Report for Thermal Storage Design

Deep Learning in Scientific Computing Project

Angelo Gnazzo

June 12, 2022

Task 1: Noisy Function Approximation

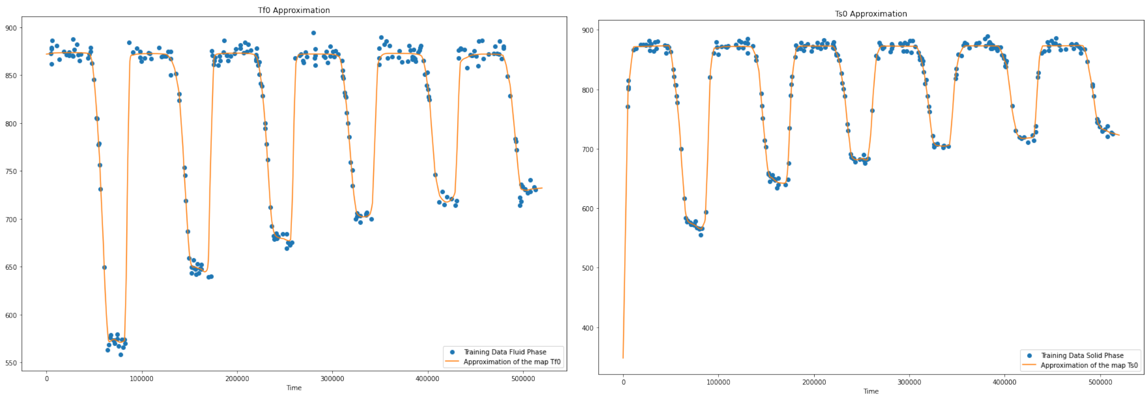
The goal of this task was to approximate the maps:

$$T_s^0 : t \mapsto T_s(0,t) \quad T_f^0 : t \mapsto T_f(0,t)$$

In this task noisy measurements of the fluid and solid temperature were provided. Since the elevate amount of noise, the data was denoised in the clouds of points in the area at high temperature for both the maps. All the data was normalized by Min-Max normalization in order to properly train the models. For both T_s^0 and T_f^0 different Feed Forward Fully Connected Neural Networks and other models such as XGBoost Regression, Random Forest and Gaussian Processes were tried, testing different hyperparameters via a grid search. For T_f^0, it was decided to perform an ensemble of an XGBoost model and a Neural Network, since both of them seemed to perform fairly well. On the predictions on the test data given by XGBoost, a smoothing via a Savitzky-Golay filter was applied, since the function produced by XGBoost was a bit discontinuous. For T_s^0 the same choice of models was carried out, (with different choices of hyperparameters) and an ensemble between a smoothed XGBoost model and a Neural Network was made again. For the latter it was noticed that both of the models in the ensemble were having some difficulties in learning the first part of the map, given the small amount of measurements present in the Training Data. For this reason, a reasonable solution was to perform, given the presumable shape of the function, a Linear Regression only on points within a first range of time indices and, on the remaining times, perform the ensemble described above. The resulting approximation maps for T_f^0 and T_s^0 are shown in Figure 1.

Task 2

The aim of this task was to learn the map from a set of several input variables to the an observable, the capacity factor. To solve this task, both Single-Level training and Multi-Level training were performed, with the latter one resulting the best model in terms of validation and training error. In the task, the original sequence of Sobol points and the transformed training sets generated at different mesh resolutions were given. It was noted that the Sobol points given were just the normalization of the input variables available in the Training Data files, so they were used to train the Neural Networks, given the fact that it is necessary to normalize the inputs of the Neural Networks to be able to efficiently training them. The training datasets were, then, created by



(a) T_f^0 **(b)** T_s^0

Figure 1: Training Data for T_f^0 and T_s^0 and respective approximation maps

concatenating the data at the coarsest resolution with the details at the other resolution levels, until the finest one. The training of all the building models of the multilevel scheme was then performed, after having carefully selected the hyperparameters for each one of the building models performing a grid search. A plot of the predicted CF values against the CF values given at the finest resolution with this Multi-level is visible in Figure 2.

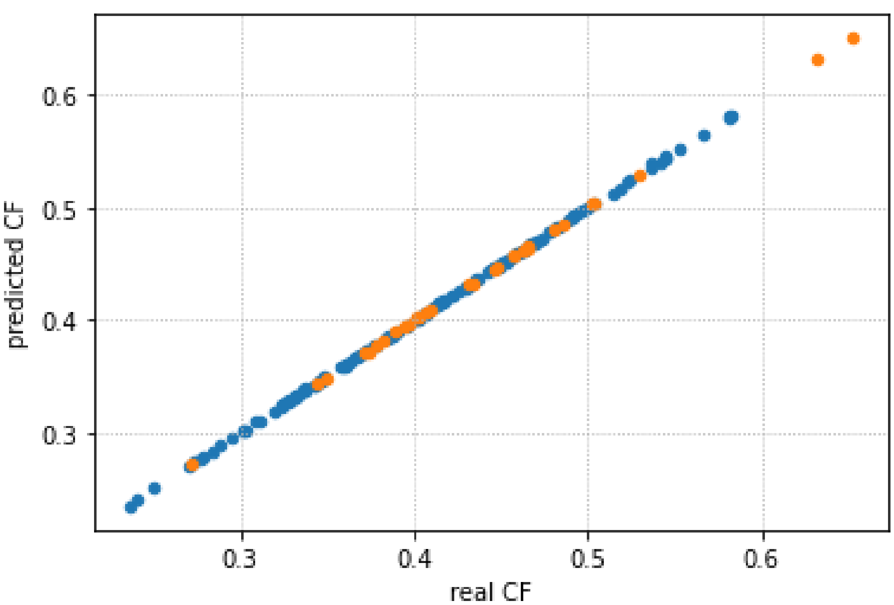
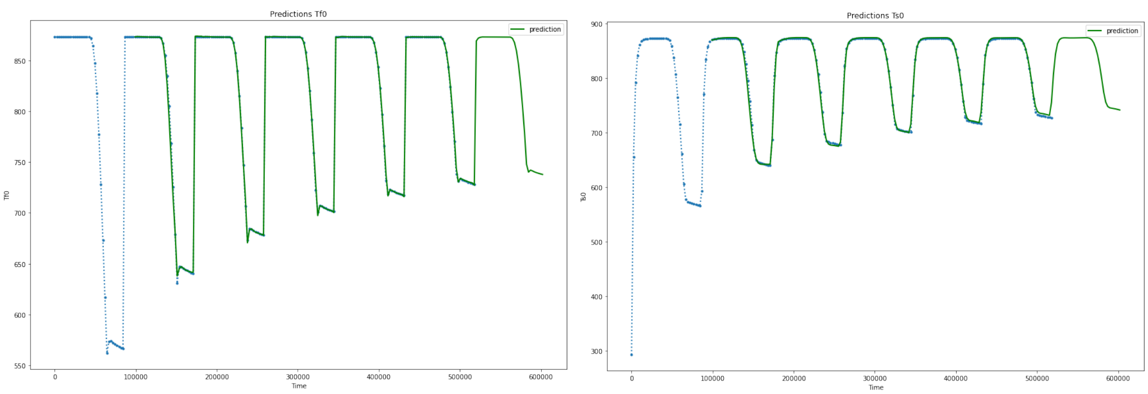


Figure 2: Comparison between the predicted CF values and the real CF values in the training set

Task 3

The goal of this task was similar to the one of Task 1 but here the object of interest was not only the approximation of the functions that describes the evolution of the fluid and solid temperature but the future predictions of those temperatures. Since the Neural Network are interpolating objects, in order to make predictions a new dataset was created in a way so that what it was given were consecutive temperature measurements for each starting time index of a fixed-length sequence of points. Then, the model was trained using the start of these sequences as input for the network and the end as responses. Similarly to Task 1, a first approach was to use a Feed-Forward Fully Connected Neural Network but it was noted that a Long Short-Term Memory Artificial Neural Network could perform better, if its hyperparameters were optimally tuned, which it was done perfoming a grid search on different parameters of the Neural Network. The predictions obtained for both the fluid and solid temperature are visible in Figure 3.



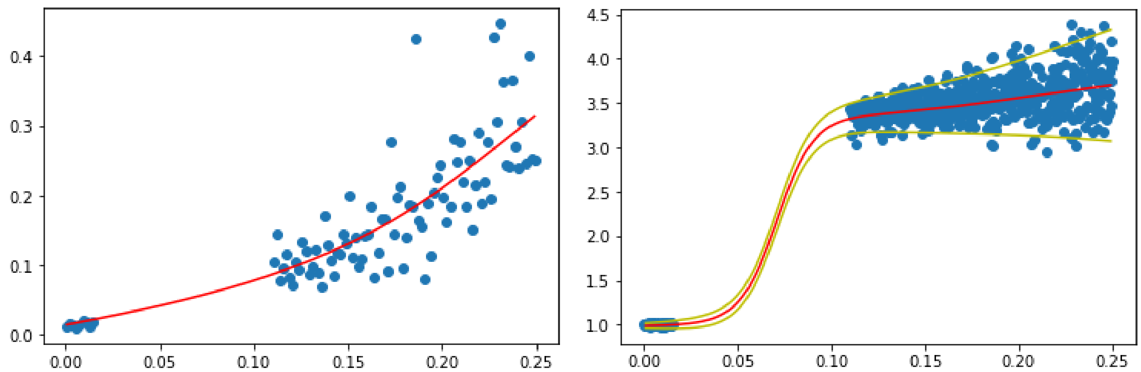
(a) T_f^0 **(b)** T_s^0

Figure 3: Training points and predictions of the LSTM models for T_f^0 and T_s^0

Task 4

Task 4.a

The goal of this task was to, given some observation of the standard deviation σ at different times, infer the map $\sigma = \sigma(t)$. The approach used to solve this problem was to divide the time domain of the given Measured Data in several intervals and, in each of these intervals, if it contained at least two observations, estimate the standard deviation (otherwise the standard deviation was set to zero). Then, a new dataset was constructed, consisting of the centers of said intervals as feature and of the estimated standard deviation as label. To approximate the map $\sigma = \sigma(t)$ a Feed Forward Fully Connected Neural Network was trained on the dataset constructed in the way explained above and its hyperparameters were chosen performing a grid search. To evaluate the accuracy of the inferred map, a Feed Forward Fully Connected Neural Network for the mean of the measurements in the Measured Data dataset was trained and it was verified empirically that the measurements available were close to the "68-95-99.7 rule", i.e. about 68% of the values were within one standard deviation of the mean, about 95% were within two standard deviations and about 99% were within three standard deviations from the mean.



(a) $\sigma(t)$ approximation

(b) 68-95-99.7 empirical rule

Figure 4: $\sigma(t)$ approximation map and mean approximation with deviation from $\sigma(t)$

Task 4.b

The objective of this task was to infer the reference velocity of the fluid u^* generating the recorded measurements. First, after having transformed the Training Data using Min-Max normalization, a Feed Forward Fully Connected Neural Network $T_{f,\theta}(t,u)$ was trained. Its hyperparameters were chosen performing a grid search and the best neural network was chosen in terms of the values of the training error, the validation error and the final training loss. With the selected model, the noisy Measured Data $(t_{1:N}, T^*_{f,1:N})$ were used to define the cost function

$$G(t_{1:N}, T^*_{f,1:N}, u) = \sum_{j=0}^{N} (T^*_{f,j} - T_{f,\theta}(t_j, u))^2$$

This cost function was minimized using Stochastic Gradient Descent as optimizer and the obtained result was also verified by creating a grid on equispaced values of the velocity, minimizing the cost function for every single one of these velocities and picking the one with the lowest value of the cost function.

Task 4.c

The purpose of this task was to perform Bayesian inference and provide 10000 samples drawn from the posterior distribution. In order to do so the following prior and likelihood were used:

$$P(u) = N(12, 4) \ ; \ P(T^*_{f,j}|t_j, u) = N(T_{f,\theta}(t, u), 0.075)$$

where $T_{f,\theta}(t,u)$ is the Feed Forward Fully Connected Neural Network used in Task 4.b to approximate the map $(t,u) \to T_f$. Then, following a Bayesian approach and making use of the Pyro library, the No-U-Turn Sampler was used to sample from the posterior distribution, setting 1000 iterations as burn-in period. The approximate posterior distribution is shown in Figure 5.

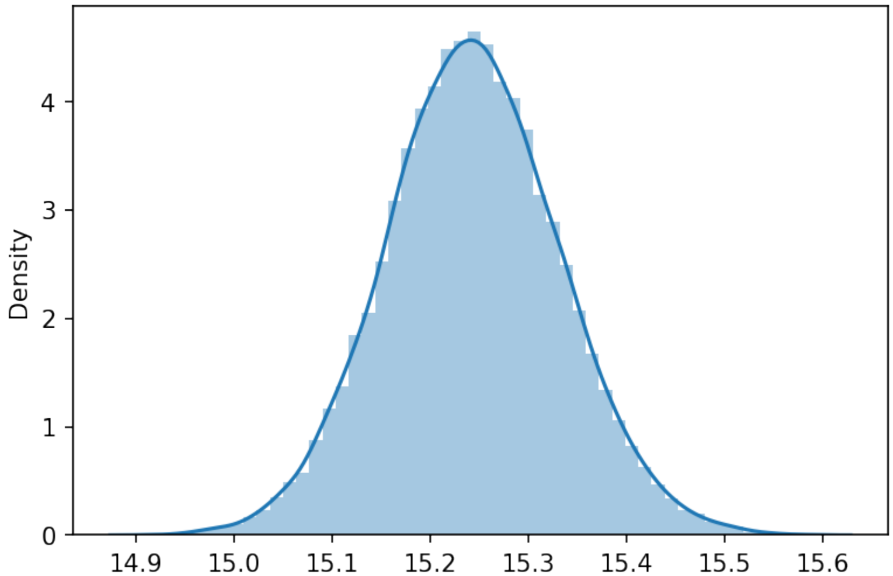


Figure 5: Approximate posterior distribution

Task 5

The goal of this task was to find the optimal design of the thermal storage. In order to solve the problem the DNNOPT algorithm was employed. As a first step, a Feed Forward Neural Network $CF_\theta(D, v)$ to approximate the map $CF(D, v)$. In order to do so, the control parameters (v, D) given in the Training Set have been normalized with a Min-Max transformation and, after a grid search with several possible parameters for the network, the optimal one was chosen using the value of the training error, the validation error and the final training loss as criteria. Then, to solve the minimization problem:

$$(D^*, v^*) = \text{argmin}_{D,v}\, G(CF(D, v))$$

the following cost function was used:

$$G(CF(D, v)) = (CF_{ref} - CF_\theta(D, v))^2 \tag{0.1}$$

This cost function was minimized using Stochastich Gradient Descent as optimizer and using 1000 equispaced starting points for D and v, with the points of the latter that were optimized. Since it was noted that the curve of optimal points varied slightly for different parameters of the Neural Network, an ensemble of 5 Neural Networks with similar performances was done. The curve of optimal points in this way is shown in Figure 6.

Moreover, another naive approach was implemented, consisting in solving the problem in a more classical numerical analysis way. Two equispaced grids for D and v were created and, for each point of the grid of D, it was choosen the value of v which minimized the difference, in absolute value, of the prediction given by $CF_\theta(D, v)$ with those two inputs and CF_{ref}. This approach was used to verify the goodness of the results obtained with the first method.

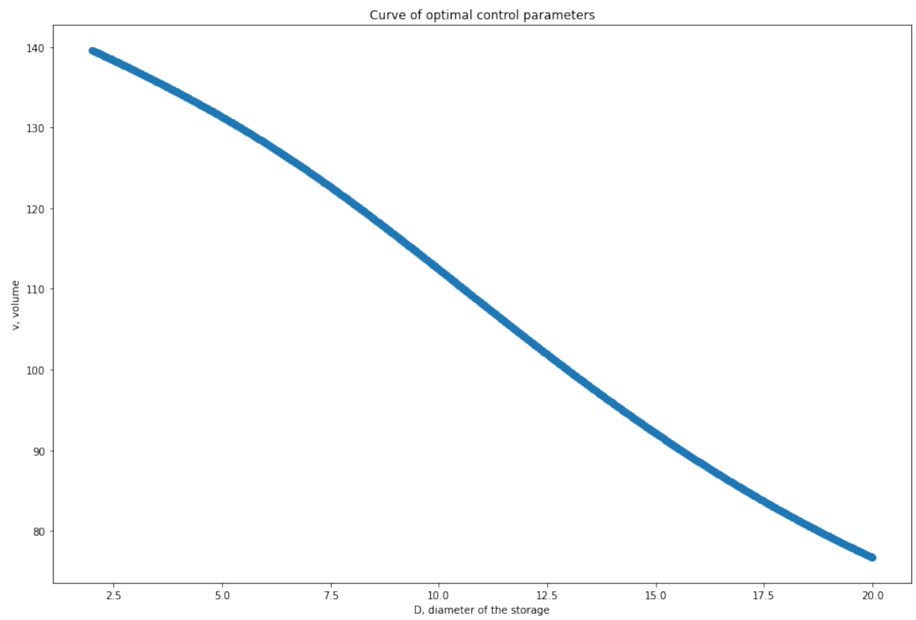


Figure 6: Optimal curve of (D^*, v^*) values.

Task 6

In this task the aim was to solve the following system of equations:

$$\begin{aligned} \frac{\partial \bar{T}_f}{\partial t} + 1\frac{\partial \bar{T}_f}{\partial x} &= 0.05\frac{\partial^2 \bar{T}_f}{\partial x^2} - 5\left(\bar{T}_f - \bar{T}_s\right) \quad x \in [0,1], t \in [0,1] \\ \frac{\partial \bar{T}_s}{\partial t} &= 0.08\frac{\partial^2 \bar{T}_s}{\partial x^2} + 6\left(\bar{T}_f - \bar{T}_s\right) \quad x \in [0,1], t \in [0,1] \end{aligned}$$

with initial and boundary conditions as:

$$\bar{T}_f(x, t=0) = \bar{T}_s(x, t=0) = 1 \ x \in [0,1]$$

$$\left.\frac{\partial \bar{T}_s}{\partial x}\right|_{x=0} = \left.\frac{\partial \bar{T}_s}{\partial x}\right|_{x=1} = \left.\frac{\partial \bar{T}_f}{\partial x}\right|_{x=1} = 0 \ t \in [0,1],$$

$$\bar{T}_f(x=0, t) = \frac{3}{1+\exp(-200(t-0.25))} + 1, \quad t \in [0,1].$$

The task was solved with Physics Informed Neural Networks. Following the PINNS tutorial, several changes were implemented in order to adjust the code to the system of equations above, such the definition of the functions *add_boundary_points*, *compute_pde_residuals* and so on. Two possible solutions with a two-outputs neural network and two distinct neural networks were considered, which turned out to be pretty much equivalent and for computational reasons, the first option was chosen, given the fact that it trained faster than the latter. The loss function to minimize was defined as the sum of the loss computed at the spatial boundaries, the one at the temporal boundary and the interior loss. Each of these were computed as the L_2 norm of the corresponding residuals. The result obtained is the one visible in Figure 7.

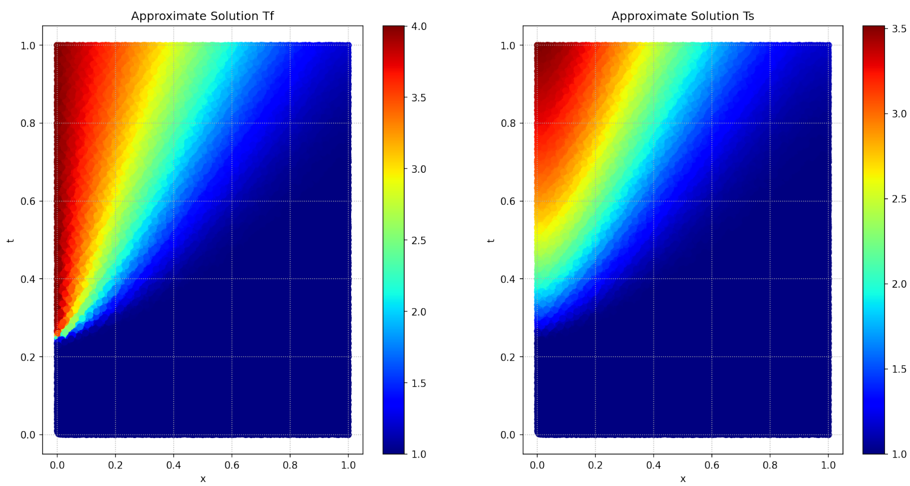


Figure 7: Optimal curve of (D^*, v^*) values. The initial Sobol points are also shown with the corresponding G value