CARNEGIE MELLON UNIVERSITY

06-665 PROCESS SYSTEMS MODELING

PROJECT REPORT

Modeling and Simulation of a CSTR

Advised by:
Prof. Fernando V. Lima

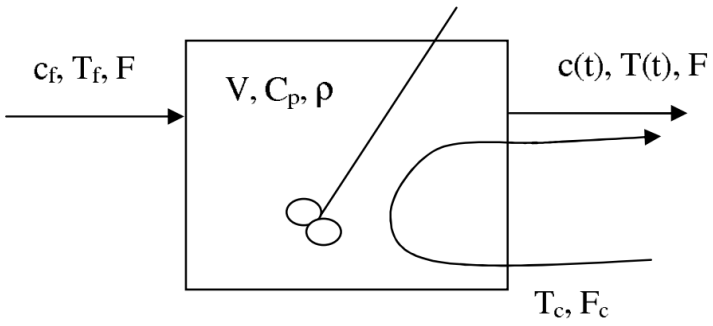
Group 2
Authors:
Yuanyi HUANG
Sheena KAPOOR
Chang CHEN
Ambrish ABHIJNAN

April 25, 2022

1 Project Summary

Neural networks is a little subfield of machine learning. They mimic the human brain through a set of algorithms. At a basic level, a neural network is comprised of four main components: inputs, weights, a bias or threshold, and an output [1]. More specifically, the study in this paper is about Multi-Layer Perceptron (MLP), a kind of classical Neural Networks (NNs), which solves the regression problem for non-linear sets by employing hidden layer. In this project, the study considers the application of MLP to an experimental pilot plant reactor apparatus. It involves a continuous stirred tank reactor (CSTR) with a hydraulic stirring system[1]. A CSTR is a reaction vessel in which reagents, reactants and often solvents flow into the reactor while the product(s) of the reaction concurrently exit(s) the vessel. In this manner, the tank reactor is considered to be a valuable tool for continuous chemical processing [2].A CSTR assumes ideal mixing and thus, is the complete opposite of a Plug Flow Reactor (PFR) [2] because it operates on a steady-state basis, where the conditions in the reactor don't change with time. This whole project includes four main parts:

1. Simulating ordinary differential equation system (ODEs) of the CSTR system,
2. Calculating the discrete time state-space form for ODEs,
3. Building the step response model based on the discrete time state-space representation,
4. Utilizing the deep Multi-Layer Neural Network model to simulate ODEs.



Figur 1: Sketch of Hick's Reactor [3]

As shown in figure 1, the liquid-phase tank reactor does not have a jacket, and does not have chemical reactions happened during stirring, the main changes be studied are the trend of concentration and temperature with changing of flow rate and heat transfer rate. This would be discussed later in the report. This CSTR system is from one of Professor Lorenz T. Biegler's research study cases about non-linear model predictive control (NMPC) optimization simulation, which was originally simulated in GAMS. However, in this project, only the ODE system and constant parameters were referred. Since the ODE system is

nonlinear, MNN simulation would be focused on instead of NMPC. Python was the main tool used to build and simulate the whole system, and the machine learning part would include some of the optimization comparing with the results from the original ODE system. This would be discussed later in the report as well.

2 Objectives

This project aims to model and simulate an ODE system for a Non-Isothermal Continuous Stirred Tank Reactor. We aim to perform all simulations in Python. For the purpose of this project, we are focusing on Hick's Reactor and obtaining the relevant Ordinary Differential Equations from Professor Lorenz T. Biegler's work [3].

The profiles of the ODEs are observed with the function solve_ivp. The continuous ODE model is converted to discrete-time state-space representation using the function scipy.signal.cont2discrete. This state-space model is used to obtain the step response model. This is done using the function scipy.signal.dstep. Plots are then created to for outputs of each input. It is important to note that there are two inputs and two outputs for the ODE system of interest.

Following this, Deep Learning simulations are performed using a Multilayer Perceptron Model. This MLP Neural Network model is used to represent the ODE system as a predictive ODE model. This Neural Network model compares with the solve_ivp model from the initial conditions to the steady state.

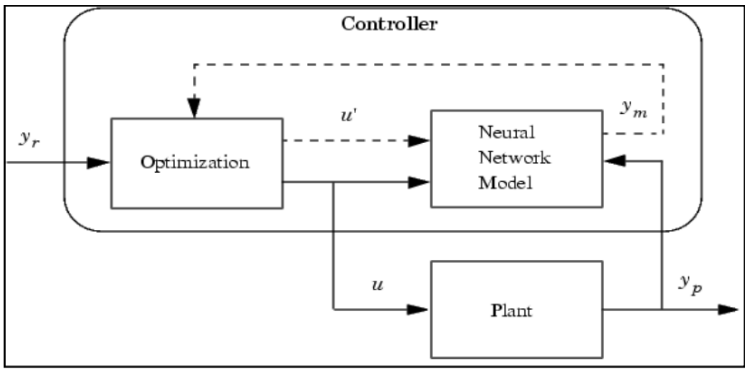
3 Literature Review

This report aims to determine the optimal control policy for a continuous stirred tank polymerization reactor, commonly referred to as Hick's reactor. The control policy involves determining optimal temperature and flow rate that brings the reactor to a steady- state operating condition. In addition, one can further optimize the reactor by minimizing the objective functions aimed to reduce the start-up time, cost of operations, etc. The objective of designing a steady-state Hick's reactor is to determine optimal steady- state controls [5]. The mathematical models available for steady- state control problem is non- linear programming problem. Non- uniform mixing of free radicals remains the biggest issue while optimizing the reactor. Moreover, studies have been conducted to determine the optimal conditions for a wide range of products. The reactor must be tuned to operate at one steady- state to another to meet the market demand. The work conducted by Hicks et. al. highlights the hindrances to the solution includes dearth of mathematical models for industrial systems that fails to explain mixing effects and kinetics involved in free radical polymerization [5]. In addition, lack of studies has been conducted for highly exothermic reactions involved in polymerization systems due to unstable system. For all the mathematical model developed it is assumed reaction to be homogeneous poly-condensations and free- radical polymerization

for simple reaction mechanics.

The classical control methodology is dependent on precise mathematical models that in general perform poorly with uncertainties and disturbance involved into the control system. Intelligent method, an alternative to the classic control methods have been introduced are being extensively used that don't require mathematical models and provide flexibility against disturbances and non-linearities in the model [6]. The nonlinear behaviour in process control may occur due to temperature dependence on rate of reactions, change in chemical composition, valve openings or manipulation in flow rates or other physical constraints. Recently, the application of neural networks for nonlinear system has been proved to be efficient. Furthermore, combination of neural networks and model- based predictive controls have showed excellent results achieving precision control [7]. The NN based nonlinear predictive control algorithm consists of system identification of nonlinear models and apply optimization based algorithm to generate precision control minimizing functions responsible for current and predictive errors [8]. The system identification algorithm can be further divided into parametric and non-parametric model identification, in other words, into online and offline mode for identification [8]. Least square algorithm along with gradient correction procedure and maximum likelihood methods are the three algorithms that are widely used for system identification. In general, least square parametric identification algorithm is extensively used for nonlinear control CSTR. Furthermore, we need to train our neural network for predicting the future values, analyzing past data for input and corresponding output. The working of artificial neural networks model can be explained as every neuron receives information from neurons present in the preceding layer, multiplied by individual weights. The summation of the weighted inputs are passed through the function scaled output [9]. The neural network model predicts the plant response over a period of time, used to obtain control signal eventually minimizing the performance function over a particular time horizon [10]. Figure 2, illustrates the process of model predictive control in a form of a block diagram. A controller consists of a optimization function with a neural network model. The optimizer (block) determines the value of u', and u, control signal, acts as an input to the plant, which determines the desired output y_p. In addition, y_m is the NN model output that is fed back to the optimization block to determine new u'.

One of the key aspect for designing robust control structure incorporating neural networks is to acknowledge the neural network structure and developing relation between process data and parameters including weights and biases. Recently, noble algorithms have been developed control algorithms which incorporates simultaneous learning and control without need for system identification. Many studies show that Levenberg- Marquardt algorithm, a second- order derivative based algorithm performed better against steepest descent, a typical first order derivative based algorithm [8].



Figur 2: Block Diagram for NN Predictive Control [9]

4 Methods

4.1 ODE System

The liquid phase stirred tank reactor is shown in Figure 1. Assuming constant liquid volume (V), density (ρ), heat capacity (C_p), coolant temperature (T_c), coolant flow (F_c) and heat of reaction (ΔH), as well as a first order Arrhenius rate law [3]. Based on the structure in Figure 1, the mass and energy balance for the reactor can be built:

$$V\frac{dc}{dt} = F(c_f - c(t)) - Vk_{10}exp(-E/RT)c(t), c(0) = c_{init} \tag{1}$$

$$\rho C_p V\frac{dT}{dt} = F\rho C_p(T_f - T(t)) + \Delta HVk_{10}exp(-E/RT)c(t) + U(F_c)A_c(T_c - T), T(0) = T_{init} \tag{2}$$

To simplify the ODE system, several constant terms were aggregated:

$$\theta = \frac{V}{F}, \alpha u = \frac{UA_c}{\rho C_p V}, y_f = \frac{\rho C_p T_f}{\Delta H} \tag{3}$$

$$n = \frac{E\rho C_p}{R\Delta H}, y_c = \frac{\rho C_p T_c}{\Delta H} \tag{4}$$

And then redefine the states:

$$c \leftarrow \frac{c}{c_f}, T \leftarrow \frac{\rho C_p T}{\Delta H} \tag{5}$$

So that the ordinary differential system becomes:

$$\frac{dc}{dt} = \frac{1 - c(t)}{\theta} - k_{10}exp(-E/RT)c(t), c(0) = c_{init} \tag{6}$$

$$\frac{dT}{dt} = \frac{(y_f - T(t))}{\theta} + k_{10}exp(-E/RT)c(t) + \alpha u(y_c - T), T(0) = T_{init} \tag{7}$$

4.2 Discrete-time State-space Form

The discrete-time state-space representation is a mathematical model of a physical system as a set of input, output and state variables related by differential equations. State variables are variables whose values evolve over time in a way that depends on the values they have at any given time and on the externally imposed values of input variables. Output variables' values depend on the values of the state variables. Based on the ODE system shown in last section, the states, input, and output variables can be determined:

$$x = y = \begin{bmatrix} c \\ T \end{bmatrix} \tag{8}$$

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\theta} \\ u \end{bmatrix} \tag{9}$$

The states x equals to the outputs y, which both include to variables: the concentration ratio c, and the system temperature term T; and the outputs u includes two variables as well: u_1 equals to the inverse of θ and u_2 equals to the heat transfer rate u. Rearrange the differential equations with input variables u_1 and u_2:

$$\frac{dC}{dt} = (1-c)u_1 - k_{10}\exp(\frac{-n}{T})c \tag{10}$$

$$\frac{dT}{dt} = (y_f - T)u_1 - k_{10}\exp(\frac{-n}{T})c + \alpha u_2(y_c - T) \tag{11}$$

We determine matrix A,B,C and D, which are linear representation of the above original nonlinear model and can be expressed using Taylor series expansion about the state and input yield, neglecting higher order derivatives.

At steady- state condition, we have:

$$y = g(x_s, u_s) + \frac{\partial g(x,u)}{\partial x}|_{x_s,u_s}(x - x_s) + \frac{\partial g(x,u)}{\partial u}|_{x_s,u_s}(u - u_s) \tag{12}$$

Moreover, $y_s = g(x_s, u_s)$

$$\dot{x'} = Ax' + Bu' \tag{13}$$

$$y' = Cx' + Du' \tag{14}$$

where $y' = y - y_s$, and $u' = u - u_s$ are the variables that represent deviation.

For linearization, the elements of the matrices can be defined as follows:

$$A_{ij} = \frac{\partial f_i}{\partial x_j}|_{x_s,u_s}, B_{ij} = \frac{\partial f_i}{\partial u_j}|_{x_s,u_s} \tag{15}$$

$$C_{ij} = \frac{\partial g_i}{\partial x_j}|_{x_s,u_s}, D_{ij} = \frac{\partial f g_i}{\partial u_j}|_{x_s,u_s} \tag{16}$$

Here, subscript ij represent the i^{th} row and the j^{th} column respectively.

$$A_{11} = -k_{10}\exp(\frac{-n}{T}) - u_1, A_{12} = -\frac{ck_{10}n\exp(-\frac{n}{T})}{T^2} \tag{17}$$

$$A_{21} = -k_{10}\exp(\frac{-n}{T}), A_{22} = -\frac{ck_{10}exp(\frac{-n}{T})}{T^2} - \alpha u_2 - u_1 \tag{18}$$

$$B_{11} = 1 - c, B_{12} = 0 \tag{19}$$

$$B_{21} = -T + y_f, B_{22} = \alpha(-T + y_c) \tag{20}$$

Sl. No:	Notation	Parameters	Values
1	C_{init}	Initial Concentration	0.1367
2	T_{init}	Initial Temperature	0.7293
3	U_{init}	Initial Cooling Water	390
4	C_{des}	Final Concentration	0.0944
5	$U_{1_{des}}$	Final Inverse Flow Rate	0.05
6	$U_{2_{des}}$	Final Heat Transfer Rate	340
7	T_{des}	Final Temperature	0.7766
8	α_1	Dimensionless Number	1×10^6
9	α_2	Dimensionless Number	1×10^6
10	α_3	Dimensionless Number	1×10^-3
11	α	Dimensionless Number	1.95×10^-4
12	k_{10}	Rate Constant	300
13	n_{cp}		3
14	y_c		0.3816
15	y_f		0.3947
16	n_{fe}		100

Table 1: Constant Parameters Table

Organizing the calculated results above, substitute the values of constant variables (see Table 1), we can get A, B, C, and D:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} -0.52972684 & -0.37544061 \\ 0.47972684 & 0.25914061 \end{bmatrix} \tag{21}$$

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} 9.0560e-1 & 0 \\ -3.8190e-1 & -7.7025e-5 \end{bmatrix} \tag{22}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{23}$$

Then follow the equation below or using function scipy.signal.cont2discrete in Python:

$$\Phi = \exp A\Delta t \tag{24}$$

$$\Gamma = [\exp A\Delta t - I]A^{-1}B \tag{25}$$

The value of Φ and Γ can be found:

$$\Phi = \begin{bmatrix} 0.51965315 & -0.32659319 \\ 0.41731107 & 1.20588349 \end{bmatrix}, \Gamma = \begin{bmatrix} 7.46126117e-1 & 1.31923405e-5 \\ -2.26312395e-1 & -8.56172099e-5 \end{bmatrix} \tag{26}$$

The matrices for C and D remain the same since x = y.

4.3 Step Response Model

The First Step Response models are determined by constructing a unit step input change to process operating steady- state. The coefficients of the models are the outputs at every time step. Moreover, S_i represents the step response coefficient for the i^{th} time, once unit change is determined. The step response model is considered to be a vector of step response coefficients and can be represented as follows:

$$S = [s_1, s_2, s_3, s_4.....s_N]^T \tag{27}$$

Here, the length of the model is N, which is sufficiently long such that the process is at or almost at steady- state condition.
Let us consider our system to be at $t_k = 0$ and having 1 step, $\Delta u(0)$:
$y(0) = 0$
$y(1) = s_1\Delta u(0)$
$y(2) = s_2\Delta u(0)$
$y(3) = s_3\Delta u(0)$
.
.
.
$y(k) = s_k\Delta u(0)$

.
.
.
$y(N) = s_N \Delta u(0)$

If we consider an another step input at $t_k = 1$ then the change in output, $\Delta u(1)$ can be determined by applying the idea of super-position as follows:

$y(0) = 0$
$y(1) = s_1 \Delta u(0)$
$y(2) = s_2 \Delta u(0) + s_1 \Delta u(1)$
$y(3) = s_3 \Delta u(0) + s_2 \Delta u(1)$
.
.
.
$y(k) = s_k \Delta u(0) + s_{k-1} \Delta u(1)$

similarly, for $y(N)$, we obtain as follows, we obtain:

$$y(N) = s_N \Delta u(0) + s_{N-1} \Delta u(1) \tag{28}$$

For a system with multiple steps at $k-1$, $k-2$,.... $k-N$,...$k-\infty$
From the super-position of all the previous input steps results as follows:

$$y(k) = s_1 \Delta u(k-1) + s_2 \Delta u(k-2) + ... + s_{N+1} \Delta u(k-N-1) + ... + s_{N+\infty} \Delta u(k-\infty) \tag{29}$$

In addition, if we assume that the step response coefficients are greater than that of step N, i.e. steady- state condition. The above equation reduces to following equation as:

$$y(k) = s_1 \Delta u(k-1) + s_2 \Delta u(k-2) + ... + s_N \Delta u(k-N) + s_{N+1} \Delta u(k-N+1) + s_N u(k-N) \tag{30}$$

The reduced equation is in form as:

$$y(k) = \sum_{i=1}^{\infty} s_i \Delta u(k-i) \tag{31}$$

$$y(k) = S_N u(k-N) + \sum_{i=1}^{N-1} s_i \Delta u(k-i) \tag{32}$$

4.4 Deep Learning Simulation for ODE System

4.4.1 Data Generation

First, we evenly pick 500 points between the lower and upper bound and for both u_1 and u_2. The lower and upper bound for u_1 is 0 and 0.2 and that for u_2 is 0 and 500.

This gives us a dataset containing 250,000 combinations of u_1 and u_2. Second, the curves with a time span of 150 minutes for the concentration y_1 and the temperature of reactor y_2 were solved by *scipy.integrate.solve_ivp* function for each combination of u_1 and u_2. Third, the whole dataset was splitted into training and testing subsets randomly by *sklearn.model_selection.train_test_split* function. Specifically, the size of the training set is 200,000 and the size of the testing set is 50,000.

4.4.2 Data Processing

We loaded the training and testing sets by customizing the *Dataset* Class in Pytorch. We overrided the *__init__()* function that initializes the dataset, the *__len__()* function that queries the length of the dataset, and the *__getitem__()* function that retrieves a specific sample by index. Then, we shuffled the training set (no need to shuffle the testing set) and set the batch size by passing the dataset to the *Dataloader* Class in Pytorch. The batch size is set to be 128.

4.4.3 Multilayer Perceptron

We use one classic type of neural networks, Multilayer Perceptron (MLP), to predict the curves with a time span of 150 minutes for y_1 and y_2 simultaneously based on input u_1 and u_2. Since there are two inputs u_1 and u_2, the input dimension of MLP is 2. As the time span is 150 minutes and the time interval is 1 minute, the dimension of both y_1 and y_2 is 150. In order to improve model efficiency, the same backbone is used to predict y_1 and y_2, which means that the output dimension of MLP is 300. The size of all hidden layers of MLP is 1024. All the linear layers in MLP is followed by a ReLU layer, except the last (output) linear layer. The diagram is shown in figure 3. Mathematically, the MLP can be formulated as:

$$z_1 = \text{input} \cdot W_1 + b_1 \tag{33}$$

$$a_1 = \text{ReLU}(z_1) \tag{34}$$

$$z_2 = a_1 \cdot W_2 + b_2 \tag{35}$$

$$a_2 = \text{ReLU}(z_2) \tag{36}$$

$$z_3 = a_2 \cdot W_3 + b_3 \tag{37}$$

$$a_3 = \text{ReLU}(z_3) \tag{38}$$

$$\text{output} = a_3 \cdot W_4 + b_4 \tag{39}$$

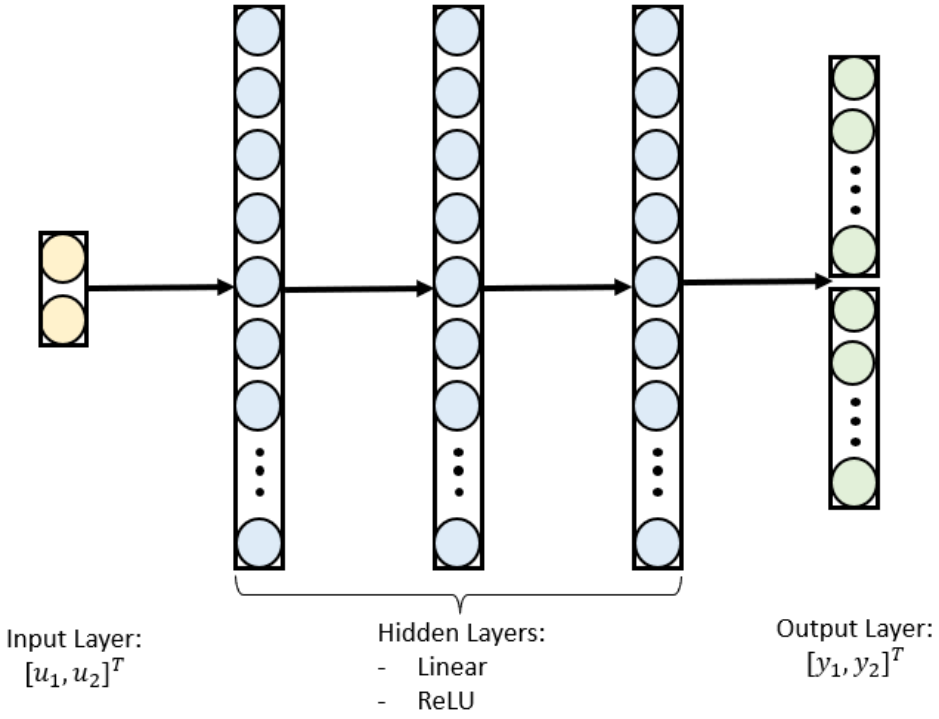
W_i refers to the weight matrix of the i-th linear layer, b_i refers to the bias of the i-th linear layer. The shape of W_1, W_2, W_3 and W_4 is $[2, 1024]$, $[1024, 1024]$, $[1024, 1024]$ and $[1024, 300]$, respectively. The shape of b_1, b_2, b_3 and b_4 is $[1024,]$, $[1024,]$, $[1024,]$ and $[300,]$. ReLU refers to the non-linear activation function.

$$\text{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{40}$$

The objective function to train MLP is the Mean Squared Error Loss, as shown below. $\hat{y}$ is the predicted value, y is the ground truth value, and N is batch size.

$$L(\hat{y}, y) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{41}$$

The optimizer to train MLP is Adam with 0.001 as learning rate and 0.0005 as weight decay.



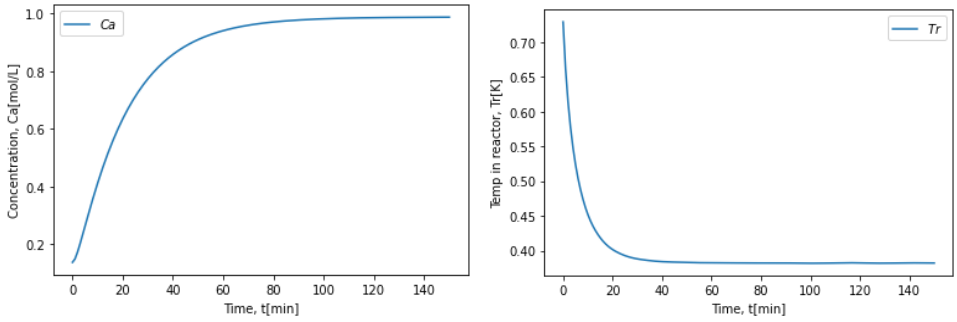
Figur 3: Multilayer Perceptron

5 Results

5.1 Concentration and Temperature Profiles

The curves of concentration and temperature for the tank in the time span of 150 minutes are plotted in Figure 4 solving by solve_ivp function from the scipy library in Python. From these plots, we can observe that the Concentration increases from its initial value, which is equal to 0.1367, to its steady state value, which is around 1. This is expected as the ratio of concentration to final concentration around the end of duration of reaction would be approaching 1.

Additionally, the temperature profile plot shows decrease from its initial value (0.7293) to a value of about 0.38206. This can be attributed to the coolant reducing the temperature in the reactor.



Figur 4: (a) Concentration profile vs time (b) Temperature profile vs time

5.2 Discrete-time State Space Model

Coefficient values A, B, C, D for continuous-time state space model and Φ, Γ, C, D for discrete-time state space model are derived in section 4.2. Following are the matrices values obtained:

$$A = \begin{bmatrix} -0.52972684 & -0.37544061 \\ 0.47972684 & 0.25914061 \end{bmatrix}, B = \begin{bmatrix} 9.0560e-1 & 0 \\ -3.8190e-1 & -7.7025e-5 \end{bmatrix}, \tag{42}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{43}$$

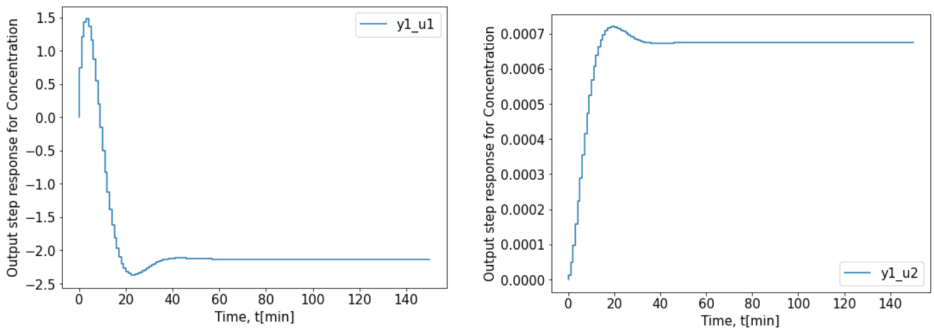
$$\Phi = \begin{bmatrix} 0.51965315 & -0.32659319 \\ 0.41731107 & 1.20588349 \end{bmatrix}, \Gamma = \begin{bmatrix} 7.73836547e-1 & 1.31923405e-5 \\ -2.26312395e-1 & -8.56172099e-5 \end{bmatrix} \tag{44}$$

C and D for discrete are the same as equations of C and D (continuous) above.

5.3 Step Response

The step response model using signal.dstep from the scipy library in Python gives the following plots.

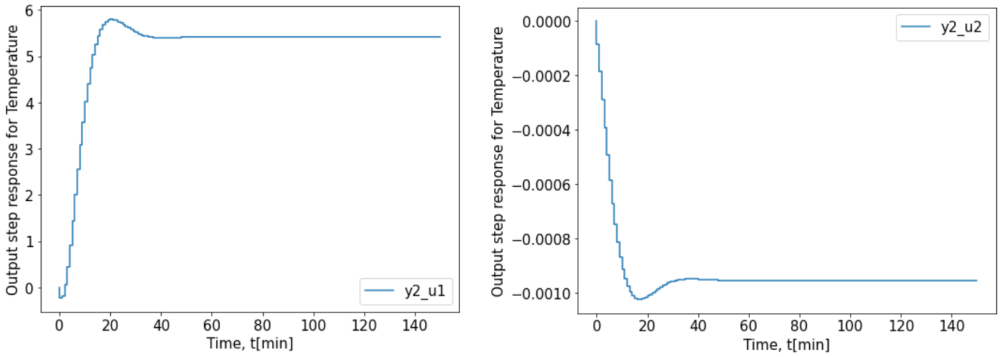
Figure 5 (a) represents the step response for output of concentration for feed flow rate (u_1) with time showing that there is gradual increase in the step response from 0 to about 1.5 and decrease from 1.5 to -2.0 later; and the system achieves steady state around t = 60 min. The increasing trend of this figure indicates that the system concentration has an overall decreasing trend and is able to reach steady state at last. The short increasing of step response indicates the unsteady state stage when that starting the CSTR. This makes sense



Figur 5: (a) Output (conc) step response for feed flow rate vs time (b) Output (conc) step response for heat transfer rate vs time

because with increasing the inlet flow rate, more product would be produce, and therefore the concentration of the reactant would decrease.

In the next plot, the figure 5 (b) shows the step response for output of temperature for feed flow rate (u_1) with time. There is a gradual increase observed in the step response from 0 to about 6, which is approximately the steady state value at t = 60 min. The plot has an overall increasing trend because this is an exothermic reaction and with larger inlet flow rate, more product would be produced and the system temperature would rise up.



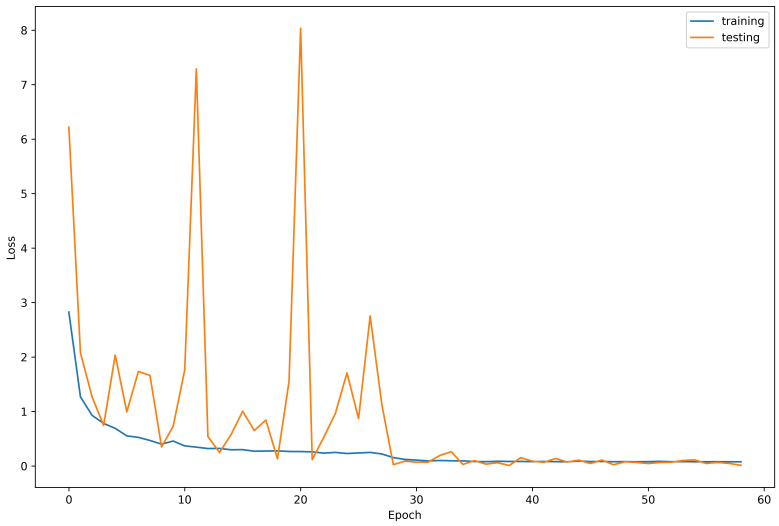
Figur 6: (a) Output (temp) step response for feed flow rate vs time (b) Output (temp) step response for heat transfer rate vs time

The step response for output of concentration for heat transfer rate (u_2) with time can be seen in figure 6 (a). There is increasing in the step response from 0 to about 0.0007. The steady state value is achieved around t = 60 min. This trend indicates that with increasing heat transfer rate, the system efficient, or the efficient of producing would be higher. However, we can see the scale on the y-axis is small, so the effect is limited.

Figure 6 (b) demonstrates the step response for output of temperature for heat transfer rate (u_2) with time. There is decrease in the step response from 0 to about steady state value

at -0.001 at t = 60 min. This trend indicates that with increasing heat transfer rate, the system temperature would be lower since the heat are taken by the coolant flow. However, we can see the scale on the y-axis is small, so the effect is limited, and the overall system temperature would still rise up due to the stronger effect on feed flow rate (u_1).

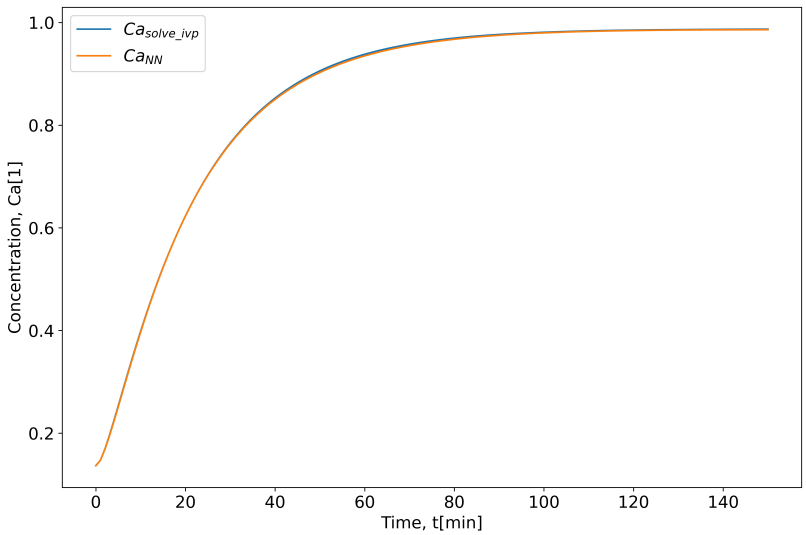
5.4 Deep Learning Prediction



Figur 7: Training and Testing Loss vs Epoch

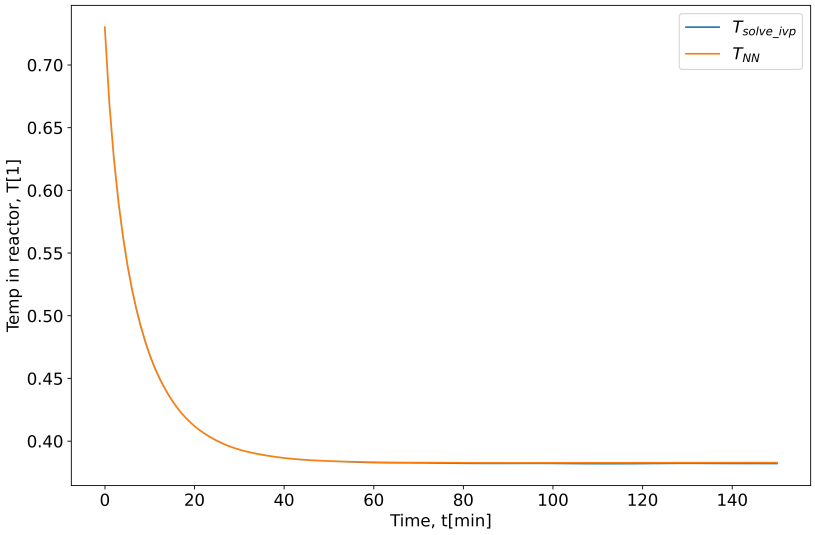
The losses quantify how good or bad a given model make predictions over a given dataset. In our case, the smaller the loss is, the more accurate the predictions given by the model are, the better the model is. The training loss is measured over the data that is seen by model (used to train the model) and the testing loss is measured over the unseen data. Thus, we should use the testing loss to evaluate the performance of the model.

In general, MLP fits really well for both concentration and temperature curves. Over the testing set, the root mean squared error (RMSE) of concentration at individual time step is 0.0073, which is 0.8 % of the average concentration at individual time step (0.8603). The root mean squared error (RMSE) of temperature at individual time step is 0.0021, which is 0.5 % of the average temperature at individual time step (0.4002). The curves for training loss and testing loss during the whole training process is shown in figure 7.



Figur 8: Concentration prediction by MLP for interpolation test

When doing interpolation test of the model (randomly sample one combination of u_1 and u_2 within the lower and upper bound), the model performs very well and the fitting curves for concentration and temperature are shown in figure 8 and figure 9.



Figur 9: Temperature prediction by MLP for interpolation test

6 Reference

[1] AI vs. Machine Learning vs. Deep Learning vs. neural networks: What's the difference? IBM. (n.d.). Retrieved April 25, 2022, from https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

[2] Mettler-Toledo International Inc. all rights reserved. (2022, February 8). Mettler Toledo Balances amp; Scales for Industry, lab, retail. METTLER TOLEDO Balances amp; Scales for Industry, Lab, Retail - METTLER TOLEDO. Retrieved April 24, 2022, from https://www.mt.com/us/en/home/products/L1 AutochemProducts/Chemical-Synthesis-and-Process-Development-Lab-Reactors/continuous-stirred-tank-reactors-cstr.html

[3] Biegler, L. T. (n.d.). Computers and Chemical Engineering - cepac.cheme.cmu.edu. Retrieved April 24, 2022, from http://cepac.cheme.cmu.edu/pasi2011/library/biegler/CACE_3678.pdf

[4] Tlacuhua, A.F.; Moreno, S.T.; Biegler, L.T. Global Optimization of Highly Nonlinear Dynamic Systems. Ind. Chem.Res 2008, 47, 2643 - 2655.

[5] Hicks, J; Mohan,A; Ray,W.A. The Optimal Control of Polymerization Reactors. The Canadian Jouranl of Chemical Engineering 1969, Vol. 47, 590 -596.

[6] Salahshour,E; Malekzadeh,M; Gordillo,F; Ghasemi,J. Quantum neural network-based intelligent controller design for CSTR using particle swarm optimization algorithm. Transaction of the Institute of Measurement and Control 2019, 41, 392-404.

[7] Shrivastava,P; Modeling and Control of CSTR using Model based Neural networks Predictive Control

[8] Shyamalagowri,M;Rajeswari,R. Neural Network Controller based Nonlinearity Identification Case Study: Nonlinear Process Reactor- CSTR. Advanced Material Research 2016, 985, 1326- 1334.

[9] Putrus,k.M. Implementation of Neural Control for Continuous Stirred Tank Reactor (CSTR), Al- Khwarizmin Engineering Journal 2011, Vol. 7, 39-55.

[10] Patterson D. W; Artificial Neural Networks Theory and Applications. Prentice Hall 1996, 220.