Generalized System Identification for Nonlinear MPC of Highly Nonlinear MIMO Systems ⋆

Ewan Chee and Xiaonan Wang *

* *Department of Chemical & Biomolecular Engineering, Faculty of Engineering, National University of Singapore, 4 Engineering Drive 4, Singapore 117585, Singapore (e-mail: {chejxec, chewxia}@nus.edu.sg).*

Abstract: Advanced model-based control strategies like Model Predictive Control can offer superior control of key process variables for multiple-input multiple-output systems. The quality of the system model is critical to controller performance and should adequately describe the process dynamics across its operating range while remaining amenable to fast optimization. Mechanistic models might be difficult to construct or might evaluate too slowly for fast-sampling purposes, while System Identification methods that yield linear models might only offer effective control within a limited range. This work thus articulates an integrated System Identification procedure for deriving black-box nonlinear continuous-time multiple-input multiple-output system models for Nonlinear Model Predictive Control. In this piece, seven candidate models were trained on data sets generated from two different methods and integrated into a Nonlinear Model Predictive Controller for a highly nonlinear Continuous Stirred Tank Reactor system. This procedure successfully identified system models that enabled effective control in both servo and regulator problems across wider operating ranges, given that the data set is sufficiently uniform across the operating range. These results were subsequently built upon in a follow-up work. This demonstration of how such system models could be identified for Nonlinear Model Predictive Control without prior knowledge of system dynamics opens further possibilities for direct data-driven methodologies for model-based control which, in the face of process uncertainties or modeling limitations, allow rapid and stable control over wider operating ranges.

Keywords: Nonlinear model predictive control, black-box modeling, system identification, machine learning, industrial applications of process control.

1. INTRODUCTION

The manufacturing industries are perpetually challenged by the need to develop processes that convert raw materials into useful products in more reliable and sustainable ways, and this has led to strong and sustained interest in the research of effective control strategies (**?**). Achieving tight control across a process' entire operating range is especially challenging, with conventional methods like Proportional-Integral-Derivative (PID) controllers being ill-equipped to handle systems that are highly nonlinear around their operating points (**?**). This motivates the study of advanced model-based control strategies like Model Predictive Control (MPC) for these systems.

MPC incorporates knowledge about the process through a system model and solves a dynamic optimization problem at each time step to yield an optimal control sequence, applying the first control action in the sequence to the system before proceeding to re-solve the problem at the next time step. MPC natively supports Multiple-Input Multiple-Output (MIMO) formulations and also allows system constraints to be directly included in the problem formulation, explicitly representing process limits that ensure plant safety and reliability. MPC has seen successful application in industries ranging from oil refining to chemical production (**?**).

The system model plays a determinant role in MPC, and should adequately describe the process dynamics across its operating range while being simple enough to allow fast optimization (**?**). These models could either be constructed from first principles, which might be difficult and costly to develop for complex processes, or derived from empirical data through System Identification (SysID). These models could be linear, nonlinear, hybrid or nonparametric, among others, and MPC products have typically relied on linear models to exploit algorithms that optimize efficiently (**?**). Successful applications of linear MPCs (LMPCs) for continuous pharmaceutical tablet manufacturing processes were reported in **?** and **?**.

LMPC might struggle to offer effective control outside a limited operating window (**?**). Nonlinear MPC (NMPC), by employing nonlinear system models and system constraints, can allow better process representation over a

⋆ 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).

wider operating range. This comes however at the expense of needing to solve non-convex optimization problems quickly and precisely (**?**).

Recent advances in computing and statistics have promised to unlock the potential of data-driven Machine Learning (ML) methods. This could improve the viability of black-box SysID methods in deriving system models for NMPC that deliver effective control over a wider operating range. This is useful when process understanding is limited to begin with, or when first-principles models take prohibitively long to evaluate for fast-sampling applications. **??** demonstrated the use of RNNs and ensemble techniques to generate MIMO system models for NMPC of a CSTR system, showing that these NMPCs performed better than LMPCs at rejecting process disturbances and showcasing parallel implementations of these that considerably reduced solution times.

This work endeavors to further explore the potential of ML in identifying MIMO system models from empirical data which enable NMPC to deliver rapid and stable control over a wider operating range, evaluating *both* control performance and solution times on *both* set-point tracking and disturbance rejection scenarios to ensure the proposed solution's practicality in real-world industrial settings. We articulate an integrated, overarching approach for learning the dynamics of continuous-time systems under control. We subsequently apply this framework to develop ML-NMPCs for a CSTR system exhibiting highly nonlinear dynamics, benchmarking these controllers' performance against an NMPC with the exact system model.

This work is organized as follows. Section 2 frames this study by defining the CSTR system and the control scenarios. Section 3 explicates the generalized black-box SysID methodology and presents the MPC formulation. Section 4 presents results and discussions.

2. PROBLEM DEFINITION

In the following exposition, $x \in \mathcal{K}^q$ is a column vector, with $\mathcal{K}$ representing some field and $q \geq 1$. The notation $[a, b]$ represents a row vector when $(a, b) \in \mathcal{K}^2$, and the T subscript represents the vector transpose operation, such that $[a, b]^T$ is a column vector. The $+\!\!+$ symbol represents column vector concatenation, and it returns another column vector, such that $x_1 +\!\!+ x_2 = [x_1^T, x_2^T]^T$ and $+\!\!+_{i\in\{1,...,n\}} x_i = [x_1^T, ..., x_n^T]^T$, where $x_i \in \mathcal{K}^{q_i}$. An $n \times p$ matrix with values in $\mathcal{K}$ is denoted $\mathcal{M}_{n,p}(\mathcal{K})$, with n-order square matrices are $\mathcal{M}_n(\mathcal{K})$. The $+\!\!+$ symbol is also used to represent the vertical concatenation of matrices with the same number of columns, such that $+\!\!+_{i\in\{1,...,n\}} B_i = [B_1^T, ..., B_n^T]^T$, where $B_i \in \mathcal{M}_{q_i,p}(\mathcal{K})$. Sets are represented by $\{1, ..., k\}$. Closed intervals are defined as $[\alpha; \beta]$, with α and β real numbers. Given $x(t) \in \mathcal{K}^q$ a continuous-time vector-valued quantity, where $t \in \mathbb{R}_+$, $x[k]$ represents its value associated with the discrete time-step k, such that $x[k] = x(t_k)$, where t_k is the real time associated with the discrete time-step k. Δ finally represents the difference operator, such that $\Delta x[k] := x[k + 1] - x[k]$.

2.1 Plant Model

The single CSTR system for this study houses the reaction described in Eq. 1, where A is the feed species, R the desired product, and S the undesired by-product. Eqs. 2 to 4 contain non-dimensionalized expressions for the system's dynamical behavior:

$$A \underset{k_4}{\overset{k_1}{\rightleftarrows}} R \underset{k_3}{\overset{k_2}{\rightleftarrows}} S \tag{1}$$

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

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

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

where $C_{i\in\{A,R,S\}} \in \mathbb{R}_+$ is the species' reactor concentration, $C_{A0} \in \mathbb{R}_+$ the feed concentration of A, $q \in \mathbb{R}_+$ the feed and the exit flow rate, $k_{j\in\{1,...,4\}} \in \mathbb{R}_+$ the reaction rate constants, $k_{0,j\in\{1,...,4\}} \in \mathbb{R}_+$ the Arrhenius pre-exponential constants, $[\frac{E}{RT_0}]_{j\in\{1,...,4\}} \in \mathbb{R}_+$ the normalized activation energies, and $T \in \mathbb{R}_+$ the system temperature. q and T are the manipulated variables in this study.

The system's state representation is as follows:

$$\dot{x} = f(x, u) \tag{5}$$

$$y = I_2 x + \epsilon \tag{6}$$

where $x := [C_A, C_R]^T$ and $u := [q, T]^T$ are the state and control vectors respectively, y the observed vector with I_2 a size-2 identity matrix, and $f : \mathbb{R}_+^2 \times \mathbb{R}_+^2 \to \mathbb{R}^2$ a general nonlinear function. ϵ is a random variable corresponding to measurement noise.

Fig. 1 shows the system's steady-state conditions as a function of T when q is 0.8 and at its nominal point. To simplify the system, it is assumed that the low-level flow and temperature control loops have negligible dead times and present no steady-state errors. The system's set-point was selected to maximize the ratio of C_R to C_A. This corresponds also to maximizing product yield and minimizing any downstream separations costs. It can be observed that strong non-linearities exist in how both C_A and C_R vary with T at the set-point, which suggest that linear controllers could perform well only within a narrow operating range. There is also input multiplicity, since two different T values could yield the same value for C_R, which presents problems for stable control in practice (**?**). Process control of this system at this set-point serves thus as a relevant and interesting control problem.

2.2 Control Scenarios

Controller performance was evaluated based on 3 scenarios. The first scenario was a servo control problem representing $\pm 5\%$ step changes to the c_R set point. This step size was selected because the control actions needed to stabilize the system at the new set-points are large enough to pose a control problem which is challenging enough to be studied meaningfully. Each step lasted for 5 time units (tus) for a total experimental length of 20 tus. The second and third scenarios were regulator control

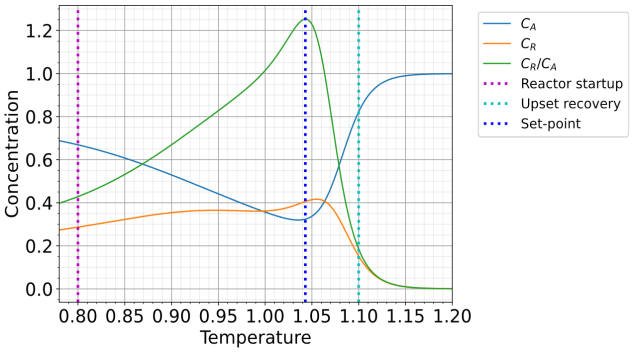

Fig. 1. System steady state conditions ($q = 0.8$).

problems corresponding to "reactor startup" and "upset recovery" respectively. These two process disturbances lay at opposite sides of the set-point, with the former having low T and C_R values and the latter having an abnormally high T value. A process controller would therefore need to optimize correctly for T values within an extended range of 0.8 to 1.1 to perform well in both these scenarios. All experimental runs for these two scenarios lasted for 5 tus.

3. METHODOLOGY

3.1 Nonlinear MPC Control

We define the cost function J at a given time-step $k \in \mathbb{N}$:

$$J(\Delta U) := \text{tr}((Y - Y^*)^T Q(Y - Y^*) + \Delta U^T R \Delta U) \quad (7)$$

where $\Delta U := [\Delta u[k], \Delta u[k+1], ..., \Delta u[k+m-1]]^T$ is the control sequence over the control horizon $m \in \mathbb{N}^*$, $Y := [y[k+1|k], ..., y[k+p|k]]^T$ the state trajectory over the prediction horizon $p \in \mathbb{N}^*$, Y^* the set-point over p, and Q and R diagonal non-negative weight matrices whose coefficients reflect the relative importance of the corresponding terms in the cost function. $\text{tr}(\cdot)$ represents the trace of the matrix. The control actions are applied in a Zero-Order Hold (ZOH) fashion throughout the sampling interval, such that $u(t) = u[k], \forall t \in [t_k; t_{k+1}[$.

The NMPC problem formulation is thus as follows:

$$\min_{\Delta U} J(\Delta U) \text{ s.t.} \quad (8)$$

$$Y = \mathcal{F}(x[k], u[k], \Delta U +\!\!+ 0_{p-m,2}) \quad (9)$$

$$\Delta u_{min} \leq \Delta u[l] \leq \Delta u_{max}, \forall l \in \{k, ..., k+m-1\} \quad (10)$$

$$u_{min} \leq u[l] \leq u_{max}, \forall l \in \{k, ..., k+m-1\} \quad (11)$$

$$y_{min} \leq y[l] \leq y_{max}, \forall l \in \{k+1, ..., k+p\} \quad (12)$$

where $\mathcal{F}$ is a nonlinear function generating the state trajectory given the system's initial state and the control sequence, and $0_{p-m,2}$ a zero matrix of dimension $(p-m) \times 2$. The concatenation of $0_{p-m,2}$ is equivalent to defining $\Delta u[l] = 0, \forall l \in \{k+m, ..., k+p-1\}$, then appending these zero values to ΔU such that the resulting matrix has p rows. u thus remains constant after the control horizon.

We note that $\mathcal{F}$ can be generated from f as Eq. 13 shows:

$$\mathcal{F}(x[k], u[k], \Delta U +\!\!+ 0_{p-m,2}) = [\mathcal{G}(1), ..., \mathcal{G}(p)]^T \quad (13)$$

where $\mathcal{G}(l) := x(t_k) + \int_{t_k}^{t_{k+l}} f(x(\tau), u(\tau))d\tau, \forall l \in \{1, ..., p\}$. Eqs. 10 to 12 represent constraints on the control actions' rates of change, constraints on their magnitude, and constraints on the system's output, respectively.

This NMPC problem is a constrained nonlinear optimization problem which can be solved using active set or interior point methods. This problem was solved using the Sequential Least Squares Quadratic Programming method which was implemented in `optimize.minimize` from *SciPy* (version 1.5). The constraints on the control actions' magnitude and system outputs, where are Eqs. 11 and 12 respectively, were manifested as soft constraints, with the coefficients of their penalty terms in the objective function taken to be 1000. The nonlinear optimizer was initialized with values sampled from $\mathcal{U}([-10^{-3}, 10^{-3}])$ at the first time step and was warm-started with the solution from the previous time step in subsequent time steps. For this work, the step size of the controller $h_{NMPC} = 0.1$ tu.

3.2 Regression Pipeline

The system models for NMPC are general nonlinear functions whose form might not be known *a priori*. The regression problem is thus concerned with inferring them from the available data, which consists of $\tilde{X} := [\tilde{x}_i]^T_{i \in \{1,...,N\}}$ the design matrix with $\{\tilde{x}_i\}_{i \in \{1,...,N\}}$ the data set of size $N \in \mathbb{N}^*$, and $\tilde{Y} := [\tilde{y}_i]^T_{i \in \{1,...,N\}}$ the labels matrix with $\{\tilde{y}_i\}_{i \in \{1,...,N\}}$ the set of labels associated to $\{\tilde{x}_i\}$.

The full data set is split into training, validation and test sets. In this work, 60% of the data set is assigned to the training set, 20% to the validation set and 20% to the test set. We note I_{train}, I_{val} and I_{test} the set of indices of $\tilde{X}$ corresponding to these sets, with $\text{card}(I_{train}) = N_{train}$, $\text{card}(I_{val}) = N_{val}$ and $\text{card}(I_{test}) = N_{test}$, such that $N_{train} + N_{val} + N_{test} = N$. $\tilde{X}_{train} := [\tilde{x}_i]^T_{i \in I_{train}}$ is the training design matrix with $\tilde{Y}_{train} := [\tilde{y}_i]^T_{i \in I_{train}}$ its labels matrix. $\tilde{X}_{val}$, $\tilde{Y}_{val}$, $\tilde{X}_{test}$ and $\tilde{Y}_{test}$ are defined similarly.

Data preprocessing encompasses a broad set of strategies to prepare the data set for the model training phase in ways that facilitate the learning of models that generalize well. It generally involves data cleaning, which serves to improve the quality of the data set by removing outliers that might have resulted from sensor faults or data input errors, and feature engineering, which can take the form of feature transformation, generation, or extraction.

In this work, we demonstrate the generation of polynomial and interaction features that represent higher-order and coupled nonlinear interactions. We pose $\tilde{X} = [\tilde{X}_j]_{j \in \{1,...,N_f\}}$, where $\tilde{X}_{j \in \{1,...,N_f\}}$ are feature vectors with $N_f \in \mathbb{N}^*$ the number of features. Equation 14 formalizes the feature generation procedure of degree $d \in \mathbb{N}^*$:

$$\tilde{X}^{(d)} = [\tilde{X}_1^{p_1} \tilde{X}_2^{p_2} \dots \tilde{X}_{N_f}^{p_{N_f}}]_{\substack{(p_1,...,p_{N_f}) \in \{1,...,\ell\}^{N_f} \text{ s.t.} \\ p_1 + \dots + p_{N_f} \leq \ell, \forall \ell \in \{1,...,d\}}} \quad (14)$$

Data standardization is also performed to allow all features to be considered equally during model training. Equation 15 describes the data standardization procedure:

$$\tilde{X}^{(d)}_{\cdot,std} = \left[\frac{\tilde{X}^{(d)}_{\cdot,j} - \bar{\tilde{X}}^{(d)}_{\star,j}}{\hat{\sigma}_{\star,j}} \right]_{j \in \{1,...,N_f^{(d)}\}} \quad (15)$$

where $\tilde{X}^{(d)}_{\cdot,j}$ represents the j-th column of $\tilde{X}^{(d)}$, $\bar{\tilde{X}}^{(d)}_{\star,j}$ the sample mean of $\tilde{X}^{(d)}_{\star,j}$, $\hat{\sigma}_{\star,j}$ the population standard

deviation of $\tilde{X}^{(d)}_{\star,j}$, and $N_f^{(d)} \in \mathbb{N}^*$ the total number of polynomial and interaction features. $\cdot$ is a placeholder referring either to the training, validation or test sets. If $\cdot$ refers to either $\tilde{X}_{train}$ or $\tilde{X}_{val}$, $\star$ refers to $\tilde{X}_{train}$. If $\cdot$ refers to $\tilde{X}_{test}$, $\star$ refers to $\tilde{X}_{train} \cup \tilde{X}_{val}$. This mapping of $\cdot$ to $\star$ is done to prevent data leakage.

The model training phase involves solving for a prediction function f such that $\tilde{y} \approx f(\tilde{x})$. Suppose that we have n candidate models, i.e. n different regression forms for f. For each $f_{i\in\{1,...,n\}}$, we solve the MSE minimization problem in Eq. 16:

$$\hat{\theta}_i = \underset{\theta_i \in \Theta_i}{\operatorname{argmin}} \frac{1}{N_{train}} \sum_{j \in I_{train}} (\tilde{y}_j - f_{i,\theta_i}(\tilde{x}_j))^2 \tag{16}$$

where Θ_i represents the parameter space for f_i. $f_i := f_{i,\hat{\theta}_i}$ finally represents the solution to this regression problem. Let $\eta_i \in H_i$ be the set of hyperparameters for f_i. Hyperparameter Optimization (HO) involves training f_i with different η_i, then selecting $\hat{\eta}_i$ which satisfies Eq. 17:

$$\hat{\eta}_i = \underset{\eta_i \in H_i}{\operatorname{argmin}} \frac{1}{N_{val}} \sum_{j \in I_{val}} (\tilde{y}_j - f_{i,\hat{\theta}_i;\eta_i}(\tilde{x}_j))^2 \tag{17}$$

Bayesian Optimization (BO) was employed in this work for HO, and the *Optuna* (version 2.0.0) package in Python was used in this work.

3.3 Regression Formulation for Generalized Black-Box Estimation in System Identification for NMPC

As we propose directly learning a representation for f in the continuous-time state representation shown in Eq. 5, we pose $\tilde{x}_j := [C_A, C_R, q, T]^T_j$ representing the features for data point $j \in I_{train}$, with $\tilde{y}_j := \left[\frac{dC_A}{dt}, \frac{dC_R}{dt}\right]^T_j$ the associated labels. To extract $\tilde{y}_j$ from $\tilde{x}_j$, the difference between the concentration measurements before and after the ZOH is taken:

$$\tilde{y}_j = \left[\frac{C_{A,j}(h) - C_{A,j}(0)}{h}, \frac{C_{R,j}(h) - C_{R,j}(0)}{h}\right]^T \tag{18}$$

where the time at the start of the ZOH is taken to be zero. h is taken as 0.1 time units (tus).

The regression forms f_i studied in this work are polynomial regression, Decision Tree (DT) regression, k-Nearest Neighbors (kNN) regression, Ensemble regression, Extra Trees (ET) regression, Gradient Boosted (GB) regression and Random Forest (RF) regression.

3.4 Data Generation Procedure

The CSTR system is numerically simulated with a linear multistep method implemented through the LSODA option in `integrate.solve_ivp` in the *SciPy* package (version 1.5). To simulate measurement error, a Gaussian noise $\epsilon \sim \mathcal{N}(0, 10^{-3})$ is included in every measurement of C_A and C_R. Given the range of values for C_A and C_R, this corresponds to a measurement error of ∼1%.

This study studies two methods for generating the data set from simulation experiments. The "Uniform" method involves randomly sampling $\tilde{x}_j$ from $\mathcal{U}([\tilde{x}_{j,min}, \tilde{x}_{j,max}])$, where $\mathcal{U}$ represents a Uniform distribution. A random sample of size N is generated, and the simulation is evolved for h tus for each point $j \in \{1, ..., N\}$. This method ensures that the resulting data set uniformly covers the controller's operating range, allowing ML models that perform well within this entire range to be learnt more easily. However, as this procedure involves impeccable system preparation and measurement, it is highly unfeasible in practice. This method serves therefore as a limiting case.

The "Oscillatory" method involves oscillating q and T at different frequencies within predefined bounds around specified values, then evolving the simulation with this control sequence. The length of one experiment was kept to around 5 tus, with the number of cycles within the ranges of q and T varying from 0 to 5. Experiments were also performed for different starting values and different ranges for q and T. Given an experiment $([C_A, C_R, q, T]^T_i)_{i\in 0,...,N_{sim}}$ with $N_{sim} \in \mathbb{N}^*$ the total number of time steps, $\tilde{x}_j := [C_A, C_R, q, T]^T_j$ and $\tilde{y}_j := \left[\frac{C_{A,j+1} - C_{A,j}}{h}, \frac{C_{R,j+1} - C_{R,j}}{h}\right]^T, \forall j \in \{0, ..., N_{sim} - 1\}$. The data from all the experiments was then combined and processed to yield $\tilde{X}$ and $\tilde{Y}$.

3.5 ML-NMPC Integration

After obtaining the system models from the black-box SysID pipeline, they are integrated into the ML-NMPC and used to generate predictions for dC_A/dt and dC_R/dt, which themselves are used to evolve C_A and C_R in time. Tuning of the ML-NMPC control parameters is done before testing its closed-loop performance on the case study. In this work, this consists of modifying m to minimize the Weighted Integrated Absolute Error (WIAE) for a ±5% set-point step experiment:

$$\text{WIAE} := \int_{t_0}^{t_f} W \|y(t) - y^*(t)\|_1 dt \tag{19}$$

where $W \in \mathcal{M}_2(\mathbb{R}_+)$ is a weight matrix, and $t_0 \in \mathbb{R}_+$ and $t_f \in \mathbb{R}_+$ the start and end times of the experiment respectively. $p = 5$ was taking to be fixed in this work. The WIAE was selected as the figure of merit for its simple interpretation, though other controller tuning statistics that explicitly include ΔU or that have other forms could also be used without any loss of generality of this procedure. In this study, $W = \text{diag}([1, 5])$, so C_R deviations are punished 5 times more than C_A deviations.

4. RESULTS AND DISCUSSION

4.1 Nonlinear MPC with Exact System Model

Fig. 2 demonstrates stable closed-loop performance of the NMPC for all three control scenarios, validating the use of NMPC as a benchmark for effective control of this system. The step experiment, startup, and recovery scenarios took 7.77 s, 2.75 s and 2.87 s to solve respectively, demonstrating average per-iteration solution times in the order of 0.1 ms.

4.2 Black-Box Estimation in System Identification for NMPC with Uniform Data Set

Fig. 3 shows the validation and test scores for the tuned models for each candidate model on the Uniform data set.

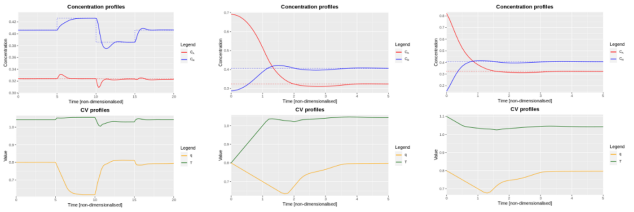

Fig. 2. NMPC in the step (left), startup (center) and recovery (right) experiments.

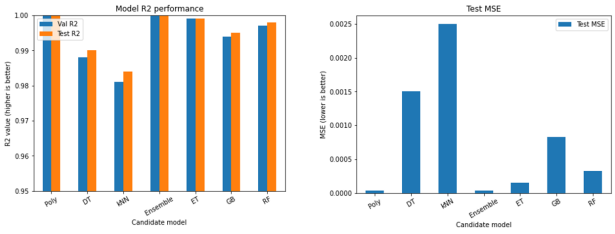

Fig. 3. R^2 scores (left) and MSE values (right) for the Uniform data set.

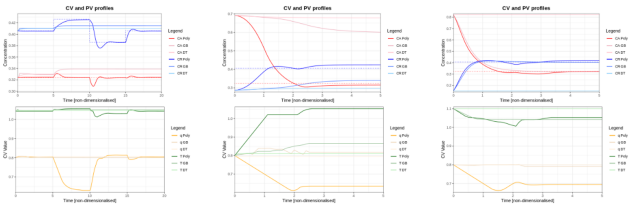

Fig. 4. ML-NMPC on Uniform data set in the step (left), startup (center) and recovery (right) experiments.

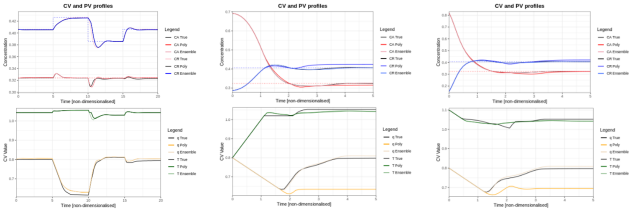

Fig. 5. ML-NMPC on Uniform data set against exact NMPC in the step (left), startup (center) and recovery (right) experiments.

For all candidate models except kNN, the validation and test R^2 values exceeded 0.98, suggesting that these models generally returned good estimates for dC_A/dt and dC_R/dt given the current system state and control action.

Fig. 4 shows ML-NMPC performance with system models from polynomial regression, GB and DT, while Fig. 5 compares the NMPC with the exact plant model against ML-NMPCs with polynomial regression and bagged regression models. Fig. 6 plots the results. Solving the ML-NMPC problem exceeded 20 mins for the remaining candidate models, suggesting that they evaluated too slowly to be feasible for fast-sampling applications.

Closed-loop control performance of the ML-NMPC with either polynomial models or bagged models was stable and approached the exact NMPC benchmark, with effective set-point tracking and disturbance rejection, and no permanent error. The ML-NMPC with the polynomial model also demonstrated average per-iteration solution times in the order of 1 ms, though it remains an order-of magnitude slower than the exact NMPC. For the bagged model, evaluation times grew to twice that of the polynomial model. It was observed that the added model complexity from bagging did not translate to tighter control, with this possibly indicating model over-fitting.

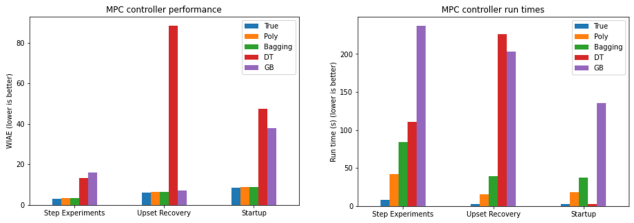

Fig. 6. Performance comparisons for ML-NMPC with the Uniform data set.

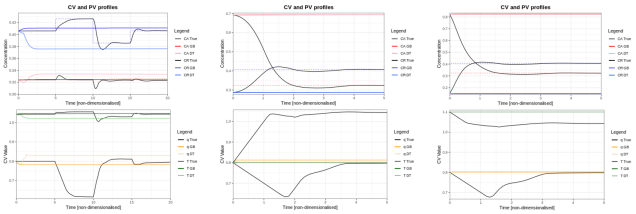

Fig. 7. ML-NMPC against NMPC on Oscillatory data set in the step (left), startup (center) and recovery (right) experiments.

For DT and GB, which are tree-based models, control performance was extremely poor except for GB in the upset recovery scenario. This observation might be attributed to two reasons. Firstly, DT and GB showed poorer validation and test scores than the polynomial models, which might suggest that, while R^2 values above 0.98 were ultimately achieved, the estimates for dC_A/dt and dC_R/dt from these tree-based models were still not good enough for effective ML-NMPC control. Secondly, as tree-based models are piece-wise constant functions (**?**), optimizers like SLSQP, which depend on good local gradient or Hessian approximations, might not function well.

4.3 Black-Box Estimation in System Identification for NMPC with Oscillatory Data Set

The candidate models were trained and tested with the Oscillatory data set, with the hyperparameters for each candidate model identical to those learned in Section 4.2. Fig. 7 compares the exact NMPC against ML-NMPCs with GB and DT models. The ODE solver did not converge for the polynomial models because the predicted C_A and C_R values grew too quickly, resulting in overflow. This could be explained by their poor test performances, suggesting that they gave poor estimates for dC_Adt and dC_Rdt. Solving the ML-NMPC problem exceeded 20 mins for the remaining candidate models, suggesting again that they evaluated too slowly to be feasible for fast-sampling applications.

Closed-loop control of ML-NMPC was completely ineffective in this case. Three reasons might explain this

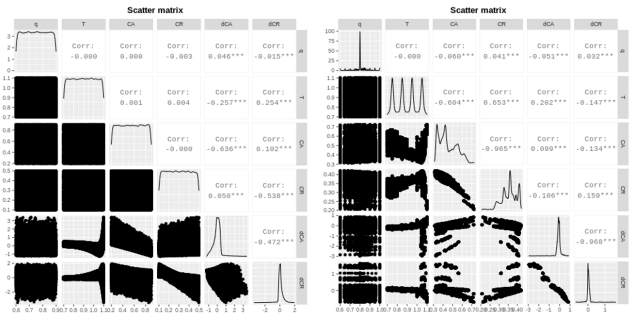

Fig. 8. Scatter matrices for the Uniform (left) and Oscillatory data sets (right).

observation. Firstly, as tree-based models were employed here, their piecewise-constant nature might not be suitable for solvers that depend on good gradient and Hessian estimates. Secondly, HO on this data set could have allowed models that generalize better to be learned. This was not performed in this piece and has been explored in a fuller version of this work. Thirdly, the Oscillatory method might not have yielded a "balanced" data set which provided an even representation across the system's operating region. The scatter matrices for the data from both data sets are shown in Fig. 8.

An "unbalanced" data set could reduce the model's generalization ability in regions which are poorly represented, negatively affecting the quality of $dC_A dt$ and $dC_R dt$ predictions there. When the goal of an MPC is to provide rapid and stable control over the entire operating range, a data set like the Uniform data set which evenly represents this range might therefore be desired.

5. CONCLUSION

In this study, a full SysID pipeline for deriving black-box nonlinear system models for NMPC of highly nonlinear MIMO systems was articulated, with this pipeline being able to accommodate different candidate models. Two different data generation methods for the ML task were studied, and an ML-NMPC formulation was presented which employs the ML system models as approximations to the system's continuous-time state representation.

Three control scenarios were identified for a MIMO single CSTR system exhibiting highly nonlinear dynamics. It was shown that an ML-NMPC with a polynomial system model, like an exact NMPC, succeeded in achieving rapid and stable control for all three scenarios, given that the data set is sufficiently uniform across the controller's operating range. Tree-based models were observed to perform poorly as system models for ML-NMPC, with this potentially due to their piecewise-constant nature. When more realistic data sets were used, ML-NMPC did not perform well in any of the three scenarios, with this bearing testament to the inherent difficulty of this control problem.

A follow-up work has employed an input signal consisting of simultaneous random step changes for all manipulated variables and has succeeded in identifying system models for ML-NMPC with this input which enabled effective control in both servo and regulator problems.