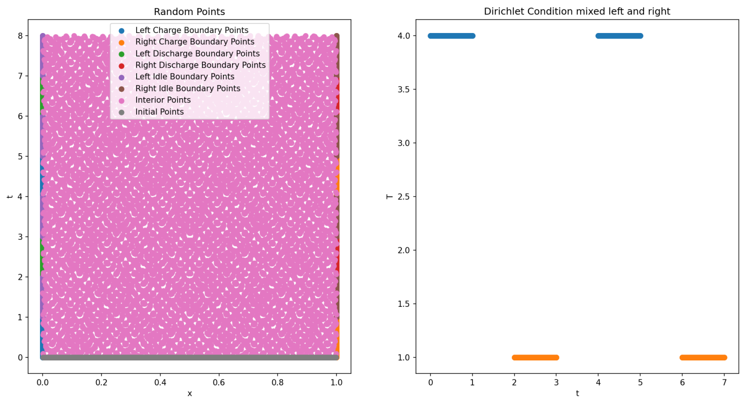

Figure 4: Sampled training data.

Another difference is that not only one phase was to be simulated. Therefore, time-dependent boundary conditions had to be applied. For this, one can simply sample each time interval independently for the boundary losses. As in task 1, Figure 4 shows the unsupervised training data and the applied Dirichlet constraints on the left and right edges. LBFGS was again used for optimization with manually set hyperparameters. No test-train splitting was performed because the PDE loss acted as a strong regularizer. Moreover, in Figure 5 the error while training can be seen.

Figure 6 shows the given samples and predictions. In general, the predicted fluid temperature agrees with the trend of the samples. Nevertheless, the sharp features of the given samples are not well predicted, which also happened in task one for the jump in the Dirichlet boundary condition. It is rather hard to verify the predicted solid temperature if one has no intuition about the expected results.

3 Task 3

In task 3, the problem is to predict the time evolution of the fluid/solid temperature at $x = 0$ only through one long measurement for times after this

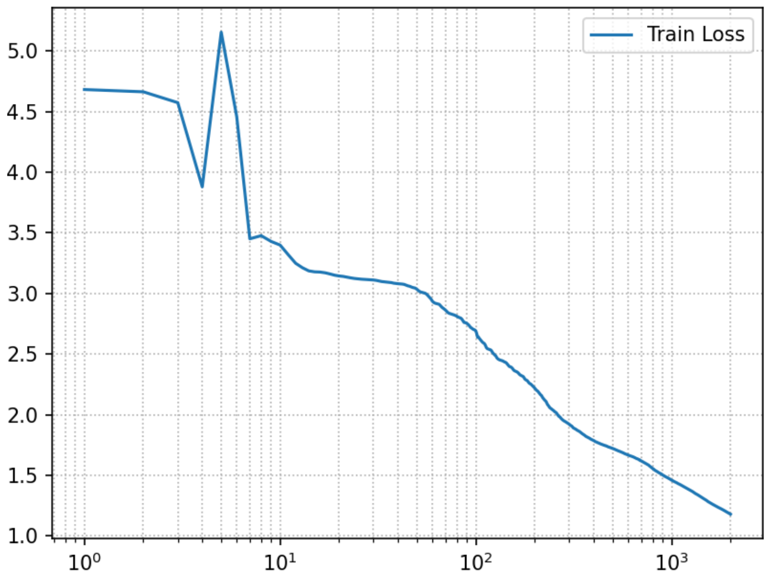

Figure 5: Loss change.

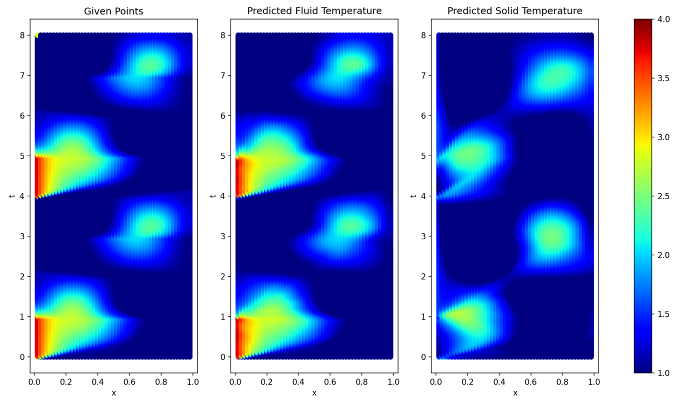

Figure 6: Given samples for the fluid temperature and predictions for both temperatures.

measurement. Operator learning should be used to solve the problem, but first the goal has to be rewritten to fit a machine learning problem.

The idea is to generate multiple supervised samples from the given measurement. From a given small number of fluid/solid temperature points the next point should be predicted. Then in the end, the network is applied iteratively to predict the time evolution, starting with the last few points of the long measurement.

This can be formulated in an operator learning frame work with a DeepONet. The input for the branch net are functions on a fixed time interval. The input for the trunk net are the time points where to predict the next point. In this case, it is only the next point. With this, the DeepONet can be trained to predict the next point in the time evolution. A tricky point is that the training functions are taken from one measurement and therefore not independent. One could argue how much this solution approach is true operator learning as presented in the lecture.

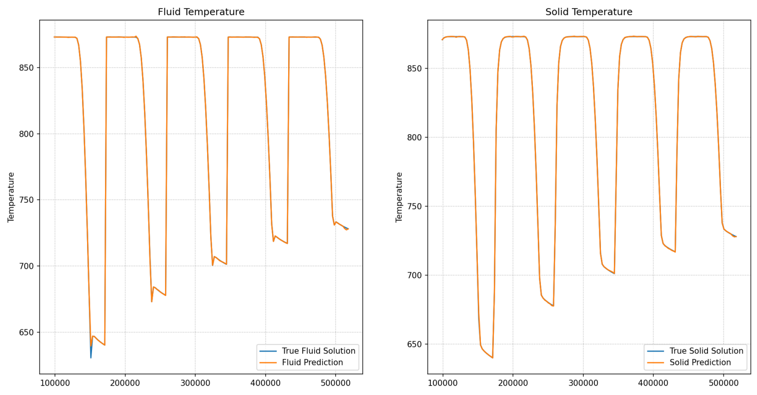

Figure 7: Prediction and given solution for the given time points.

In Figure 7 the predicted time evolution is shown. The predictions are well fitted to the inputs. Through a test-train split, one can see that the predictions are not overfitted.

In Figure 8 the predicted time evolution is shown. The predictions after 500000s are the future predictions and they match the trend of the previous time series with some aberrations.

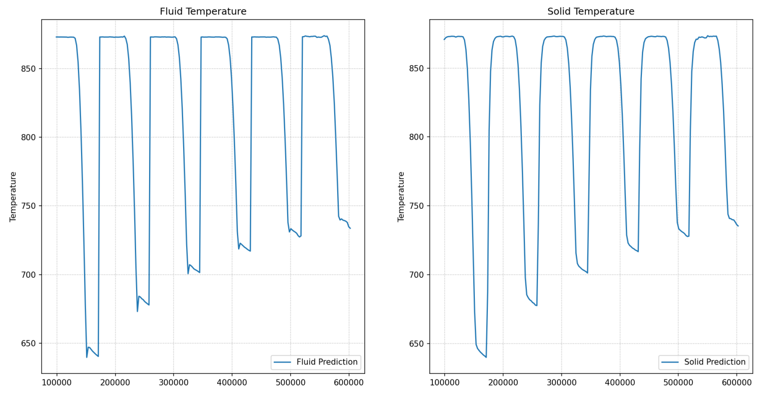

Figure 8: Prediction on the given time points and the future ones.