mathematics

Article

Recurrent Neural Network-Based Model Predictive Control for Continuous Pharmaceutical Manufacturing

Wee Chin Wong, Ewan Chee, Jiali Li and Xiaonan Wang *

Department of Chemical & Biomolecular Engineering, Faculty of Engineering, National University of Singapore, 4 Engineering Drive 4, Singapore 117585, Singapore; weechin.wong@gmail.com (W.C.W.); ewan-jun-xian.chee@student.ecp.fr (E.C.); e0276496@u.nus.edu (J.L.)

* Correspondence: chewxia@nus.edu.sg; Tel.: +65-6601-6221

Received: 5 August 2018; Accepted: 26 October 2018; Published: 7 November 2018

Abstract: The pharmaceutical industry has witnessed exponential growth in transforming operations towards continuous manufacturing to increase profitability, reduce waste and extend product ranges. Model predictive control (MPC) can be applied to enable this vision by providing superior regulation of critical quality attributes (CQAs). For MPC, obtaining a workable system model is of fundamental importance, especially if complex process dynamics and reaction kinetics are present. Whilst physics-based models are desirable, obtaining models that are effective and fit-for-purpose may not always be practical, and industries have often relied on data-driven approaches for system identification instead. In this work, we demonstrate the applicability of recurrent neural networks (RNNs) in MPC applications in continuous pharmaceutical manufacturing. RNNs were shown to be especially well-suited for modelling dynamical systems due to their mathematical structure, and their use in system identification has enabled satisfactory closed-loop performance for MPC of a complex reaction in a single continuous-stirred tank reactor (CSTR) for pharmaceutical manufacturing.

Keywords: pharmaceuticals; critical quality attributes (CQAs); recurrent neural networks; model predictive control (MPC); system identification

1. Introduction

The pharmaceutical industry has shown a growing interest in adopting the concept of continuous manufacturing [1,2] to reduce waste, cost, environmental footprints and lead-times. Coupled with developments in continuous manufacturing technologies and quality by design (QbD) paradigms, there has thus recently been an increasing demand for more advanced system identification and process control strategies for continuous pharmaceutical manufacturing [1].

Continuous manufacturing has the potential of extending the palette of permissible reaction conditions and enabling reaction outcomes that are quite challenging when performed under batch conditions [3–5], and its application to the production of complex Active Pharmaceutical Ingredients (APIs) is an emerging research domain. However, the reactions involved are generally both reversible and coupled with many side reactions, which leads to highly non-linear systems that impose difficulties for system identification. The need to operate in tight and extreme conditions also necessitates stricter control requirements [6–8]. The complexity of the system, as well as its potential impact on reaction yields, downstream processing requirements and critical quality attributes (CQAs) of final products, render model identification and control of continuous API reactions particularly challenging.

Advanced model-based control technologies promise to improve control of CQAs and are seen as vital to enabling continuous pharmaceutical manufacturing, and their performance hinges on the quality of the model obtained from system identification. Many rigorous physics-based models have been proposed to describe different API reactions [6,9,10] and they have clear physical interpretations. However, these models can suffer from complicated structures and may incur excessive computational costs during online control. The need for models that evaluate quickly for online control purposes is not unique to the pharmaceutical industry, and extensive work has also been done for thin-film deposition in the microelectronics industry [11,12]. Moreover, it may be extremely difficult and expensive to derive accurate models of non-linear systems from first principles in the first place [13]. In contrast, experimental data-driven heuristic models aim to describe underlying processes with relatively simpler mathematical expressions and are thus generally easier to obtain [14]. Neural networks belong to this class of data-driven models, and they have been extensively applied in the chemical and biochemical processing industries, particularly in modelling complex non-linear processes whose process understanding is limited to begin with [15–18]. The goal of system identification is to create models that capture the relationships between the process state variables and control variables, and data-based empirical models for system identification have been widely studied in previous work [19,20]. Artificial Neural Networks (ANNs) are interesting candidate models for these purposes due to their ability to approximate any continuous function, subject to assumptions on the activation function. RNNs constitute a sub-class of ANNs, and they are structured to use hidden variables as a memory for capturing temporal dependencies between system and control variables. These networks have been extensively used in sequence learning problems like scene labeling [21] and language processing [22], demonstrating good performance. Ref. [23] provides an overview of the neural network applications, particularly from a process systems engineering perspective. In this work, the use of RNNs for system identification of a complex reaction in a single CSTR for pharmaceutical API production will be demonstrated.

The design of effective model-based control algorithms for continuous manufacturing is also a key research focus in academia and industry [24], and the model predictive control (MPC) method is a prominent example that has been widely employed in industrial applications [25–27]. It has also been applied in the control of CQAs in continuous pharmaceutical manufacturing [10,24], and an example can be found in [10] where the authors presented two plant-wide end-to-end MPC designs for a continuous pharmaceutical manufacturing pilot plant. This plant covered operations from chemical synthesis to tablet formation, and the MPC designs were obtained using the quadratic dynamic matrix control (QDMC) algorithm. The results showed that, by monitoring the CQAs in real time, critical process parameters (CPPs) could be actively manipulated by feedback control to enable process operation that is more robust and flexible. The advantages that MPC has over conventional proportional-integral-derivative (PID) control are highlighted in [24], which compared their control performance for a feeding blending unit (FBU) used in continuous pharmaceutical manufacturing.

Previous studies have explored the potential of using RNNs with MPC. For example, ref. [28] investigated the use of RNNs in controlling non-linear dynamical systems. RNNs have also been used for predictive control of reactions in CSTRs [29]. However, this CSTR study was limited to relatively simple kinetics like single-step irreversible reactions, i.e., $A \rightarrow B$. This paper illustrates therefore how RNNs can be used with MPC of reactions with more complex kinetics.

The paper is organized as follows. Section 2 describes the plant model, which is the subject of system identification, and the control scenarios that will be used to assess closed-loop control performance. Section 3.1 articulates the RNN-based system identification methodology and Section 3.2 describes the MPC problem formulation. Results and discussions are shown in Section 4.

2. Plant Model and Control Scenarios

The following sub-sections describe (i) the true plant model for which system identification will be performed and (ii) the two control scenarios that will be used to assess closed-loop control performance.

2.1. Plant Model

The reaction system comprising of a single CSTR, as was also analysed in [30,31], will be the object of this study. For this work, the focus is on system identification using RNNs with a view towards closed-loop control with MPC, thereby marking a difference from how this reaction system was studied in previous literature. The reaction is described by Equation (1), where A is the feed species, R is the reaction intermediate and the desired product, and S is the undesired by-product:

$$A \underset{\mathrm{k}_4}{\overset{\mathrm{k}_1}{\rightleftarrows}} R \underset{\mathrm{k}_3}{\overset{\mathrm{k}_2}{\rightleftarrows}} S \tag{1}$$

The equations used to represent the dynamical behavior of this system are based on normalized dimensionless quantities and are as shown in Equations (2)–(4) below:

$$\frac{dC_A}{dt} = q[C_{A0} - C_A] - k_1 C_A + k_4 C_R \tag{2}$$

$$\frac{dC_R}{dt} = q[1 - C_{A0} - C_R] + k_1 C_A + k_3[1 - C_A - C_R] - [k_2 + k_4]C_R \tag{3}$$

$$k_j = k_{0,j} \exp\left\{\left[-\frac{E}{RT_0}\right]_j \left[\frac{1}{T} - 1\right]\right\}, j \in \{1,2,3,4\} \tag{4}$$

where, $C_{i\in\{A,R,S\}} \in \mathbb{R}_+$ refers to the species concentration in the reactor, $C_{A0} \in \mathbb{R}_+$ the feed concentration of A, $q \in \mathbb{R}_+$ the feed flow rate, and $k_{j\in\{1,2,3,4\}} \in \mathbb{R}_+$ the rate constants for the respective reaction. For each rate constant, $k_{0,j\in\{1,2,3,4\}} \in \mathbb{R}_+$ is its associated Arrhenius pre-exponential constant, and $[\frac{E}{RT_0}]_{j\in\{1,2,3,4\}} \in \mathbb{R}$ its associated normalised activation energy.

The reactions are first-order and reversible. The reactor is operated isothermally and perfect mixing is assumed. All reactor concentrations are measured in this example. The manipulated variables are q and T, and their values are also measured. It is also assumed that the only species entering the CSTR are A and R, such that their feed concentrations sum up to unity. As such, C_S is easily calculable as a function of time. The values of $k_{0,j\in\{1,2,3,4\}}$ and $[\frac{E}{RT_0}]_{j\in\{1,2,3,4\}}$ can be found in [31] and are reproduced in the Appendix A for ease of reference.

Figure 1 shows that, at steady-state conditions for a given q, C_R reaches a maximum at a certain temperature, beyond which the reaction gets driven back to the left of Equation (1) to eventually yield only A and no R. This system possesses rich and relevant dynamics for continuous pharmaceutical manufacturing and is therefore worth detailed investigation.

The control problem for this system is challenging due to the existence of input multiplicities [30,32], which occur when a set of outputs can be reached by more than one set of inputs. This implies the existence of sign changes in the determinant of the steady-state gain matrix within the operating region. Linear controllers with integral action therefore become unstable for this problem.

In what follows, as is common in digital control, the discrete-time domain with time-steps denoted by $k \in \mathbb{Z}_+$ will be used. The state vector, x, and manipulated vector (MV), u, are also defined as follows: $x \triangleq [C_A, C_R]'$, $u \triangleq [q, T]'$. The underlying plant model can therefore be described by a non-linear discrete-time difference equation, as Equations (5) and (6) show:

$$x_{k+1} = \Phi(x_k, u_k) \tag{5}$$

$$y_k = x_k \tag{6}$$

where $k \in \mathbb{Z}_+$ is the discrete-time index, and x_k and u_k the state vector and manipulated vector at time-step k respectively. $\Phi(.)$ is the state-transition equation consistent with Equations (2)–(4), and full state feedback is assumed, such that the measured output y_k equals x_k for all time-steps.

The system identification problem consists of finding an approximation of Equations (2)–(4), and forms the subject of Section 3.1 in this paper. It will be supposed that the $p-$step ahead prediction problem is of vital interest for MPC. Throughout this paper, all problems are considered from time-step zero, and a sampling time Δt, of 0.1 time units is used.

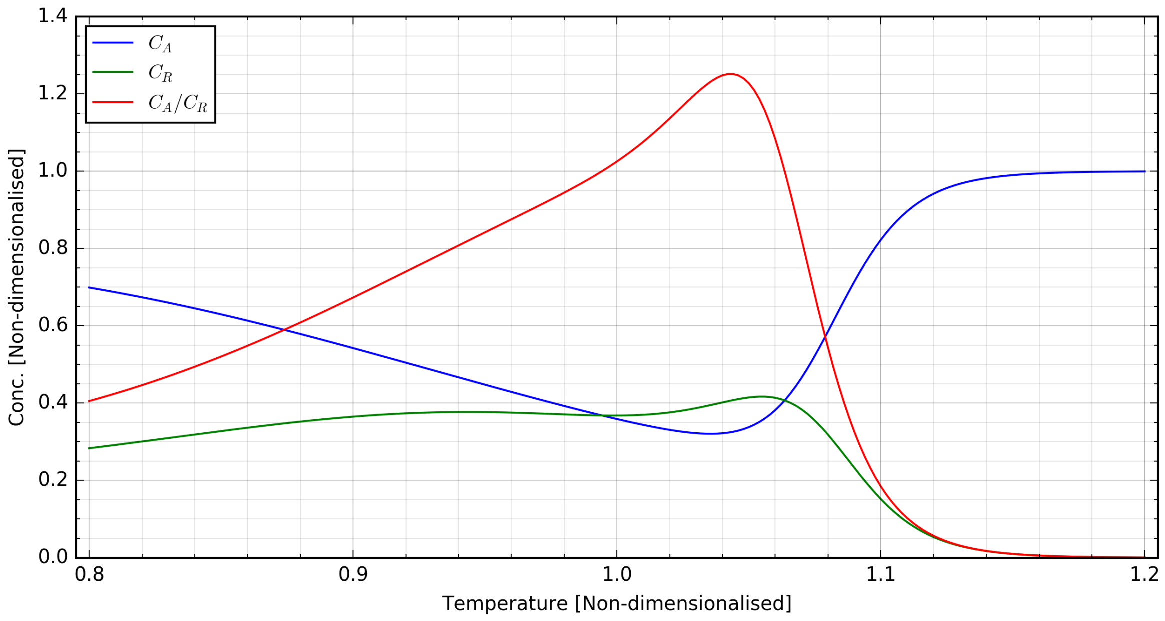


Figure 1. Steady-state conditions as a function of T with $q = 0.8$.

2.2. Closed-Loop Control Scenarios

The approach taken to construct the RNN for MPC is to train the RNN with data from the training set, then evaluate the model performance using previously unseen data from the test set. However, since the main focus of this study is process control, the final evaluation will be performed by comparing the closed-loop performance of the RNN-based MPC (RNN-MPC) against a benchmark non-linear MPC (NMPC) controller that directly uses the true plant model, as described in Equations (2)–(4), as its internal model.

Two control scenarios were selected for performance evaluation and the operating conditions for each scenario are reflected in the first two rows of Table 1. These scenarios are namely cases I and II, and they correspond respectively to a reactor system start-up case and an upset-recovery case. In case I, the system is assumed to initially be at a low temperature state with a relatively low product concentration. In case II, the initial conditions correspond instead to a high-temperature and low-yield state. Figure 2 shows the locations of the initial conditions for these control scenarios with respect to the set-point, with case I represented by the purple point to the left of the peak in the green C_R plot, and case II represented by the cyan point to the right of the same peak.

Table 1. Initial conditions and set-points for control scenarios.

Control Scenario	C_A	C_R	q	T
I: Start-up	0.692	0.287	0.800	0.800
II: Upset-recovery	0.822	0.152	0.800	1.100
Set-point (maximum C_R/C_A)	0.324	0.406	0.800	1.043

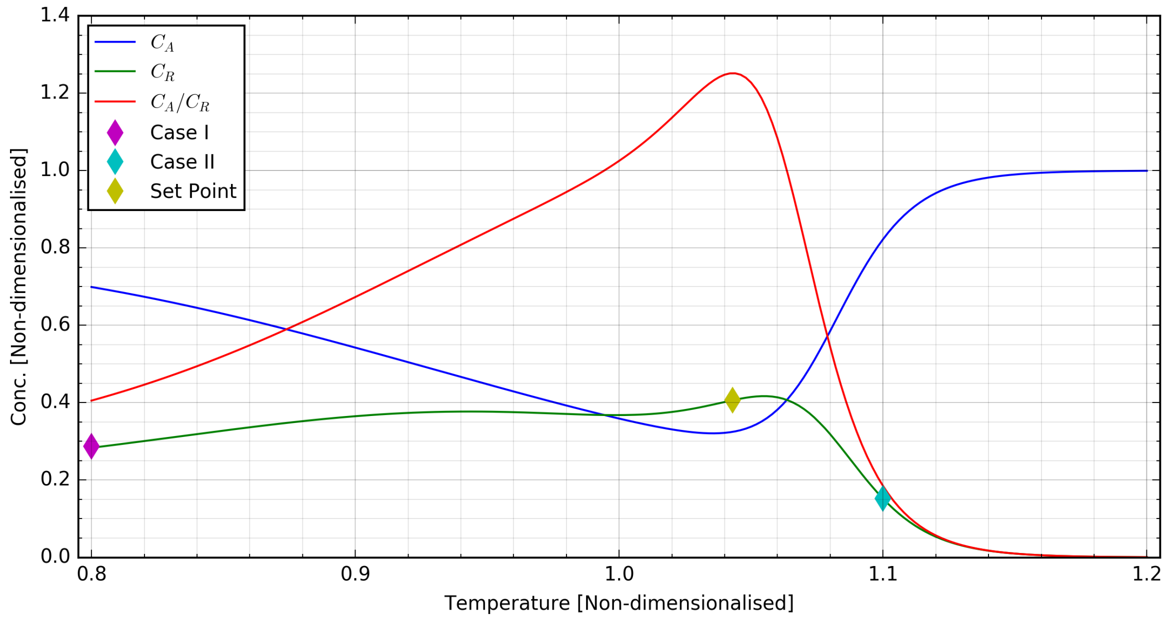


Figure 2. Initial conditions for cases I (start-up) and II (upset-recovery) relative to the set-point.

The set-point is judiciously located where the ratio of the product concentration to the feed concentration, C_R/C_A, is maximised. This set-point corresponds to a product concentration that is slightly lower than what is maximally achievable. The rationale for choosing this set-point over the point for maximum C_R is that it maximizes yield whilst minimising downstream separations cost. The operating conditions corresponding to the target set-point are shown in the last row of Table 1.

3. Methodology

3.1. Non-Linear Time-Series System Identification via Recurrent Neural Networks

ANNs serve as a potential source of models that capture the dynamics of highly non-linear systems sufficiently well while being relatively easy to obtain and evaluate online, and RNNs are a sub-class of ANNs that are structured to better capture temporal dependencies in the system. RNNs are particularly useful for making p-step ahead predictions of state variables for predictive control, because the prediction for time-step p depends on the present state and all control actions in time-step $k \in \{0, ..., p-1\}$. The prediction for time-step $p-1$ used in the above prediction for time-step p depends similarly on the present state and all control actions in time-step $k \in \{0, ..., p-2\}$, and so on.

Figure 3 illustrates the structure of an RNN layer in its compact and unfolded forms. The unfolded form reveals the repeating cells forming each layer, and each cell is taken in this study to represent a time-step, such that the state of the cell representing time-step $k \in \{0, ..., p-1\}$ serves as the input for a cell representing time-step $k+1$. Each cell contains N number of hidden nodes that encode the state representation, where N is user-specifiable. The equations that mathematically characterise a single RNN cell in a single-layer RNN are as shown in Equations (7) and (8) below:

$$h_k = W_{h,h}h_{k-1} + W_{u,h}u_{k-1} + b_1 \tag{7}$$

$$\hat{y}_k = \Phi(h_k + b_2) \tag{8}$$

where $k \in \{1, ..., p\}$ is the discrete-time index with p the prediction horizon, $h_k \in \mathbb{R}^N$ the (hidden) state of the cell representing time-step k, $u_{k-1} \in \mathbb{R}^2$ the cell input, $\hat{y}_k \in \mathbb{R}^2$ the cell output which corresponds to the state vector prediction for time-step k, $(b_1, b_2) \in (\mathbb{R}^N)^2$ the offset vectors, $W_{u,h} \in \mathbb{R}^{N\times 2}$ and $W_{h,h} \in \mathbb{R}^{N\times N}$ the input-to-hidden-state weight matrix and the hidden-state-to-hidden-state weight matrix respectively, and $\Phi : \mathbb{R}^N \to \mathbb{R}^2$ an activation function that is applied element-wise. h_0 is initialised in this study by using y_0.

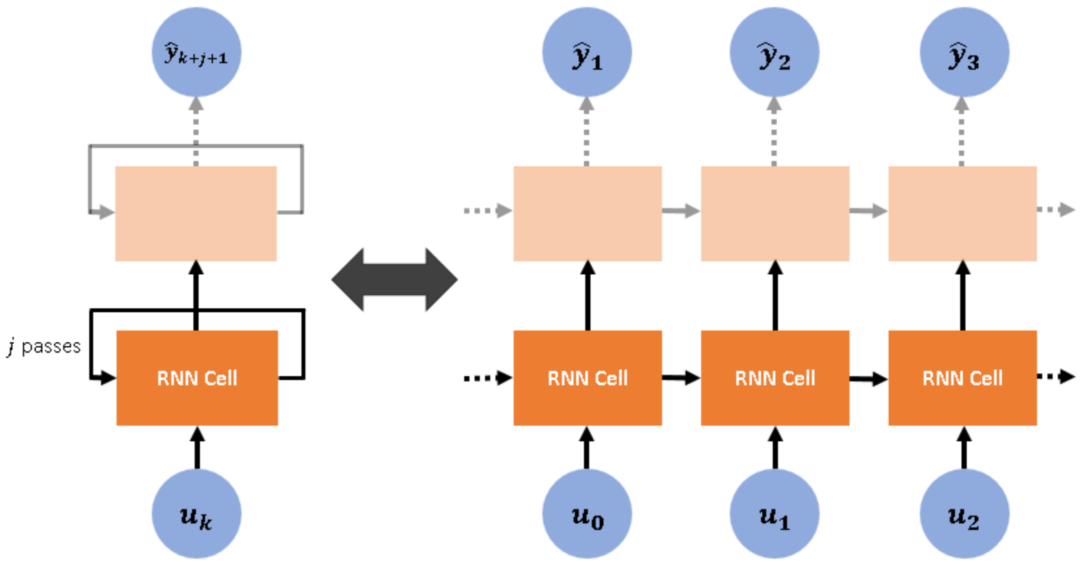


Figure 3. Illustration of a recurrent neural network (RNN) layer in compact form (**left**) and unfolded form (**right**).

Equations (7) and (8) are generalisable to deep RNNs containing more than one layer. For an l-layer RNN, the activation functions for layers $j \in \{2, ..., l\}$ take the form $\Phi : \mathbb{R}^N \to \mathbb{R}^N$, and the input-to-hidden-state weight matrices have dimensions $N \times N$ instead. Deep RNNs may be preferred to single-layer RNNs for their enhanced ability to learn features of the dynamical system that occur "hierarchically", but this comes at the expense of longer RNN training times stemming from more parameters to train.

The regressors required to predict $\hat{y}_{k \in \{1,...,p\}}$ are henceforth represented by $\phi_k := \{y_0, u_0, \ldots, u_{k-1}\}$, and they are introduced into the RNN in a fashion illustrated in Figure 4. Equation (9) below serves as a shorthand to describe the RNN:

$$\hat{y}_k = \hat{f}_{RNN}(\phi_k), k \in \{1, ..., p\} \tag{9}$$

An RNN is characterised by the values of $W_{h,h}$, $W_{u,h}$, b_1 and b_2 for all layers, and these values constitute the set of parameters. These parameters are learnt from training data by minimising the predictive error of the model on the training set as determined through a user-specified loss function. The learning process is performed through the back-propagation through time (BPTT) algorithm that estimates the gradient of the loss function as a function of the weights, and an optimisation algorithm that uses the calculated gradient to adjust the existing weights. The adaptive moment estimation algorithm (Adam) [33] is an example of an optimisation algorithm that is widely used.

The RNN parameters may be difficult to train in some cases due to problems with vanishing and exploding gradients, and RNN cells with different structures can be used to circumvent this issue. Long Short-Term Memory cells (LSTMs) were developed to address these problems, and they are used as the repeating cell in this study. Further details on LSTM cells may be found in the Appendix B.

To generate the data set for training the RNNs in this study, a numerical model of the CSTR system based on Equations (2)–(4) is needed, because this data consists precisely of this system's response to MV perturbations. This numerical model was implemented in Python version 3.6.5, and its temporal evolution between sampling points was simulated using the explicit Runge-Kutta method of order 5(4) through the `scipy.integrate.solve_ivp` function.

MV perturbations are then introduced into the system to obtain its dynamic response. This procedure mimics the manual perturbation of the system through experiments, which become necessary in more complicated reaction systems whose reaction kinetics and overall dynamics resist expression in neat analytical forms. These perturbations take the form of pre-defined sequences of control actions, $\{u_{\text{exp},k}\}_{k\in\{0,\ldots,T_e-1\}}$, where the "exp" subscript refers to "experiment" and T_e represents the final time-step for the experiment. These control actions are separated by Δt.

To simulate the system's dynamic response to the experimental sequence, the control actions associated to the element in this sequence, $u_{\text{exp},k} = [q_{\text{exp},k}, T_{\text{exp},k}]'$, $k \in \{0,\ldots,T_e-1\}$, are applied at time-step k for a period of Δt, during which the system's evolution with this MV is simulated using the Runge-Kutta method of order 5(4). Once Δt elapses, $u_{\text{exp},k+1}$ is applied with the system evolution proceeding in the same manner. This procedure repeats until the final time-step, T_e, is completed.

A history of the system's experimental dynamic response is associated to each experimental sequence, which takes the form of $\{y_{\text{exp},k}\}_{k\in\{1,\ldots,T_e\}}$. $y_{\text{exp},k}$ is the measured system output at time-step k after $u_{\text{exp},k-1}$ has been applied to the system for a period of Δt. This history corresponds to the labels in machine learning terminology, and the data set is thereafter constructed from both the experimental sequences and their associated labels. For the p-step ahead prediction problem, each data point thus takes the form $\{y_k, u_k, u_{k+1},\ldots,u_{k+p-1}\}$ with the associated label $\{y_{k+1},\ldots,y_{k+p}\}$, $k \in \{0,\ldots,T_e-p\}$. $T_e - p$ data points can thus be extracted from each experimental sequence. Multiple experimental sequences can be constructed by perturbing the MVs in different ways, and such a sequence is shown in Figure 5 with its associated system dynamical response, with this sequence showing variations of q in the range $[0.70, 1.05]$ for fixed values of T. For this study, these sequences were generated in a fashion similar to Figure 5, by introducing linear changes of different frequencies to one component of the MV spanning its experimental range while keeping the other MV component constant.

Before training the RNNs, it is necessary to split the data set into the training set and the test set. The training set contains the data with which the RNN trains its weight matrices, and the test set will be used to determine the models with the best hyperparameters. The two hyperparameters considered in this study are the number of hidden nodes, which also correspond to the dimensionality of the hidden states, and the number of hidden layers.

The experimental simulations and the extraction of training data were performed using Python version 3.6.5 for this study. Keras version 2.1.5, an open-source neural network library written in Python, was used for RNN training with TensorFlow version 1.7.0 as its backend.

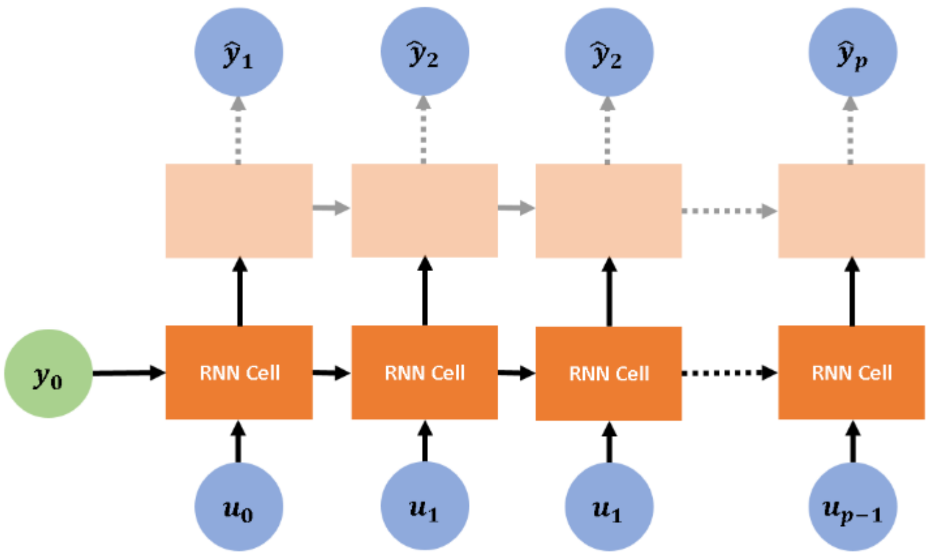


Figure 4. RNN structure for the p-step ahead prediction problem.

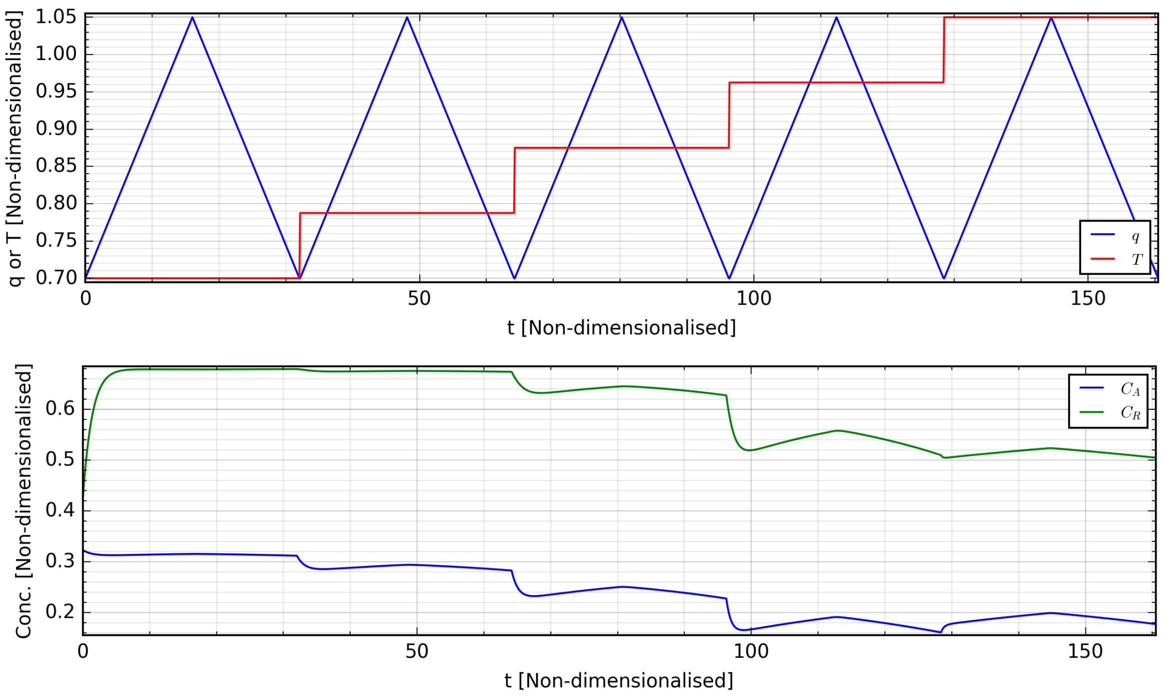


Figure 5. Example of an experimental sequence and the dynamic response with $\Delta t = 0.1$ time units.

3.2. Control Problem Formulation

MPC is a control strategy that selects a set of m future control moves, $\{u_0, ...u_{m-1}\}$, that minimises a cost function over a prediction of p steps by incorporating predictions of the dynamical system for these p steps, $\{\hat{y}_1, ..., \hat{y}_p\}$. The cost function is typically chosen to punish large control actions, which implies greater actuator power consumption, and differences between the state vector and the set-point at each time-step. Input and output constraints can also be factored into the MPC formulation. Since MPC performance depends on the quality of the predictions of the system, obtaining a reasonably accurate model through system identification is crucial.

The MPC problem can be formulated as shown in Equations (10)–(13) below:

$$\min_{\{\Delta u_0, \Delta u_1, \ldots, \Delta u_{m-1}\}} \left\{ \sum_{k=1}^{p} (\hat{y}_k - y_k^*)' Q_y (\hat{y}_k - y_k^*) + \sum_{k=0}^{m-1} \Delta u_k' Q_u \Delta u_k \right\} \text{ s.t.} \tag{10}$$

$$\hat{y}_k = \hat{f}_{RNN}(\phi_k), k \in \{1, \ldots, p\} \tag{11}$$

$$u_k \in [u_{\min,k}, u_{\max,k}], k \in \{0, \ldots, m-1\} \tag{12}$$

$$\Delta u_k \in [\Delta u_{\min,k}, \Delta u_{\max,k}], k \in \{0, \ldots, m-1\} \tag{13}$$

where $p \in \mathbb{Z}_+$ is the prediction horizon, $m \in \{1, \ldots, p\}$ the control horizon, $\hat{y}_k \in \mathbb{R}^2$ the prediction of the state vector for the discrete-time step k obtained from the RNN, $\hat{f}_{RNN}$ described in Equation (9), $y_k^* \in \mathbb{R}^2$ the set-point at time-step k, $u_k \in \mathbb{R}^2$ the MV for time-step k, $\Delta u_k \triangleq u_k - u_{k-1}$ the discrete-time rate of change of the MV which corresponds to the control action size at time-step k, $(Q_y, Q_u) \in (\mathbb{R}^{2\times 2})^2$ symmetric positive semi-definite weight matrices, and $(u_{\min,k}, u_{\max,k}, \Delta u_{\min,k}, \Delta u_{\max,k}) \in (\mathbb{R}^2)^4$ the lower and upper bounds for the control action and the rate of change of the control action at time-step k. In the above formulation, it is assumed that there are no changes in actuator position beyond time-step $m-1$, i.e., $\Delta u_{m-1} = \Delta u_{m+k} = 0, k \in \{0, \ldots, p-m-1\}$.

This optimization problem is not convex in general and thus does not possess special structures amenable to global optimality. This is therefore a Non-Linear Programming (NLP) problem, and modern off-the-shelf solvers can be used to solve it. This problem is solved at every time-step to yield the optimal control sequence for that time-step, $\{\Delta u_0^*, \ldots, \Delta u_{m-1}^*\}$. The first element, Δu_0^*, is then applied to the system until the next sampling instant, where the problem is solved again to yield another optimal control sequence. This procedure is then repeated in a moving horizon fashion.

The MPC controller in this study was implemented in Python version 3.6.5 through the `scipy.optimize.minimize` function, and the sequential least squares quadratic programming (SLSQP) algorithm was selected as the option for this solver.

4. Results and Discussion

The RNN-MPC workflow consists of (i) gathering experimental sequence data with perturbations of the MV to generate the data set; (ii) learning the RNNs from the training set and validating their performance with test data, and selecting the RNNs with the best test performance; and (iii) integrating the chosen RNN with the MPC and finally evaluating its control performance.

4.1. System Identification

RNNs with 250, 500 and 1000 hidden nodes were trained for this study, and the number of RNN layers for these models was also varied from 1 to 3. The validation performance of these models on the test set was quantified through the root-mean-square error (RMSE) metric, which is defined as shown in Equation (14) below:

$$\text{RMSE} = \sqrt{\left(\frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}}\sum_{j=1}^{p}\left((\hat{y}_j)_i - (y_{\text{exp},j})_i\right)^2\right)} \tag{14}$$

where $N_{\text{test}} \in \mathbb{Z}_+$ is the number of data points in the test set, $p \in \mathbb{Z}_+$ the prediction horizon, $(\hat{y}_j)_i \in \mathbb{R}^2$ the j-ahead output prediction for data point i, and $(y_{\text{exp},j})_i \in \mathbb{R}^2$ the corresponding experimentally-determined output which also represents the ground truth that predictions are tested against. The smaller the RMSE of the model on the test data, the better its general predictive power.

Table 2 tabulates the validation performance of the RNNs on the test data, and it can be observed that prediction performance improves with an increasing number of hidden nodes. This is because hidden states of greater dimensionalities offer greater latitude for capturing more features of the underlying system, contributing to its temporal dynamics. An improvement of the validation performance can also be observed with an increase of the number of hidden layers from 1 to 2. This observation is attributable to the potential of deep hierarchical models to better represent some functions than shallower models [34].

Table 2. Root-mean-square error (RMSE) over test data of RNNs trained over 1000 epochs.

No. Layers / No. Nodes	250	500	1000
1	0.0299	0.0268	0.0206
2	0.0238	0.0118	0.0083
3	0.0262	0.0119	0.0125

However, performance deteriorated when the number of layers was increased from 2 to 3, and this may be due to the additional layers introducing more model parameters that needed to be tuned, such that the training data set became too small to tune all of them correctly.

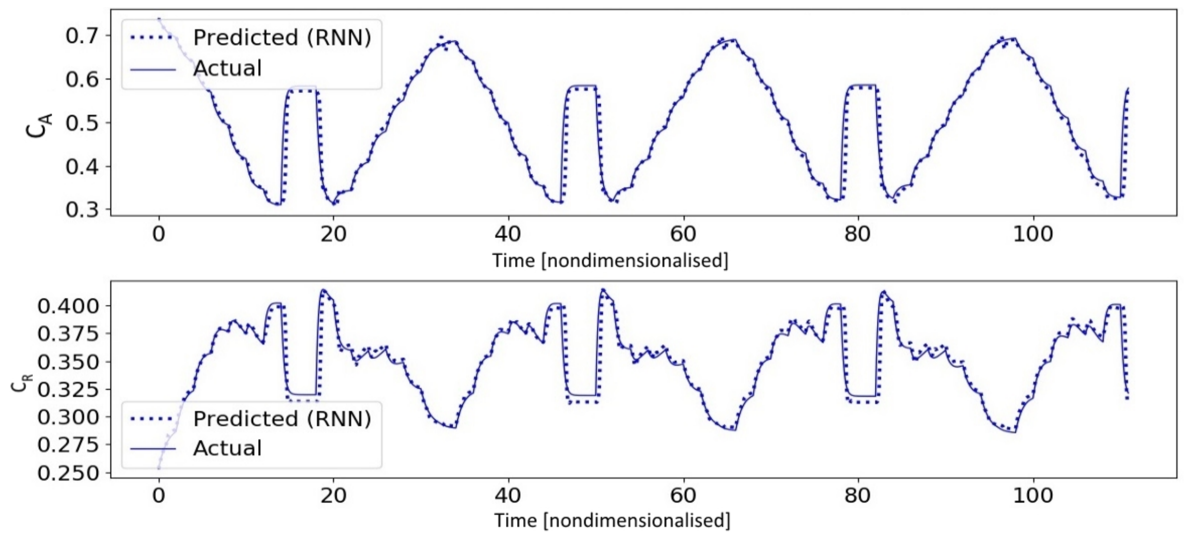
Table 2 reveals that RNNs with two hidden layers and 1000 hidden nodes for each RNN cell gave the best validation performance over the test set, with this representing the optimised hyperparameter configuration. An RNN with two hidden layers and 2000 hidden nodes was also trained and it was observed that performance deteriorated when even more hidden nodes were used. This result is tabulated in Table 3, and possible explanations may be that the 2000-node network overfitted the training data, or that the use of 2000-node cells introduced more parameters to the model, rendering the existing training data set too small to tune the additional parameters correctly. RNN with two hidden layers will be used for subsequent closed-loop control studies.

Table 3. RMSE over test data of RNNs with hidden layers and either 1000 or 2000 nodes trained over 1000 epochs.

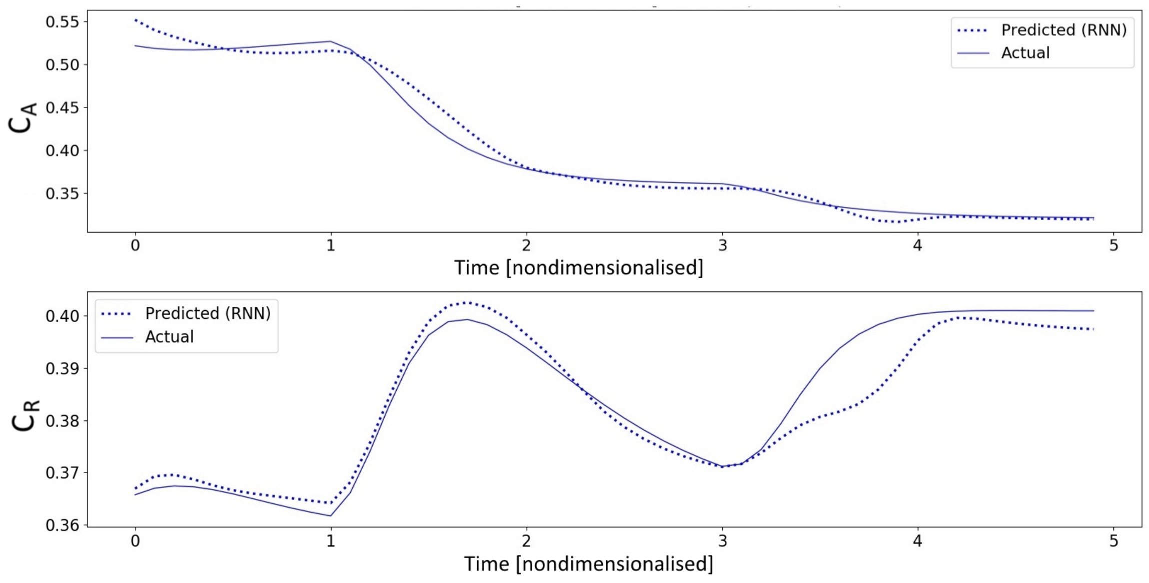
No. Layers	No. Nodes	RMSE
2	1000	0.0083
2	2000	0.0177

It is also worth noting that validation performance is also a function of the quality of the training set. This performance can be further improved by using a larger training set reflecting the system dynamics within the control range, particularly for any control scenarios earmarked as important.

The prediction performances of this optimised RNN on the training and test sets are respectively shown in Figure 6 below for completeness.



(**a**)



(**b**)

Figure 6. System identification performance of the optimised RNN. (**a**) Training performance of the optimised RNN. (**b**) Validation performance of the optimised RNN on test data.

Figure 6a reveals a good fit of the RNN to the training data, and testifies to the model's ability to reflect highly dynamic outputs from highly dynamic training data. Figure 6b shows that the model succeeded in capturing the general trends for previously unseen test data. Even though this fit was not perfect, particularly for C_R between 3 and 4 time units, it will be demonstrated in a later section that satisfactory closed-loop performance can still be achieved with predictions from this RNN.

4.2. RNN-MPC Closed-Loop Control Performance

As described in Section 2.2, the objective in both control scenarios is to bring the system quickly and smoothly to a target set-point where the ratio of C_R to C_A is maximised, because this maximises yield while minimising downstream separations cost. Since MPC strategies based on single linear models do not perform well for this example [32], non-linear MPC solutions will be required.

The MPC formulation was shown in Equations (10)–(13), and the prediction and control horizons for the MPC controller implemented for this study were set to 10 for both control scenarios, i.e., $p = m = 10$. Since the sampling rate, Δt, was set as 0.1 time units, p and m correspond to 1.0 time unit. The total simulation time, N_{sim}, was set to 40, corresponding to 4.0 time units. The weight matrices for the controller, Q_y and Q_u, were defined as $\text{diag}[2.4, 5.67]$ and $\text{diag}[25, 25]$ respectively. The coefficients selected for Q_y reflect that controlling C_R is more important than controlling C_A, and Q_u contain coefficients that are even larger to avoid control actions that are over-aggressive, since these control actions cause system instability. For the constraints, $[u_{\min,k}, u_{\max,k}] = [[0.75, 0.5]', [0.85, 1.1]']$, $\forall k \in \{0, N_{\text{sim}-1}\}$ and $[\Delta u_{\min,k}, \Delta u_{\max,k}] = [[-0.1, -0.1]', [0.1, 0.1]']$, $\forall k \in \{0, N_{\text{sim}-1}\}$.

In what follows, the closed-loop control performance of the RNN-MPC controller will be benchmarked against the NMPC controller whose internal model is the actual full plant model as described in Equations (2)–(4). This NMPC controller was implemented using the same software as that of the RNN-MPC controller, whose implementation was described in Section 3.2.

The indices used to evaluate control performance are defined as shown in Equations (15) and (16) below:

$$J = \sum_{k=1}^{N_{\text{sim}}} \left((y_k - y_k^*)' Q_y (y_k - y_k^*) + \Delta u_k' Q_u \Delta u_k \right) \tag{15}$$

$$\mathcal{I} = \left(1 - \frac{J_{\text{RNN}} - J_{\text{NMPC}}}{J_{\text{NMPC}}} \right) \times 100\% \tag{16}$$

where $y_{k \in \{1,\ldots,N_{\text{sim}}\}} \in \mathbb{R}^2$ and $y^*_{k \in \{1,\ldots,N_{\text{sim}}\}} \in \mathbb{R}^2$ are the measured output and the set-point at time-step k respectively, $\Delta u_k \in \mathbb{R}^2$ the size of the control action at time-step k, and $(Q_y, Q_u) \in (\mathbb{R}^{2\times 2})^2$ the symmetric positive semi-definite weight matrices.

Figures 7–10 below show the performance of the RNN-MPC and NMPC controllers on both control scenarios for 250, 500, 1000 and 2000 hidden nodes respectively. Table 4 tabulates the performance index averaged over both control scenarios for each RNN-MPC controller for the same set of values for the number of hidden nodes.

Table 4. Performance of closed-loop RNN-MPC as a function of RNN nodes (two layers).

No. Nodes	Average Performance Index, $\mathcal{I}_{\text{avg}}$	Comments
250	93.7	Steady-state Offset
500	95.8	Steady-state Offset
1000	100.0	Desired Performance
2000	98.6	Steady-state Offset

The results show that the RNN-MPC controllers exhibited stability even with only 250 hidden nodes for both control scenarios, suggesting a robustness associated with the RNN-MPC combination.

For the RNN-MPC controller with 250 hidden nodes, while the NMPC controller exhibited no steady-state offset, this RNN-MPC controller showed significant steady-state offsets for both control scenarios. However, the initial transient response of the system under RNN-MPC control was similar to its NMPC benchmark for both scenarios, suggesting that the RNN succeeded in capturing the global dynamics of the system, and that 250 nodes may have not been sufficient for the model to learn the subtler dynamics that would have allowed the controller to bring the system precisely back to the set-point. A notable improvement in performance was observed with 500 nodes, although steady-state offsets remained for both scenarios.

The best closed-loop control performance was obtained with 1000 nodes, and this is consistent with the finding that this RNN had the best validation performance on the test data. The control performance of this RNN-MPC was in fact comparable to its NMPC counterpart, with an $\mathcal{I}_{\text{avg}}$ of 100.0. For 2000 nodes, a deterioration of control performance could be seen with the reappearance of the steady-state offsets for both scenarios, and this may be attributable to the RNN's poorer validation performance on the test data.

To assess the temporal performance of the RNN-MPC controller, 1000 model inputs were generated randomly and the average computational times required by the NMPC and the RNN-MPC controllers to generate a single prediction for use in the optimiser are tabulated in Table 5. The RNN for this temporal benchmarking has 1000 hidden nodes and two layers, and the NMPC generates predictions by using the Runge-Kutta method of order 5(4) to march the system forward in time. The predictions for both controllers were generated on an Intel(R) Xeon(R) CPU E5-1650 with 16 GB RAM.

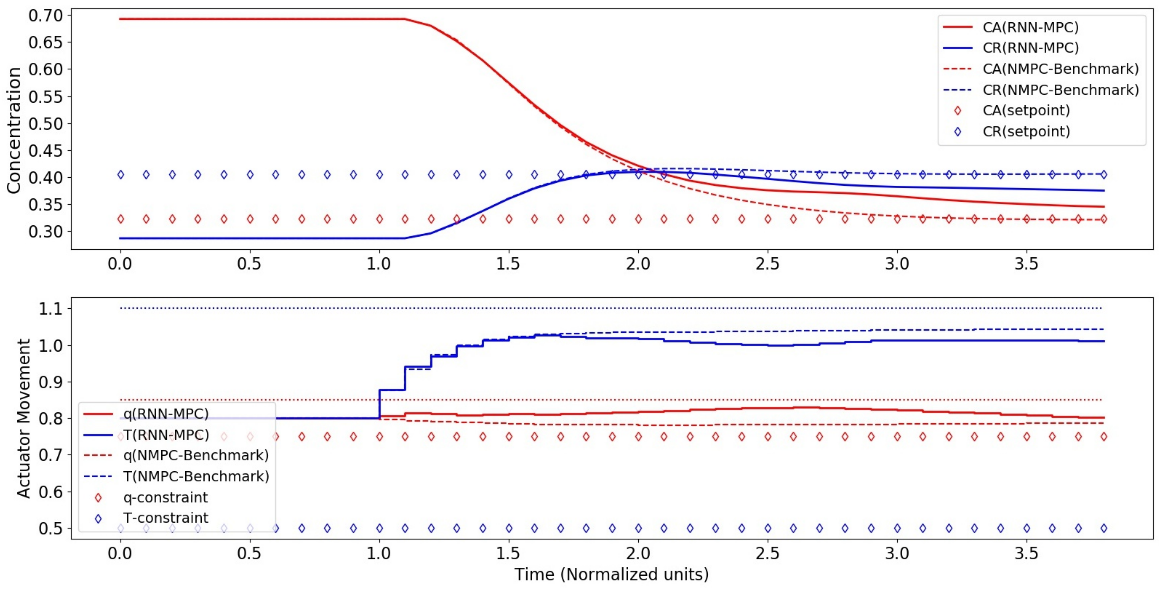
Table 5. Time required to generate a single prediction averaged over 1000 predictions.

	NMPC	**RNN-MPC**
Time required	1.55 ms	1.17 ms

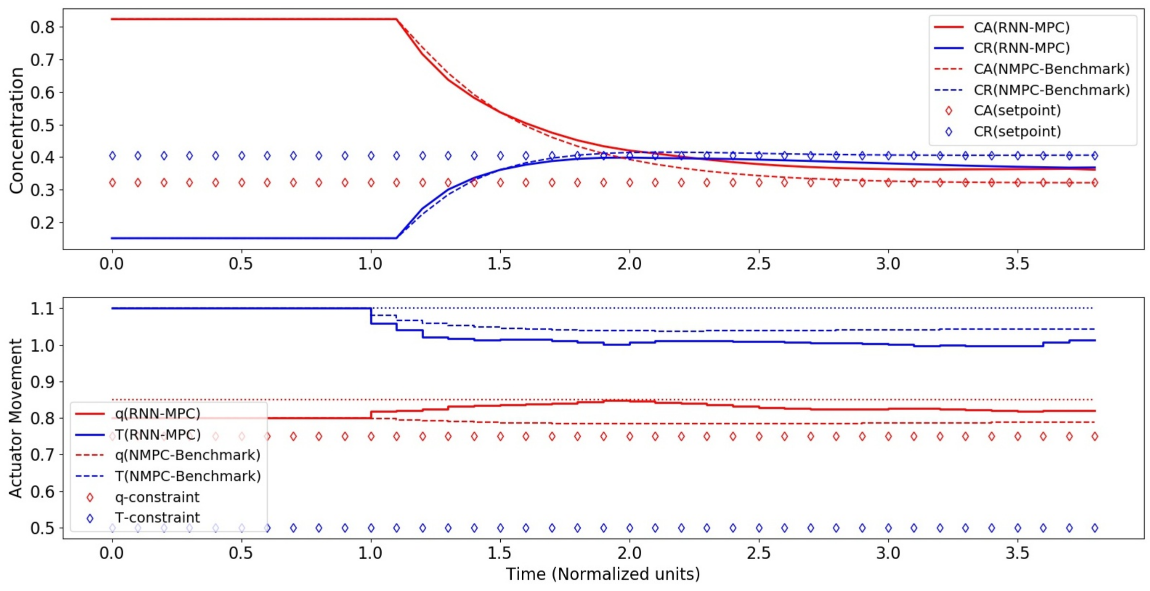
The results in Table 5 show that, for the simulation context described in the previous paragraph, the times required by the internal models of both controllers to generate a single prediction are of the same order of magnitude, with the RNN even outperforming its NMPC counterpart slightly. This testifies to the RNNs being a model that can generate predictions for MPC control at least as fast as the true plant model for this CSTR system.

This CSTR system is a comparatively simple dynamical system, and other systems can have more complex exact representations whose evaluation for prediction purposes may take a considerably longer time. Even though these models generate very good predictions, they would be too slow to evaluate to be useful for online control. Having observed from earlier discussions that stable control performance can be achieved if the RNN succeeds in capturing the global dynamics, imperfect predictions may in fact be tolerable if it entails an RNN that is easier to train and faster to evaluate. RNNs can therefore serve to balance the needs of MPC for good and quick predictions.

To summarise, even if the system identification by the RNNs does not perfectly describe the underlying plant dynamics, stable closed-loop control performance can be achieved as long as the global dynamics of the system are captured. Further refinements to the RNN can be made by optimising its hyperparameters, which in this study have been selected to be the number of hidden nodes and hidden layers, to obtain a configuration that is capable of capturing the subtler dynamics of the system while avoiding overfitting on the training set.

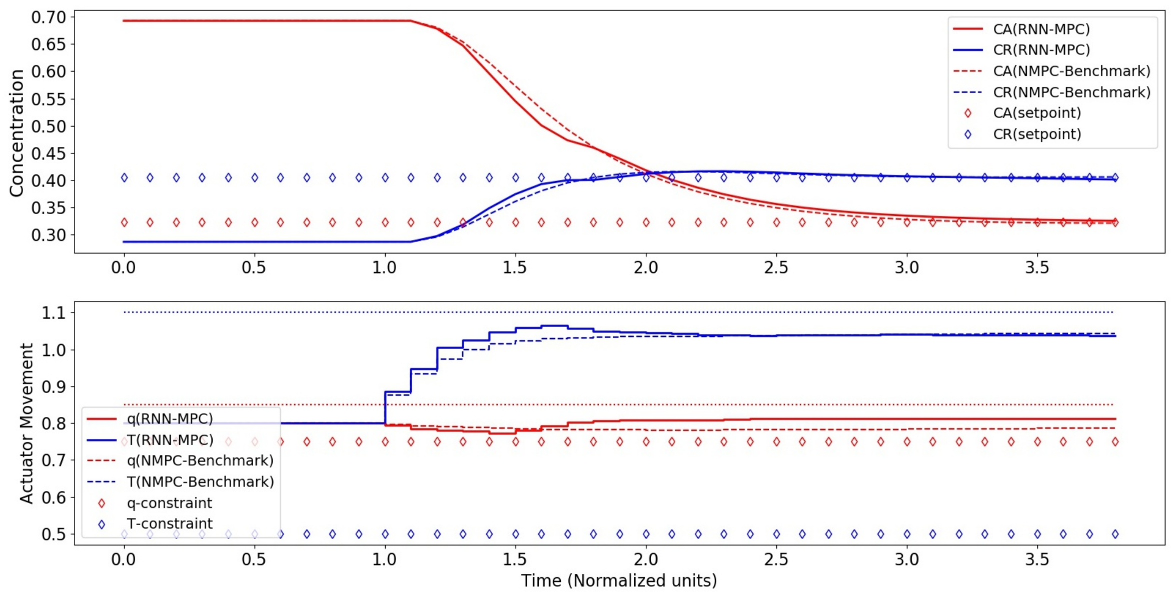


(**a**)

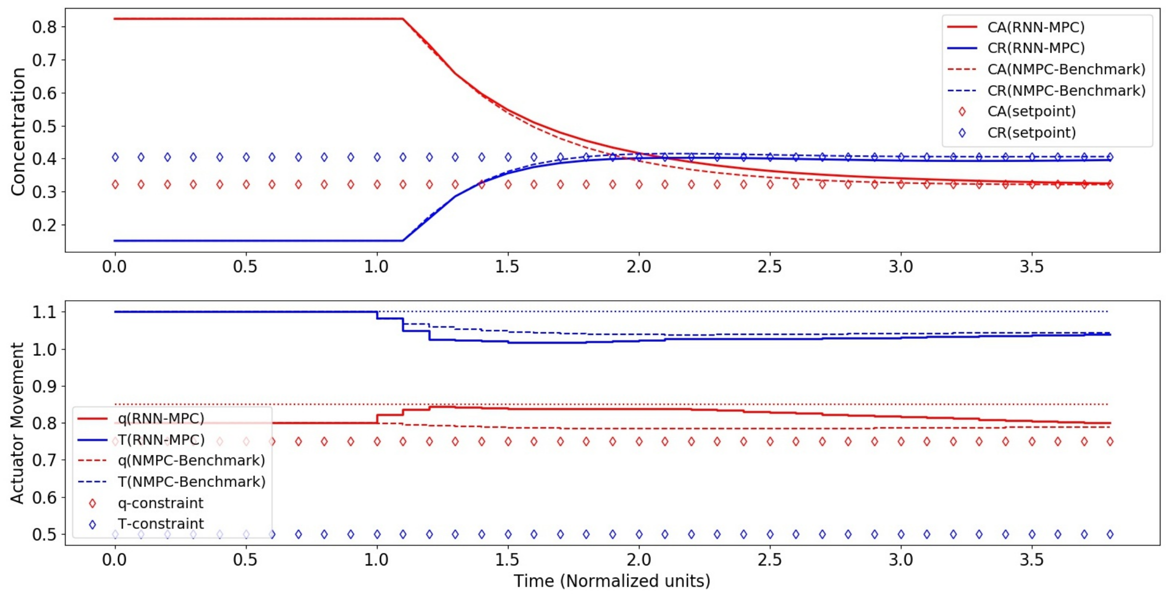


(**b**)

Figure 7. Control performance of RNN-MPC with 250 hidden nodes and two RNN layers. (**a**) Control performance for case I (start-up). (**b**) Control performance for case II (upset-recovery).

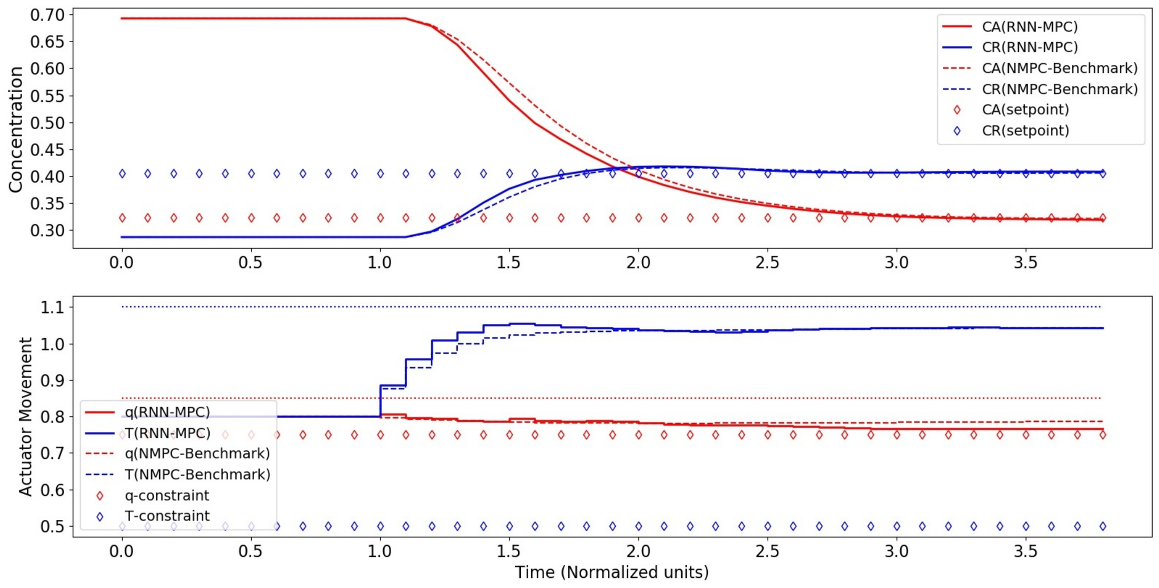


(**a**)

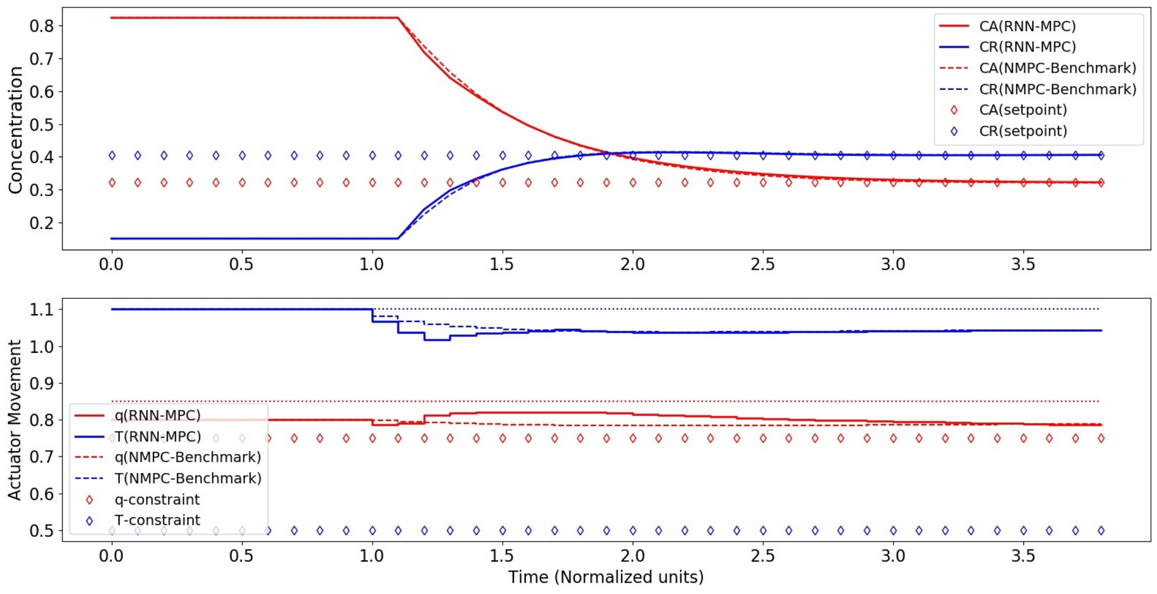


(**b**)

Figure 8. Control performance of RNN-MPC with 500 hidden nodes and two RNN layers. (**a**) Control performance for case I (start-up). (**b**) Control performance for case II (upset-recovery).

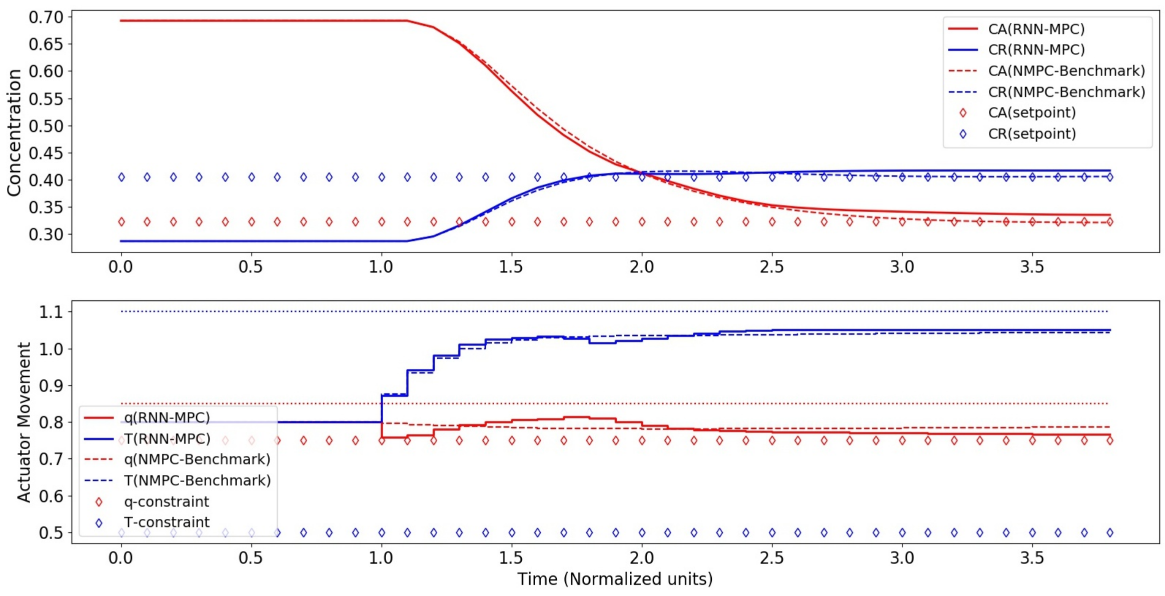


(**a**)

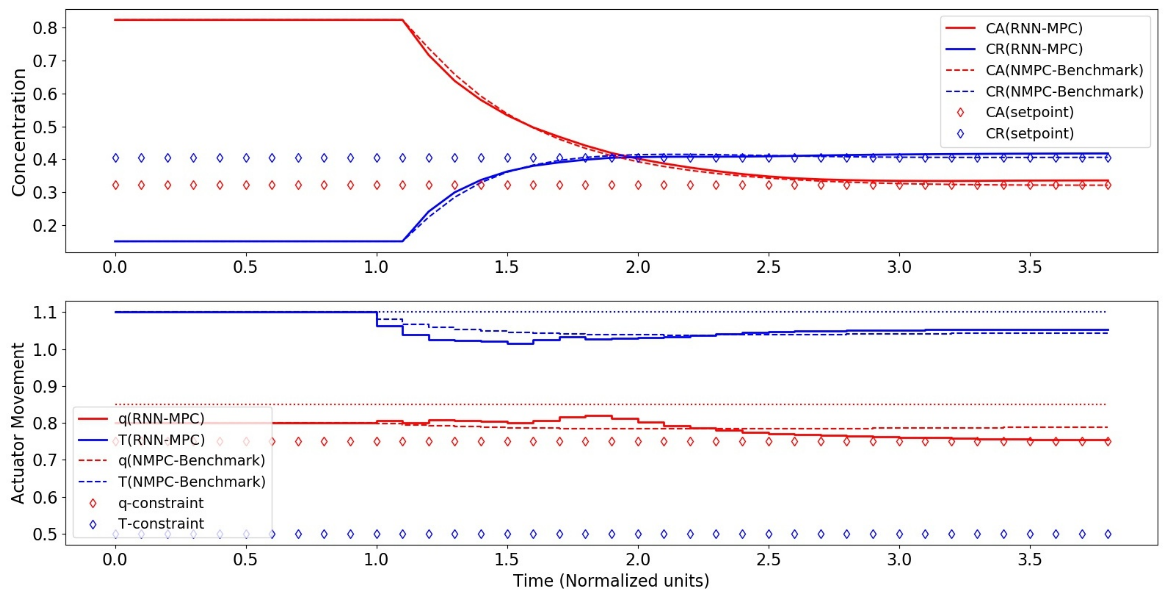


(**b**)

Figure 9. Control performance of RNN-MPC with 1000 hidden nodes and two RNN layers. (**a**) Control performance for case I (start-up). (**b**) Control performance for case II (upset-recovery).



(**a**)



(**b**)

Figure 10. Control performance of RNN-MPC with 2000 hidden nodes and two RNN layers. (**a**) Control performance for case I (start-up). (**b**) Control performance for case II (upset-recovery).

5. Conclusions and Future Research

In this study, a multiple-input multiple-output CSTR that houses a reaction exhibiting output dynamics was studied. The methods for generating the control-oriented RNNs and integrating their use into MPC controllers were articulated, and the performance of the RNN-MPC controllers was benchmarked against NMPC controllers that used the true plant model directly as its internal model.

Two control scenarios were simulated, one involving reactor start-up and the other involving upset-recovery, and the results showed that even if the RNNs do not perfectly describe the underlying plant dynamics, RNN-MPC controllers can give stable closed-loop control performance for these scenarios as long as the global dynamics of the system are captured. These RNNs can be refined through hyperparameter optimisation to obtain configurations that are capable of capturing the subtler dynamics of the system while avoiding overfitting on the training set, further augmenting their effectiveness as control-oriented models for MPC.

It is finally noted again that linear controllers with integral action perform poorly for systems exhibiting input multiplicities, and that rigorous physics-based models like the model used for the benchmark NMPC controller can be difficult to obtain in practice and may be too expensive to evaluate to be useful for online MPC control. The approach presented in this study exploits the ability of RNNs to capture the temporal dynamics of the system, and it was shown that RNN-based system identification methods can give control-oriented models for MPC that offer satisfactory performance for two important control scenarios. As the creation of these RNNs is a data-driven process which yields progressively more effective models with larger amounts of process and sensor data, these models serve indeed as an alternative source of control-oriented models that capture the dynamics of highly non-linear systems sufficiently well while being relatively easy to obtain and evaluate online. This opens promising avenues for the application of RNNs or other ML models in enabling continuous pharmaceutical manufacturing practices.

A potential way to bring this study forward is to compare different ML or RNN architectures for MPC control-oriented system identification. Measurement noise can be incorporated to simulate realistic sampling scenarios and evaluate the robustness to noise of the ML system identification and the ML-MPC control techniques. Cross-validated optimisation of more hyperparameters characterising the ML model, which can potentially include the regular dropout rates between hidden layers, the learning rates and gradient clipping values among others, as well as a more detailed study of the effects of the experimental sequences on the quality of the ML model trained, can also give future practitioners useful heuristics for creating and tuning MLs for their own purposes.

Future research can also involve extending the methodology to multiple CSTRs, and to reaction kinetics of increasing complexity and of more immediate relevance to the pharmaceutical industry. These can be done in view of an eventual real-world implementation of these RNN-MPC or ML-MPC control strategies, from which further comparisons of these controllers to conventional PID controllers, and other MPC strategies like linear MPC and offset-free MPC, can also be made.

Author Contributions: Wee Chin Wong and Xiaonan Wang conceived and designed the methodology; Wee Chin Wong and Ewan Chee performed the simulations; Wee Chin Wong, Ewan Chee, Jiali Li and Xiaonan Wang analyzed the data and wrote the paper.

Funding: The authors thank the MOE AcRF Grant in Singapore for financial support of the projects on Precision Healthcare Development, Manufacturing and Supply Chain Optimization (R-279-000-513-133) and Advanced Process Control and Machine Learning Methods for Enhanced Continuous Manufacturing of Pharmaceutical Products (R-279-000-541-114).

Acknowledgments: We would like to acknowledge S. A. Khan from the Department of Chemical and Biomolecular Engineering, National University of Singapore for his input on reaction kinetics and multiple fruitful discussions. We also thank the Pharma Innovation Programme Singapore (PIPS), which provided the source of motivation for this work.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Adam	Adaptive Moment Estimation
ANN	Artificial Neural Network
API	Active Pharmaceutical Ingredient
BPTT	Back-Propagation Through Time
CPP	Critical Process Parameter
CQA	Critical Quality Attribute
CSTR	Continuous-Stirred Tank Reactor
FBU	Feeding Blending Unit
LSTM	Long Short-Term Memory

MPC	Model Predictive Control
MV	Manipulated Vector
NLP	Non-Linear Programming
NMPC	Non-Linear MPC
PID	Proportional-Integral-Derivative (Control)
QbD	Quality by Design
QDMC	Quadratic Dynamic Matrix Control
RMSE	Root-Mean-Square Error
RNN	Recurrent Neural Network
RNN-MPC	RNN-based MPC
SLSQP	Sequential Least Squares Quadratic Programming

Appendix A. Kinetic Parameters for Plant Model

For this study, C_{A0} was assigned a value of 0.8. The vector of Arrhenius pre-exponentials, k_0, was defined as $[1.0, 0.7, 0.1, 0.006]'$, and the vector of normalized activation energies, $\frac{E}{RT_0}$, as $[8.33, 10.0, 50.0, 83.3]'$.

Appendix B. Long Short-Term Memory Cells

In conventional RNNs, the BPTT algorithm results in the partial derivative of the error function being multiplied numerous times to the weights corresponding to the various connections, or edges, of the RNN. Depending on the spectral radius of the recurrent weight matrix, this leads to either the vanishing or exploding gradients problem, which severely limits the learning quality of the network.

LSTM cells have been proposed to mitigate this issue. These cells use several gating functions, like the 'forget', 'input' and 'output' gating functions, that serve to modulate the propagation of signals between cells. This cell structure avoids the gradient vanishing or exploding problem.

The basic LSTM cell structure is mathematically expressed as follows in Equations (A1)–(A6) below:

$$h_k = o_k * \tanh\left(C_k\right) \tag{A1}$$

$$C_k = f_k * C_{k-1} + i_k * \tilde{C}_k \tag{A2}$$

$$\tilde{C}_k = \tanh\left(W_C[h_{k-1}, u_k]' + b_c\right) \tag{A3}$$

$$i_k = \sigma\big(W_i.[h_{k-1}, u_k]' + b_i\big) \tag{A4}$$

$$f_k = \sigma\big(W_f.[h_{k-1}, u_k]' + b_f\big) \tag{A5}$$

$$o_k = \sigma\big(W_o.[h_{k-1}, u_k]' + b_o\big) \tag{A6}$$

where $k \in \mathbb{Z}_+$ is the time index, $h_k \in \mathbb{R}^N$ the hidden state variable, and $u_k \in \mathbb{R}^2$ the input variable. $f_k \in [0,1]$, $i_k \in [0,1]$ and $o_k \in [0,1]$ are the 'forgetting', 'input' and 'output' gates respectively, and are characterised by their respective weight matrices and bias vectors, with $\sigma : \mathbb{R}^N \to [0,1]^N$ an activation function. The input gate controls the degree to which the cell state, represented by Equation (A2) and distinct from the hidden state variable, is affected by candidate information, and the output gate controls how this cell state affects other cells. The forget gate modulates the self-recurrent connection of the cell, allowing it thus to partially remember the previous cell state in a fashion similar to traditional RNNs. $*$ refers to a point-wise multiplication.

References

1. Lakerveld, R.; Benyahia, B.; Heider, P.L.; Zhang, H.; Wolfe, A.; Testa, C.J.; Ogden, S.; Hersey, D.R.; Mascia, S.; Evans, J.M.; et al. The application of an automated control strategy for an integrated continuous pharmaceutical pilot plant. *Org. Process Res. Dev.* **2015**, *19*, 1088–1100. [CrossRef]
2. Schaber, S.D.; Gerogiorgis, D.I.; Ramachandran, R.; Evans, J.M.B.; Barton, P.I.; Trout, B.L. Economic analysis of integrated continuous and batch pharmaceutical manufacturing: A case study. *Ind. Eng. Chem. Res.* **2011**, *50*, 10083–10092. [CrossRef]

3. Glasnov, T. *Continuous-Flow Chemistry in the Research Laboratory: Modern Organic Chemistry in Dedicated Reactors at the Dawn of the 21st Century*; Springer International Publishing: Basel, Switzerland, 2016; p. 119.
4. Gutmann, B.; Cantillo, D.; Kappe, C.O. Continuous-flow technology—A tool for the safe manufacturing of active pharmaceutical ingredients. *Angew. Chem. Int. Ed.* **2015**, *54*, 6688–6728. [CrossRef] [PubMed]
5. Poechlauer, P.; Colberg, J.; Fisher, E.; Jansen, M.; Johnson, M.D.; Koenig, S.G.; Lawler, M.; Laporte, T.; Manley, J.; Martin, B.; et al. Pharmaceutical roundtable study demonstrates the value of continuous manufacturing in the design of greener processes. *Org. Process Res. Dev.* **2013**, *17*, 1472–1478. [CrossRef]
6. Benyahia, B.; Lakerveld, R.; Barton, P.I. A plant-wide dynamic model of a continuous pharmaceutical process. *Ind. Eng. Chem. Res.* **2012**, *51*, 15393–15412. [CrossRef]
7. Susanne, F.; Martin, B.; Aubry, M.; Sedelmeier, J.; Lima, F.; Sevinc, S.; Piccioni, L.; Haber, J.; Schenkel, B.; Venturoni, F. Match-making reactors to chemistry: A continuous manufacturing-enabled sequence to a key benzoxazole pharmaceutical intermediate. *Org. Process Res. Dev.* **2017**, *21*, 1779–1793. [CrossRef]
8. Mascia, S.; Heider, P.L.; Zhang, H.; Lakerveld, R.; Benyahia, B.; Barton, P.I.; Braatz, R.D.; Cooney, C.L.; Evans, J.M.B.; Jamison, T.F.; et al. End-to-end continuous manufacturing of pharmaceuticals: Integrated synthesis, purification, and final dosage formation. *Angew. Chem. Int. Ed.* **2013**, *52*, 12359–12363. [CrossRef] [PubMed]
9. Brueggemeier, S.B.; Reiff, E.A.; Lyngberg, O.K.; Hobson, L.A.; Tabora, J.E. Modeling-based approach towards quality by design for the ibipinabant API step modeling-based approach towards quality by design for the ibipinabant API step. *Org. Process Res. Dev.* **2012**, *16*, 567–576. [CrossRef]
10. Mesbah, A.; Paulson, J.A.; Lakerveld, R.; Braatz, R.D. Model predictive control of an integrated continuous pharmaceutical manufacturing pilot plant. *Org. Process Res. Dev.* **2017**, *21*, 844–854. [CrossRef]
11. Rasoulian, S.; Ricardez-Sandoval, L.A. Stochastic nonlinear model predictive control applied to a thin film deposition process under uncertainty. *Chem. Eng. Sci.* **2016**, *140*, 90–103. [CrossRef]
12. Rasoulian, S.; Ricardez-Sandoval, L.A. A robust nonlinear model predictive controller for a multiscale thin film deposition process. *Chem. Eng. Sci.* **2015**, *136*, 38–49. [CrossRef]
13. Hussain, M.A. Review of the applications of neural networks in chemical process control simulation and online implementation. *Artif. Intell. Eng.* **1999**, *13*, 55–68. [CrossRef]
14. Cheng, L.; Liu, W.; Hou, Z.G.; Yu, J.; Tan, M. Neural-network-based nonlinear model predictive control for piezoelectric actuators. *IEEE Trans. Ind. Electron.* **2015**, *62*, 7717–7727. [CrossRef]
15. Xiong, Z.; Zhang, J. A batch-to-batch iterative optimal control strategy based on recurrent neural network models. *J. Process Control* **2005**, *15*, 11–21. [CrossRef]
16. Tian, Y.; Zhang, J.; Morris, J. Modeling and optimal control of a batch polymerization reactor using a hybrid stacked recurrent neural network model. *Ind. Eng. Chem. Res.* **2001**, *40*, 4525–4535. [CrossRef]
17. Mujtaba, I.; Hussain, M. *Applications of Neural Networks and Other Learning Technologies in Process Engineering*; Imperial College Press: London, UK, 2001.
18. Nagy, Z.K. Model based control of a yeast fermentation bioreactor using optimally designed artificial neural networks. *Chem. Eng. J.* **2007**, *127*, 95–109. [CrossRef]
19. Alanqar, A.; Durand, H.; Christofides, P.D. On identification of well-conditioned nonlinear systems: Application to economic model predictive control of nonlinear processes. *AIChE J.* **2015**, *61*, 3353–3373. [CrossRef]
20. Wang, X.; El-Farra, N.H.; Palazoglu, A. Proactive Reconfiguration of Heat-Exchanger Supernetworks. *Ind. Eng. Chem. Res.* **2015**, *54*, 9178–9190. [CrossRef]
21. Byeon, W.; Breuel, T.M.; Raue, F.; Liwicki, M. Scene labeling with LSTM recurrent neural networks. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3547–3555.
22. Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
23. Lee, J.H.; Shin, J.; Realff, M.J. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Comput. Chem. Eng.* **2018**, *114*, 111–121. [CrossRef]
24. Rehrl, J.; Kruisz, J.; Sacher, S.; Khinast, J.; Horn, M. Optimized continuous pharmaceutical manufacturing via model-predictive control. *Int. J. Pharm.* **2016**, *510*, 100–115. [CrossRef] [PubMed]
25. Rawlings, J.B.; Mayne, D.Q. *Model Predictive Control: Theory and Design*; Nob Hill: Madison, WI, USA, 2009.
26. Tatjewski, P. *Advanced Control of Industrial Processes, Structures and Algorithms*; Springer: London, UK, 2007.

27. Garcia, C.E.; Morshedi, A. Quadratic programming solution of dynamic matrix control (QDMC). *Chem. Eng. Commun.* **1986**, *46*, 73–87. [CrossRef]
28. Pan, Y.; Wang, J. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Trans. Ind. Electron.* **2012**, *59*, 3089–3101. [CrossRef]
29. Seyab, R.A. Differential recurrent neural network based predictive control. *Comput. Chem. Eng.* **2008**, *32*, 1533–1545. [CrossRef]
30. Koppel, L.B. Input multiplicities in nonlinear, multivariable control systems. *AIChE J.* **1982**, *28*, 935–945. [CrossRef]
31. Seki, H.; Ooyama, S.; Ogawa, M. Nonlinear model predictive control using successive linearization—Application to chemical reactors. *Trans. Soc. Instrum. Control Eng.* **2004**, *E-3*, 66–72.
32. Bequette, B.W. Non-linear model predictive control : A personal retrospective. *Can. J. Chem. Eng.* **2007**, *85*, 408–415. [CrossRef]
33. Kingma, D.P.; Ba, J. Adam: A Method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
34. Pascanu, R.; Gulcehre, C.; Cho, K.; Bengio, Y. How to construct deep recurrent neural networks. *arXiv* **2013**, arXiv:1312.6026.