Contents lists available at ScienceDirect

Chemical Engineering Research and Design

journal homepage: www.elsevier.com/locate/cherd

Machine learning-based model predictive control of diffusion-reaction processes

Aarsh Dodhia[a], *Zhe Wu*[a], *Panagiotis D. Christofides*[a,b,*]

[a] *Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA*
[b] *Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095-1592, USA*

ARTICLE INFO

Article history:
Received 31 May 2021
Received in revised form 4 July 2021
Accepted 6 July 2021
Available online 12 July 2021

Keywords:
Parabolic PDEs
Galerkin's method
Neural network modeling
Model predictive control
Diffusion-reaction processes

ABSTRACT

In this work, we develop a machine-learning-based predictive control design for nonlinear parabolic partial differential equation (PDE) systems using process state measurement time-series data. First, the Karhunen-Loève expansion is used to derive dominant spatial empirical eigenfunctions of the nonlinear parabolic PDE system from the data. Then, these empirical eigenfunctions are used as basis functions within a Galerkin's model reduction framework to derive the temporal evolution of a small number of temporal modes capturing the dominant dynamics of the PDE system. Subsequently, feedforward neural networks (FNN) are used to approximate the reduced-order dominant dynamics of the parabolic PDE system from the data within a desired operating region. Lyapunov-based model predictive control (MPC) scheme using FNN models is developed to stabilize the nonlinear parabolic PDE system. Finally, a diffusion-reaction process example is used to demonstrate the effectiveness of the proposed machine-learning-based predictive control method.

© 2021 Institution of Chemical Engineers. Published by Elsevier B.V. All rights reserved.

1. Introduction

Diffusion-reaction processes have widespread industrial applications, for example, chemical vapor deposition, catalytic reactors, and crystal growth processes. These processes are governed by nonlinear parabolic partial differential equation (PDE) systems, and their dominant dynamic behavior can be characterized by a finite number of degrees of freedom (Christofides, 2001). The traditional approach to controlling systems governed by linear parabolic PDEs involves application of eigenfunction expansion techniques for deriving a reduced-order model of finite dimensional ordinary differential equations (ODEs) that describe the dominant dynamics of the PDE system (e.g., Balas, 1979; Curtain and Pritchard, 1978). Specifically, the solution of the original PDE system is initially expanded as the sum of an infinite series of the eigenfunctions of the spatial differential operator with time-varying coefficients. This expansion is subsequently used to derive an infinite set of ODEs for the coefficients of the expansion. Then, a finite-dimensional ODE model is derived by discarding an infinite set of equations. The finite-dimensional ODE model is subsequently used as the basis for the synthesis of finite-dimensional controllers. Significant work in the 90s was aimed at constructing lower-order nonlinear ODE models to generate nonlinear low-order controllers for nonlinear parabolic PDE systems (Christofides, 2001). These models were developed by combining Galrekin's method with the concept of approximate inertial manifolds and empirical eigenfunctions.

Most research in the area of model predictive control (MPC) has been focused on lumped-parameter process models. The challenge of deriving predictive controllers for distributed parameter systems, modeled by PDEs has attracted considerably less attention. Specifically, early work focused on using spatial discretization techniques such as finite differences to derive higher-order ODE models. Model predictive control techniques were then applied to these models, leading

* *Corresponding author at:* Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA 90095-1592, USA.
E-mail address: pdc@seas.ucla.edu (P.D. Christofides).
https://doi.org/10.1016/j.cherd.2021.07.005
0263-8762/© 2021 Institution of Chemical Engineers. Published by Elsevier B.V. All rights reserved.

to computationally expensive designs which are difficult to implement on-line (Dufour et al., 2003). Subsequently, MPC designs using low-order approximate models obtained via nonlinear Galerkin's method were developed including economic MPC designs (Dubljevic et al., 2006; Lao et al., 2014). With the rapid development of machine learning techniques in the past decade, and the ability of neural networks to approximate nonlinear functions according to the universal approximation theorem, they could be used to derive nonlinear prediction models for model predictive control with superior computational efficiency. Moreover, in the case where the process model is unknown, artificial neural networks can be trained off-line using process historical data to provide function approximators for model-based controllers. Motivated by the above considerations, in this work, we utilize neural network modeling techniques to model nonlinear parabolic PDE systems. Subsequently, we use MPC, which is an advanced control method that solves a numerical optimization problem to find the optimum control law. An important requirement for MPC is a process model which can predict the states of the system with desired accuracy. This process model can be derived by accounting for the physiochemical mechanisms that affect the process, or be derived using data-driven approaches (Hunt et al., 1992; Akesson and Toivonen, 2006; Huang and Kadali, 2008; Wu et al., 2019b) from process solution time-series data. Machine learning techniques such as neural networks have been successfully utilized in recent works in the design and implementation of model-based controllers (e.g., Han et al., 2013; Ali et al., 2015; Wang et al., 2016).

In this work, we first obtain empirical eigenfunctions using the Karhunen-Loève (K-L) expansion technique. Through Galerkin's method, ODE models are then derived to capture the dominant dynamics of the nonlinear parabolic PDE system using the obtained empirical eigenfunctions as basis functions. Subsequently, feedforward neural networks (FNN) are trained from PDE solution time-series data projected on the empirical eigenfunctions to describe the temporal evolution of the dominant modes. A Lyapunov-based MPC (LMPC) scheme using the FNN model is then developed following the formulations in Mhaskar et al. (2006) and Wu et al. (2019b) to stabilize the nonlinear parabolic PDE system. Closed-loop simulations of a diffusion-reaction process under the LMPC using low-order FNN models are compared with that using high-order discretization of the PDE model in the LMPC to demonstrate the efficacy of the proposed machine-learning-based predictive control method.

2. Preliminaries

2.1. Class of parabolic PDE systems

Consider a class of nonlinear parabolic partial differential equation systems of the form:

$$\frac{\partial \bar{x}}{\partial t} = A\frac{\partial \bar{x}}{\partial z} + B\frac{\partial^2 \bar{x}}{\partial z^2} + Wu(t) + f(\bar{x}(z,t)) \tag{1}$$

subject to the boundary conditions:

$$\begin{aligned} &\bar{x} = 0, \quad z = 0; \\ &\bar{x} = 0, \quad z = \pi; \end{aligned} \tag{2}$$

and the initial condition:

$$\bar{x}(z,0) = \bar{x}_0(z) \tag{3}$$

where $\bar{x}(z,t) = [\bar{x}_1(z,t) \cdots \bar{x}_{n_x}(z,t)]^T$ denotes the state vector of the system, $f(\bar{x}(z,t))$ denotes a nonlinear vector function, $z \in [0, \pi]$ is the spatial coordinate, $t \in [0, \infty)$ is the time, A, B, W are matrices and vectors of appropriate dimensions and $u(t)$ denotes the n_u-dimensional manipulated input vector and is subject to the following input constraints:

$$u_{\min} \le u(t) \le u_{\max} \tag{4}$$

where u_{min} and u_{max} are the lower and upper bound vectors of the manipulated input $u(t)$.

2.2. Karhunen-Loève expansion

The derivation of finite dimensional approximations of PDEs is imperative to design controllers for such nonlinear PDE systems. The principal hurdle in developing a general model reduction scheme for systems of the form of Eq. (1) is the presence of nonlinear terms. To overcome this problem, the Karhunen-Loève (K-L) expansion technique is applied to an ensemble of process solution data to derive a small set of empirical eigenfunctions which describe the dominant spatial patterns of the nonlinear PDE system (Sirovich, 1987; Holmes et al., 2012). These computed eigenfunctions will be used as basis functions within the Galerkin's model reduction framework (Park and Cho, 1996; Baker and Christofides, 2000). Specifically, we first generate a large ensemble set of the solutions of our system described by Eq. (1), which is denoted by $\{\bar{v}_k\}$ consisting of N states, $\bar{v}_k(z)$ sampled at uniform time intervals for computational ease. The ensemble average is defined as follows:

$$\langle \bar{v}_k \rangle = \frac{1}{K}\sum_{n=1}^{K} \bar{v}_n(z) \tag{5}$$

where K is the number of data samples. To obtain the most typical and characteristic functions $\phi(z)$, we analyze the fluctuations in sample states $\{\bar{v}_k\}$, given by the equation below:

$$v_k = \bar{v}_k - \langle \bar{v}_k \rangle \tag{6}$$

Subsequently, we define the covariance matrix B as follows:

$$B^{\kappa k} = \frac{1}{K}\int_0^{\pi} v_\kappa(z) v_k(z) dz \tag{7}$$

and obtain the eigenvalues by solving the following equation:

$$Bc = \lambda c \tag{8}$$

where $c = [c_1, \ldots, c_K]$ denotes the eigenvectors matrix, which can be used to construct $\phi(z)$ with the following equation:

$$\phi(z) = \sum_k c_k v_k(z) \tag{9}$$

When the eigenfunctions are ordered such that the eigenvalues satisfy $\lambda_1 > \lambda_2 > \cdots > \lambda_{l+1}$, then the following equation holds for a given decomposition of a state:

$$v_k(z) = \sum_{l=1}^{L} \gamma_l \phi_l(z) \tag{10}$$

The projection on the subspace spanned by the empirical eigenfunctions will on average contain the most energy possible compared to all other linear decompositions for any L. The calculated eigenvalues, once normalized, represent the percentage of energy, or equivalently, time that the solution of the PDE system spends along the spatial structure of the empirical eigenfunction.

2.3. *Galerkin's method*

In this section, we use Galerkin's methods to derive low-order dynamic systems of nonlinear ordinary differential equations that represent the dominant dynamics and thereby solutions of the nonlinear PDE system of Eq. (1). To simplify the presentation, we assume that we have available an orthogonal and complete set of analytical eigenfunctions which span the entire domain of the nonlinear process as basis functions $\phi_k(z)$ which satisfy the boundary conditions listed in Eq. (2). In practice, $\phi_k(z)$ will be the set of empirical eigenfunctions computed through K-L expansion.

We formulate the PDE system as an infinite dimensional system in the Hilbert space $\mathcal{H}([0, \pi]; \mathrm{IR}^{n_x})$, with $\mathcal{H}$ being the space of measurable vector functions defined on $[0, \pi]$, with inner product and norm:

$$\begin{aligned} (\omega_1, \omega_2) &= \int_0^{\pi} (\omega_1(z), \omega_2(z))_{\mathrm{IR}^{n_x}} dz, \\ \|\omega_1\|_2 &= (\omega_1, \omega_1)^{\frac{1}{2}} \end{aligned} \tag{11}$$

where ω_1, ω_2 are two elements of $\mathcal{H}([0, \pi]; \mathrm{IR}^{n_x})$ and the notation $(\cdot, \cdot)_{\mathrm{IR}^{n_x}}$ denotes the standard inner product in IR^{n_x}. The state function $x(t)$ on the state-space $\mathcal{H}$ is defined as

$$x(t) = \bar{x}(z, t), \quad t > 0, \quad 0 \le z \le \pi, \tag{12}$$

and the operator $\mathcal{A}$ is defined as

$$\mathcal{A}x = A\frac{d\bar{x}}{dz} + B\frac{d^2\bar{x}}{dz^2}, \quad 0 \le z \le \pi. \tag{13}$$

Then, the system of Eq. (1) takes the following infinite-dimensional quasi-linear form:

$$\dot{x}(t) = \mathcal{A}x(t) + \mathcal{B}u(t) + \mathcal{F}(x(t)), \quad x(0) = x_0 \tag{14}$$

where $x_0 = \bar{x}_0(z)$, $\mathcal{B}u(t) = Wu(t)$, and $\mathcal{F}(x(t))$ is a nonlinear vector function in the Hilbert space. For the operator $\mathcal{A}$, the eigenvalue problem takes the form

$$\mathcal{A}\phi_k = \lambda_k \phi_k, \quad k = 1, \ldots, \infty \tag{15}$$

subject to

$$\begin{aligned} &\phi_k(0) = 0, \\ &\phi_k(\pi) = 0 \end{aligned} \tag{16}$$

where ϕ_k is an eigenfunction corresponding to the kth eigenvalue of the operator $\mathcal{A}$.

Under the assumption that most diffusion-reaction processes have large separations of slow and fast modes of the spatial operator, the eigenspectrum of operator $\mathcal{A}$ can be partitioned into a finite part consisting of m slow eigenvalues and a stable infinite complement containing the remaining fast eigenvalues (Lao et al., 2014). We apply standard Galerkin's method to the infinite-dimensional system of Eq. (14) to derive a finite-dimensional system. Let $\mathcal{H}_s$ and $\mathcal{H}_f$ be modal subspaces of $\mathcal{A}$ defined as $\mathcal{H}_s = \mathrm{span}\{\phi_1, \phi_2, \ldots, \phi_m\}$ and $\mathcal{H}_f = \mathrm{span}\{\phi_{m+1}, \phi_{m+2}, \ldots\}$. Let P_s and P_f denote the orthogonal projection operators that project the state x onto the subspaces $\mathcal{H}_s$ and $\mathcal{H}_f$ of $\mathcal{A}$, respectively (i.e., $x_s = P_s x \in \mathcal{H}_s$ and $x_f = P_f x \in \mathcal{H}_f$). The state x of the system of Eq. (14) can be decomposed as follows:

$$x = x_s + x_f = P_s x + P_f x \tag{17}$$

Applying P_s and P_f to the system of Eq. (14) and using the above decomposition for x, the system of Eq. (14) can be re-written in the following equivalent form:

$$\begin{aligned} \dot{x}_s(t) &= \mathcal{A}_s x_s(t) + \mathcal{F}_s(x_s(t), x_f(t)) + \mathcal{B}_s u(t), \quad x_s(0) = P_s x(0) = P_s x_0 \\ \dot{x}_f(t) &= \mathcal{A}_f x_f(t) + \mathcal{F}_f(x_s(t), x_f(t)) + \mathcal{B}_f u(t), \quad x_f(0) = P_f x(0) = P_f x_0 \end{aligned} \tag{18}$$

where $\mathcal{A}_s = P_s\mathcal{A}$, $\mathcal{B}_s = P_s\mathcal{B}$, $\mathcal{A}_f = P_f\mathcal{A}$, $\mathcal{B}_f = P_f\mathcal{B}$, $\mathcal{F}_f = P_f\mathcal{F}$, $\mathcal{F}_s = P_s\mathcal{F}$. In the above system, $\mathcal{A}_s = \mathrm{diag}\{\lambda_j\}$, $j = 1, \ldots, m$ is a diagonal matrix of dimension $m \times m$ and may contain unstable eigenvalues (i.e., $Re(\lambda_j) > 0$). The operator $\mathcal{A}_f$ is an unbounded exponentially stable differential operator. Neglecting the fast modes, the resulting ODE system is

$$\dot{x}_s(t) = \mathcal{A}_s x_s(t) + \mathcal{F}_s(x_s(t), 0) + \mathcal{B}_s u(t), \quad x_s(0) = P_s x_0 \tag{19}$$

which is a finite-dimensional system that describes the slow (dominant) dynamics of the PDE system of Eq. (1).

3. Machine learning modeling of temporal evolution

In this section we discuss the order reduction of $\bar{x}(z, t)$ using the dominant empirical functions $\phi_k(z)$ as basis functions in a Galerkin's model reduction framework. Applying the model reduction on available process data gathered from open-loop simulations of the PDE system of Eq. (1), we intend to train and develop machine-learning-based feedforward neural network models which capture the dynamics of the nonlinear PDE system with a sufficiently high accuracy.

To this end, we first reduce the PDE system to an ODE model. Using Galerkin's method, we obtain a higher-order ODE system describing the temporal evolution of the amplitudes of the corresponding m slow and dominant eigenfunctions. The state $\bar{x}(z, t)$ can be written as:

$$\bar{x}(z, t) = \sum_{i=1}^{m} c_i(t)\phi_i(z) \tag{20}$$

Substituting the above equation in Eq. (14), we obtain:

$$M\dot{c}(t) = MA_s c(t) + F_s(c(t)) + B_s(u(t)) \tag{21}$$

where $c(t) = [c_1(t), c_2(t), \ldots, c_i(t), \ldots, c_m(t)]^T$ with elements $c_i(t) \in$ IR associated with the amplitudes of the first m eigenfunctions. The matrix $A_s = \text{diag}\{\lambda_i\}$ is a $m \times m$ matrix (i.e., $i = 1, \ldots, m$), the matrices B_s and the nonlinear vector fields F_s can be constructed through the appropriate inner product, e.g.,

$$B_{s,i} = \begin{bmatrix} (\phi_1(z),\ u(t)) \\ (\phi_2(z),\ u(t)) \\ \vdots \\ (\phi_m(z),\ u(t)) \end{bmatrix} \tag{22}$$

where M is a $m \times m$ matrix given by:

$$M = \begin{bmatrix} (\phi_1(z),\ \phi_1(z)) & (\phi_1(z),\ \phi_2(z)) & \ldots & (\phi_1(z),\ \phi_m(z)) \\ (\phi_2(z),\ \phi_1(z)) & (\phi_2(z),\ \phi_2(z)) & \ldots & (\phi_2(z),\ \phi_m(z)) \\ & \vdots & & \\ (\phi_m(z),\ \phi_1(z)) & (\phi_m(z),\ \phi_2(z)) & \ldots & (\phi_m(z),\ \phi_m(z)) \end{bmatrix}. \tag{23}$$

The initial conditions of the ODEs in Eq. (14) are

$$Mc(0) = \begin{bmatrix} (\phi_1(z),\ \bar{x}(z,0)) \\ (\phi_2(z),\ \bar{x}(z,0)) \\ \vdots \\ (\phi_m(z),\ \bar{x}(z,0)) \end{bmatrix}, \tag{24}$$

After obtaining the eigenfunctions from K-L expansion which capture the dominant dynamics of the nonlinear system, we run extensive open-loop simulations within the region of interest by manipulating actuator inputs $u(t)$ and initial states of the system $\bar{x}(0)$. Inputs $u(t)$ are implemented in a sample-and-hold fashion, as a piecewise constant function over a sampling period Δ and the system of Eq. (1) is integrated using the finite differences method with a sufficiently small integration time step $h < \Delta$. Using the eigenfunctions, we decompose the higher-order states of the system attained from the open-loop simulations to obtain temporal evolution of the amplitudes of the eigenfunctions for the PDE system $c(t) = [c_1(t), c_2(t), \ldots, c_i(t), \ldots, c_m(t)]^T$. Using the data set generated by these simulations, we train a feedforward neural network (FNN) to capture the process dynamics within the operating region. The FNN model is developed with 2 hidden layers, with H1 and H2 neurons in each layer. Fig. 1 denotes the skeleton of the FNN model, which takes the current state of the reduced system $c(t_k)$ and the manipulated input profile $u(t)$ as inputs and returns the predicted state of the system at the next time step, $c(t_k + h)$ as its output, where h is the integration time step. The data sets generated by the set of ensembles are then partitioned using the split functions in Python to training, validation and testing sets. This concludes the formulation of the FNN model which captures the nonlinear dynamics of the system.

Remark 1. The matrix M would be an identity matrix, $\mathbb{I}_m$ in case of analytically derived eigenfunctions. But in most data-driven approaches when the process model is unknown or K-L expansion is used over an ensemble of states to compute empirical eigenfunctions, the matrix M is close to I_m and hence should be accounted for to reduce errors.

Remark 2. FNN models are developed in Python using Keras (an open source network of libraries) (Chollet et al., 2015). The optimization problem of minimizing fitting error is solved by using the Adam estimation method, with the loss function calculated as the mean squared error between the predicted and true states.

Remark 3. We choose FNN structure to build the data-driven model in this work since the neural network is developed to predict one integration time step only. In case that the model is designed to predict multiple time steps, recurrent neural networks (RNN) and nonlinear autoregressive network with exogenous inputs (NARX) models can be used in time-series modeling.

4. Lyupanov-based model predictive control

In this section, we develop a Lyupanov-based model predictive controller (MPC) using the FNN model derived for the nonlinear process with a sufficiently small modeling error. We assume there exists a stabilizing feedback controller $u = \hat{\Phi}_{nn}(c)$ that renders the origin of the system of Eq. (14) exponentially stable in a neighborhood around the origin. The stabilizability assumption implies there exists a control Lyapunov function $\hat{V}(c)$ such that the following inequalities hold for all c in a neighborhood D around the origin:

$$\hat{\alpha}_1|c|^2 \leq \hat{V}(c) \leq \hat{\alpha}_2|c|^2, \tag{25a}$$

$$\frac{\partial \hat{V}(c)}{\partial c} F_{nn}(c, \hat{\Phi}_{nn}(c)) \leq -\hat{\alpha}_3|c|^2, \tag{25b}$$

$$\left| \frac{\partial \hat{V}(c)}{\partial c} \right| \leq \hat{\alpha}_4|c| \tag{25c}$$

where $\hat{\alpha}_1$, $\hat{\alpha}_2$, $\hat{\alpha}_3$, $\hat{\alpha}_4$ are positive constants, and $F_{nn}(c, u)$ represents the FNN model of Eq. (21) (i.e., $MF_{nn}(c, u) := MA_s c(t) + F_s(c(t)) + B_s(u(t))$). We first search the entire state-space to characterize a set of states $\hat{\phi}_u$ where Eq. (25) holds under $u = \hat{\Phi}_{nn}(c) \in U$, and U is the control action constraint set $U := \{u_{min} \leq u_i \leq u_{max},\ i = 1, 2, \ldots, m\} \subset \mathbf{R}^m$. Then, we characterize the closed-loop stability region $\Omega_{\hat{\rho}}$ for the nonlinear system of Eq. (14) as a level set of Lyapunov function within the set $\hat{\phi}_u$, $\Omega_{\hat{\rho}} := \{c \in \hat{\phi}_u \mid \hat{V}(c) \leq \hat{\rho}\}$, $\hat{\rho} > 0$.

Based on the stabilizing control law $u = \hat{\Phi}_{nn}(c) \in U$, the Lyapunov-based MPC design using neural network model is given by the following optimization problem (Wu et al., 2019b):

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} L(\tilde{c}(t), u(t))dt \tag{26a}$$

$$\text{s.t.} \quad \dot{\tilde{c}}(t) = F_{nn}(\tilde{c}(t), u(t)) \tag{26b}$$

$$u(t) \in U, \quad \forall\ t \in [t_k, t_{k+N}) \tag{26c}$$

$$\tilde{c}(t_k) = c(t_k) \tag{26d}$$

$$\dot{\hat{V}}(c(t_k), u) \leq \dot{\hat{V}}(c(t_k), \hat{\Phi}_{nn}(c(t_k))), \quad \text{if} \quad c(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}} \tag{26e}$$

$$\hat{V}(\tilde{c}(t)) \leq \rho_{nn}, \quad \forall\ t \in [t_k, t_{k+N}), \quad \text{if} \quad c(t_k) \in \Omega_{\rho_{nn}} \tag{26f}$$

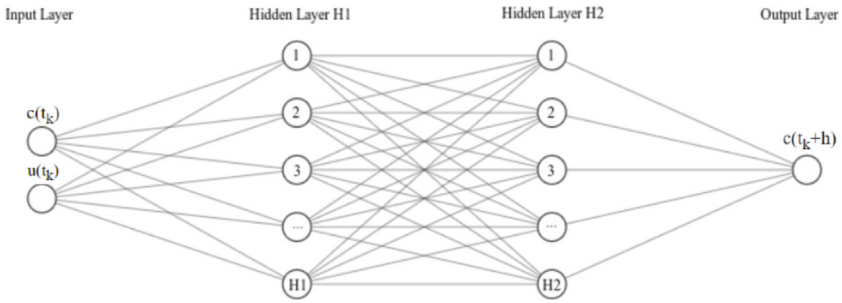

Fig. 1 – The schematic of feedforward neural network, where *H1* and *H2* are the hidden layers, $c(t_k)$ is the state vector of the temporal modes, $c(t_k + h)$ is the predicted state vector of the temporal modes and $u(t_k)$ is the manipulated input vector.

where $\tilde{c}$ is the state trajectory predicted by the neural network, $S(\Delta)$ is the set of piecewise constant functions with period Δ, and N is the number of sampling periods in the prediction horizon. The control Lyapunov function $V(c) = c^T P c$ is designed with a positive definite P matrix. $\dot{\hat{V}}(c, u)$ is used to represent $\frac{\partial \hat{V}(c)}{\partial c}(F_{nn}(c, u))$. The optimal input trajectory computed by the LMPC is denoted by $u^*(t)$, which is calculated over the entire prediction horizon $t \in [t_k, t_{k+N})$. The control action computed for the next sampling period of the prediction horizon $u^*(t_k)$ is sent by the LMPC to be applied over one sampling period and the LMPC is re-evaluated with new state measurements at the next sampling time.

4.1. *Computational methodology*

This section discusses the computational procedures for the development of FNN models and optimization of LMPC. A diffusion-reaction process example will follow the section, where the detailed algorithm of methodology discussed in this section is implemented. The detailed steps of the LMPC system flowchart are shown in Fig. 2 and are presented as follows:

4.1.1. *Development of FNN models*

1. Since the analytical process model accounting for the various physiochemical mechanisms that affect the process may be unknown, we employ a data-driven approach to obtain an ensemble data set of the system states from extensive open-loop simulations of the PDE system.
2. Perform K-L expansion on this ensemble data set to obtain a set of empirical eigenfunctions. The calculated eigenvalues, once normalized, represent the percentage of energy, or equivalently, time that the solution of the PDE system spends along the spatial structure of the empirical eigenfunction.
3. Generate a set of the first m eigenfunctions in descending order of their eigenvalues, and compute the fraction of energy stored in this set over the total ensemble of data. If the fraction of energy stored in the set is large enough (>99%) then proceed to step 5, else continue to step 4.
4. Since the energy stored in the set of empirical eigenfunctions is not large enough, we add one more eigenfunction to this set and repeat step 3.
5. The set now accounts for >99% energy in the ensemble, and will be used as the set of dominant empirical eigenfunctions. Convert the ensemble $< \bar{x}(z, t) >$ of open-loop simulations to $<c(t)>$ using Galerkin's method for model reduction using the set of dominant empirical eigenfunctions as basis functions.
6. Train FNN models in Python on the data set $<c(t)>$ which represents dominant temporal patterns of the nonlinear PDE system. The FNN model takes the current state of the reduced system $c(t)$ and the manipulated input profile $u(t)$ as inputs and returns the predicted state of the system at the next time step, $c(t + h)$ as its output.
7. Calculate the modeling error of the FNN model. If it is less than the modeling error threshold ψ, proceed to step 9, else continue to step 8.
8. Since the modeling error is still large, retrain FNN model by modifying network parameters and repeat step 7.
9. After obtaining the FNN process model, proceed to model predictive control. This step ends the development of FNN models.

Remark 4. In this work, we chose 99% as the threshold to determine whether the set of dominant empirical eigenfunctions can well represent the original high-dimensional system, and can be used as the basis functions within the Galerkin's model reduction framework. With a smaller percentage, fewer eigenfunctions may be chosen for basis functions, but the representation of the original system may not be as good as that under 99%.

4.1.2. *LMPC computation procedure*

1. Obtain a measurement of the current state $\bar{x}(z, t_k)$ at $t = t_k$.
2. Convert $\bar{x}(z, t_k)$ to $<c(t_k)>$ using Galerkin's method for model reduction with the set of dominant empirical eigenfunctions obtained from K-L expansion as basis functions.
3. Solve the LMPC optimization problem of Eq. (26) and obtain the optimal input $u^*(t)$ which is calculated over the entire prediction horizon $t \in [t_k, t_{k+N})$.
4. Control action computed for the first sampling period of the prediction horizon $u^*(t_k)$ is sent to the nonlinear PDE system.
5. At the next sampling time, LMPC receives a new state measurement, and is resolved following steps 1–4. The LMPC is executed iteratively until the end of the operation period.

5. Application to a diffusion-reaction process

In this section, we use a diffusion-reaction chemical process example to illustrate the application of machine learning modeling and predictive control of parabolic PDE systems. We first present the model of the diffusion-reaction process and compute the empirical eigenfunctions using K-L expansion. Then, we apply Galerkin's method to derive a reduced-order model, and generate an FNN model that accounts for temporal variations based on data generated from open-loop simula-

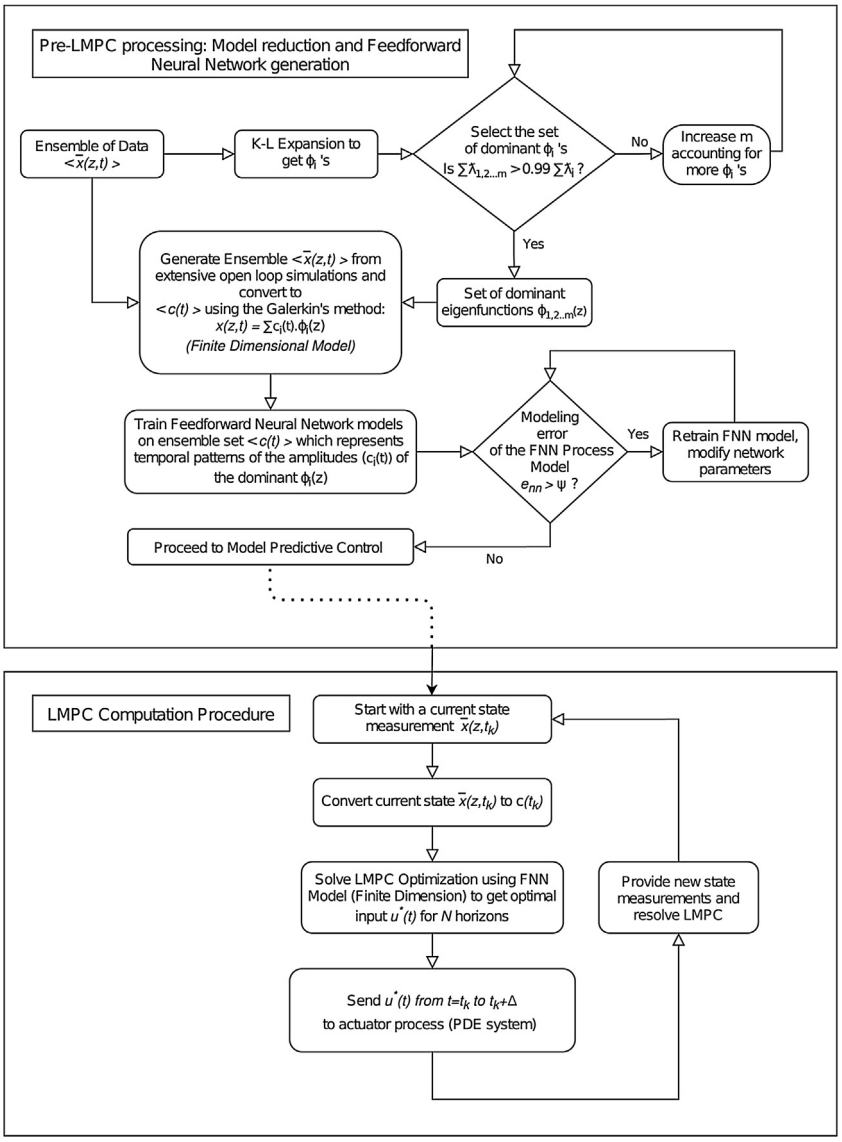

Fig. 2 – A flowchart for the Lyapunov-based model predictive controller design and implentation.

tions of the PDE system. Finally, we carry out closed-loop simulations of the PDE system under the LMPC using the reduced-order FNN model.

5.1. *Process description*

Consider a diffusion-reaction process given by the following nonlinear parabolic PDE equation:

$$\frac{\partial \bar{x}}{\partial t} = \frac{\partial^2 \bar{x}}{\partial z^2} + \beta_T\left(e^{\frac{-\gamma}{1+\bar{x}}} - e^{-\gamma}\right) + \beta_U(b(z,t)u(t) - \bar{x}) \tag{27}$$

where $\bar{x}$ denotes the state vector of the system, γ, β_T and β_U are dimensionless process parameters, $u(t) = [u_1(t)\ u_2(t)]^T$ is the input vector, and $b(z,t) = [b_1(t)\ b_2(t)]$ is a function which controls how the inputs are distributed in space and time. The system is subject to Dirichlet boundary conditions:

$$\begin{aligned} z = 0 \quad &: \quad \bar{x}(0,t) = 0 \\ z = L \quad &: \quad \bar{x}(L,t) = 0 \\ t = 0 \quad &: \quad \bar{x}(z,0) = \bar{x_0}(z) \end{aligned} \tag{28}$$

where $\gamma = 4$, $\beta_T = 50$, $\beta_U = 2$ and $L = \pi$. The process has two equidistant actuators, $b_1(t) = \frac{\delta(z-\pi/3)}{\beta_U}$ and $b_2(t) = \frac{\delta(z-2\pi/3)}{\beta_U}$, where δ is the standard Dirac delta function.

5.2. *Computation of empirical eigenfunctions and Galerkin's method*

To compute the set of empirical eigenfunctions, we construct an ensemble of solutions by solving a higher-order discretization of Eq. (27). For an initial condition $\bar{x}(z,0) = 5$, and a series of time-varying manipulated input profiles, we generate 25,000 state profiles, from which we extract 5,000 "snapshots" at equal time intervals. Then, we use as an ensemble to compute empirical eigenfunctions using the K-L expansion technique. We compute three eigenfunctions that describe the dominant spatial development patterns in the ensemble of solutions, accounting for more than 99% energy included in the ensemble. Fig. 3 shows the three dominant empirical eigenfunctions arranged in descending order of eigenvalues (i.e., $\lambda_1 > \lambda_2 > \lambda_3$).

Using the three empirical eigenfunctions as basis functions of the PDE, and employing Galerkin's method, we construct a third-order reduced model for the PDE system. Fig. 4 compares the evolution of the state of the system from the higher-order discretized (numerical) solution and from the

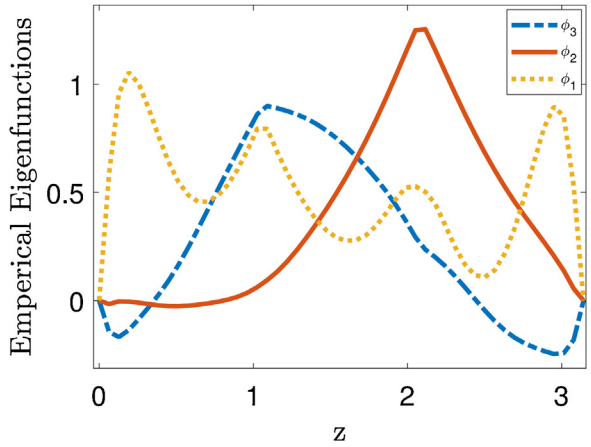

Fig. 3 – Three dominant empirical eigenfunctions extracted from K-L expansion for an ensemble of data which are used as basis functions in Galerkin's method.

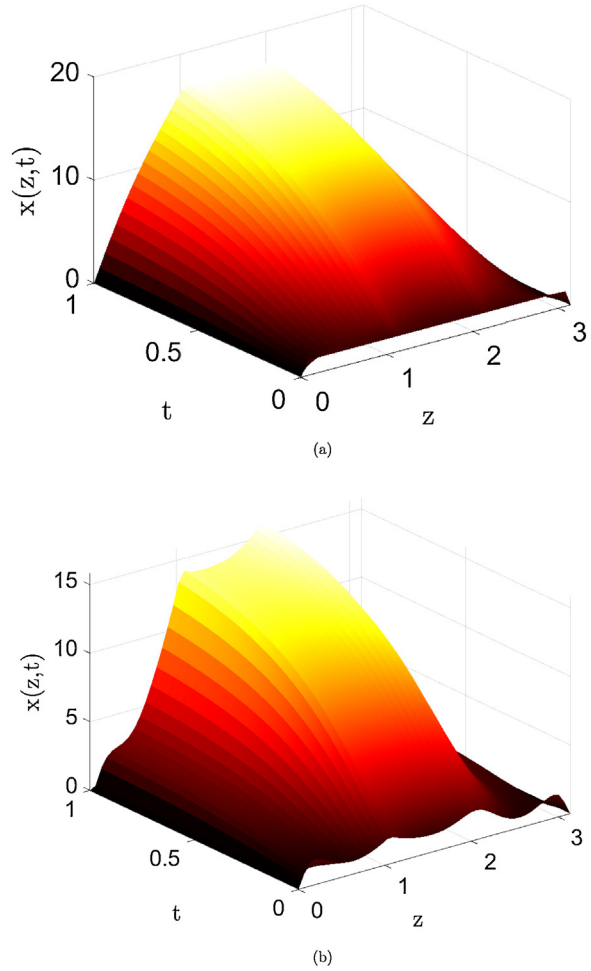

Fig. 4 – (a) Higher-order discretized solution of the parabolic PDE, and (b) third-order Galerkin's solution (reduced-order) generated using dominant empirical eigenfunctions as basis functions.

third-order reduced model solution for the same initial conditions and time-varying manipulated input profiles. Using the reduced-order model framework, we derive the temporal evolution of the temporal modes (i.e., c_i's in Eq. (20)), capturing the dominant dynamics of the PDE system. This will be used as our training data set for developing neural network models.

Remark 5. To build the dataset, we first discretize the operating region and choose initial conditions that are evenly spread in this region. Then, each time we choose an initial condition and an input value within the input range to carry out open-loop simulation for one integration time step. Since the simulation only runs for one integration time step, for any initial condition within the operating region, the state trajectories are guaranteed to be bounded in the operating region.

Remark 6. In this example, we use extensive open-loop simulation data to generate the dataset since the system is operated around a stable steady-state. In case that the system is operated around an unstable steady-state, we may consider using closed-loop data under a PI controller that ensures system stability to build data-driven models. Interested readers may refer to Hassanpour et al. (2020), Shardt and Huang (2011), and Shardt et al. (2015) for model development using closed-loop data. Additionally, in the presence of mismatch between the first-principles model (for simulation) and the actual process model, we can first develop a machine learning model offline using data from the first-principles model. After the model is applied to the real process, we can employ online learning with real-time data to improve machine learning models. Methods such as event-triggered and error-triggered mechanisms can be utilized to activate online learning to ensure the boundedness of modeling error for all time (Wu et al., 2019a). If both the first-principles model and real data are available, in addition to the online learning approach, we can use both data to develop a machine learning model at the offline training stage. In Wu et al. (2021), we have demonstrated that following the co-teaching training algorithm, the machine learning model is able to capture the underlying process dynamics using both the noisy real data and noise-free first-principles model data.

5.3. *Neural network modeling of temporal evolution*

In this section we discuss how to generate an FNN model to approximate the reduced-order dominant dynamics for the nonlinear parabolic PDE system of Eq. (27). Using the three eigenfunctions obtained for our system from K-L expansion, we rewrite the high-dimensional state $\bar{x}(t)$ as shown in Eq. (20). Various simulations are carried out for different initial conditions $\bar{x}_0(t)$, and manipulated input profiles $u(t) = [u_1(t), u_2(t)]^T$. The ensemble of solutions for a given initial condition and input scheme are generated over N sampling periods of size Δ for $c(t)$. The various ensemble sets are congregated to create the whole data set for the FNN model in Python.

The FNN model is developed with 2 hidden layers, with 64 neurons each layer, connected to a fully-connected layer. Fig. 1 denotes the skeleton of the FNN model, which takes the current state of the reduced system $c(t_k)$ and the manipulated input profile $u(t)$ as inputs and returns the predicted state of the system at the next timestep, $c(t_k + h)$ as its output. The data set generated by the set of ensembles is then partitioned using the split functions in Python to training, validation and testing sets. The FNN model is trained using Keras, which is a state-of-the-art, open source machine learning library in Python. The Adam optimization algorithm associated with the mean squared error loss function is used to optimize the weights in the model. To determine if the FNN model is sufficiently accurate, we calculate the error between the temporal modes

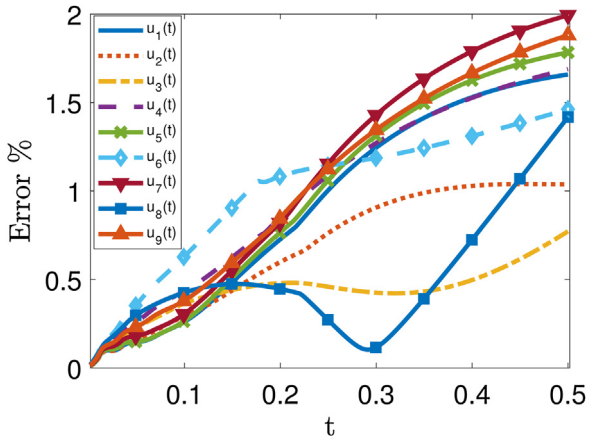

Fig. 5 – Prediction error of FNN model from reduced-order model computed using Eq. (29) for various input profiles and initial conditions in the operating region.

predicted from the FNN model and Galerkin's method as follows:

$$e(t) = \frac{||c_{NN}(t) - c(t)||_2}{||c(t)||_2} \tag{29}$$

where $c_{NN}(t)$ is the state predicted by the FNN model and $c(t)$ is the solution obtained from Galerkin's method. Fig. 5 shows the prediction error of the FNN model for different initial conditions and manipulated input profiles. The increase of error over time is due to compounding of errors in prediction from the first instant; however, it is demonstrated that the modeling error remains less than 2% and saturates after the system reaches the steady-state values, which indicates that the FNN model can capture the nonlinear dynamics well. Fig. 6 compares the state profiles from the prediction of the FNN model and the reduced-order Galerkin's method model (i.e., third-order Galerkin's method approximation of the PDE with the first three empirical eigenfunctions as basis functions) for the same initial conditions and time-varying input profiles.

Remark 7. Note that the inputs to the PDE system are implemented in a sample-and-hold fashion. Inputs are fed as piece-wise constant functions of time $u(t) = u(t_k)$, $\forall\, t \in [t_k, t_k + \Delta]$ where Δ is the sampling period. The system is integrated with a sufficiently small integration time step h (which is much smaller than the sampling time Δ) for the calculation of the high-order discretization solutions.

Remark 8. The optimal neural network hyperparameters are chosen from a grid search. Specifically, with MSE as loss function and Adam as training algorithm, we start with a simple neural network with one hidden layer and a few neurons, and gradually increase the number of neurons and layers until no further improvement is noticed. Additionally, during this process, we introduce early stopping to avoid over-fitting.

5.4. *Lyapunov-based model predictive control*

In the optimization problem of Eq. (26), the control Lyapunov function $V(c) = c^T P c$ is designed with the positive definite identity P matrix $\mathbb{I}_3$. The LPMC cost function is defined as $L(c(t), u(t)) = 100||c||_1 + 0.001||u||_2^2$, such that origin is the minimum of $L(c, u)$; quadratic costs were also considered but the one ultimately employed was found to give the best closed-

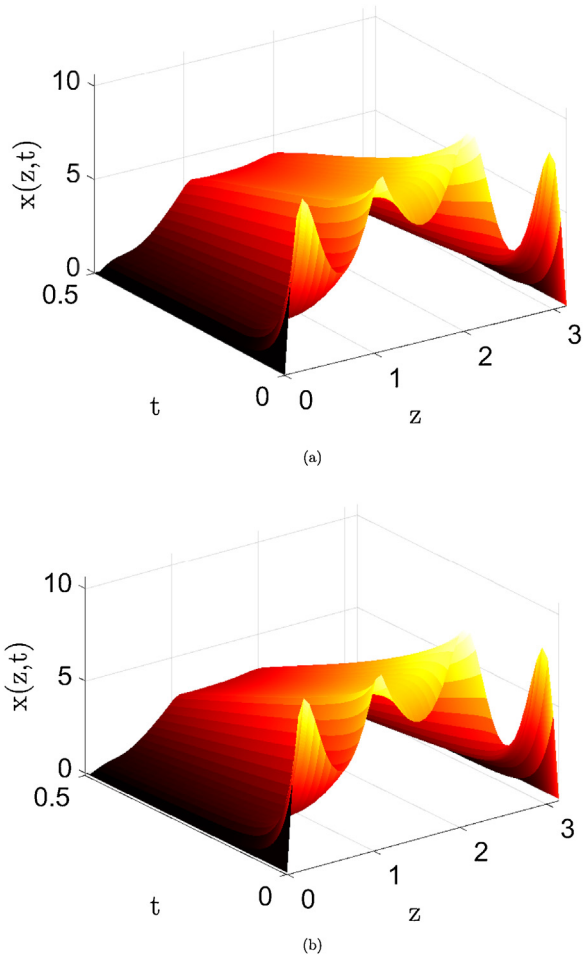

Fig. 6 – (a) Feedforward neural network model prediction, and (b) third-order model obtained via Galerkin's method for $x_0 = 7.5$ and actuator inputs $u(t) = -100$ located at $z = \pi/3$ and $z = 2\pi/3$.

loop performance. The constraint of Eq. (26b) is the FNN model used to predict the states of the closed-loop system of Eq. (27). Eq. (26c) defines the input constraints applied over the entire prediction horizon. Eq. (26d) defines the initial condition $\tilde{c}(t_k)$ of Eq. (26b), which is the state measurement at $t = t_k$. The constraint of Eq. (26e) forces the closed-loop state to move toward the origin if $c(t_k) \in \Omega_{\hat{\rho}} \backslash \Omega_{\rho_{nn}}$, where $\Omega_{\rho_{nn}}$ is terminal set. However, if $c(t_k)$ enters $\Omega_{\rho_{nn}}$, the states predicted by the FNN model of Eq. (26b) will be maintained in $\Omega_{\rho_{nn}}$ for the entire prediction horizon. The process is initially operated at $x_0(z) = 7.5$, which corresponds to $c(t = 0) = [1.84, 4.18, 9.25]^T$ using Eq. (24). The two manipulated inputs are signals given by the left and right actuators with bounds $|u_1(t)| < 300$ and $|u_2(t)| < 300$, respectively. The states of the closed-loop system are $c(t) = [c_1(t), c_2(t), c_3(t)]^T$ and the manipulated inputs are $u(t) = [u_1(t), u_2(t)]^T$, with the origin being the desired equilibrium point of the system. The control objective is to manipulate $u(t)$ such that the process can be stabilized at the origin under LPMC using the FNN models. The finite difference method with an explicit Euler implementation is used to numerically simulate the dynamic model of Eq. (27) with an integration time step $h = 10^{-3}$. The nonlinear optimization of the LPMC of Eq. (26) is solved using the Python module of the IPOPT (Wächter and Biegler, 2006) software package named PyIpopt with a sampling period $\Delta = 0.1$ and number of sampling periods in the prediction horizon $N = 4$.

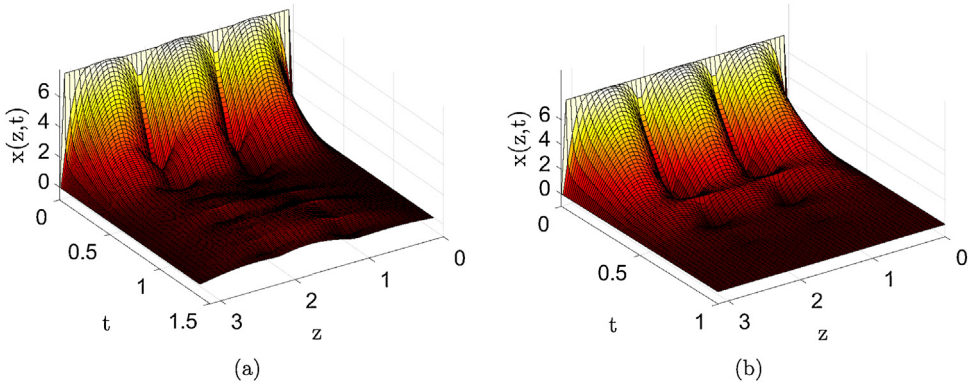

Fig. 7 – Closed-loop state profiles under the LMPC of Eq. (26) using (a) FNN model, and (b) higher-order discretization of the parabolic PDE, for the initial condition $x_0(z) = 7.5$.

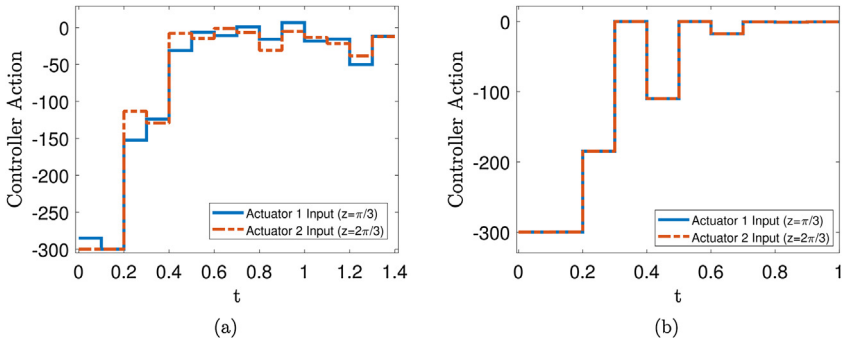

Fig. 8 – Control action profiles $u^*(t)$ under LMPC using (a) FNN model, and (b) higher-order discretization of the parabolic PDE, for the initial condition $x_0(z) = 7.5$.

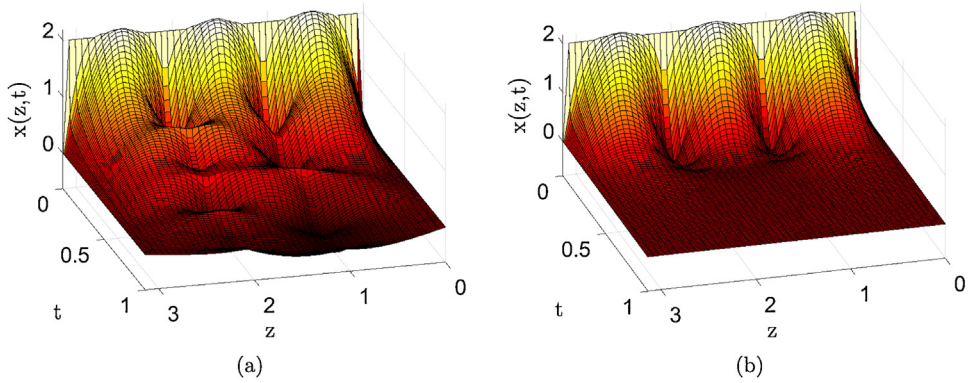

Fig. 9 – Closed-loop state profiles for the LMPC using (a) FNN model, and (b) higher-order discretization of the parabolic PDE, for the initial condition $x_0(z) = 2$.

Fig. 7(a) shows the evolution of the closed-loop state under the optimal manipulated input trajectory $u^*(t)$ computed by solving the LMPC using the FNN model. For $t = 0.6$, the system enters $\Omega_{\rho_{nn}}$ and is maintained there for the remaining period. For $t > 0.6$, inputs oscillate around 0 in an attempt to maintain the system state in the region. Fig. 8(a) displays the optimal control action obtained by solving the LMPC optimization problem of Eq. (26), where it can be seen that the inputs meet the constraints for all times. Fig. 7(b) shows the evolution of the closed-loop state under the LPMC using the higher-order discretized solution of the original PDE model, which serves as a benchmark to compare the FNN process model MPC results. Fig. 8(b) displays the optimal control action under the LMPC using the higher-order discretized model.

Comparing the results of Figs. 7 and 8, we observe that the closed-loop performance using neural network model is close to that using higher-order discretized solution of the original PDE model. In Fig. 8, it is observed that the neural network model remains aggressive in its efforts to bring the state to the origin, leading to some correction toward the end of the operation period, whereas the first principles solution maintains actuator input values below 0. Additionally, Figs. 9 and 10show the closed-loop state and control action profiles for $x_0(z) = 2$. Results obtained for this case are consistent with those of the previous case, and the closed-loop state profiles for both models are ultimately maintained within a small neighborhood around the origin. The closed-loop simulation study demonstrates that the FNN model provides a desired approximation

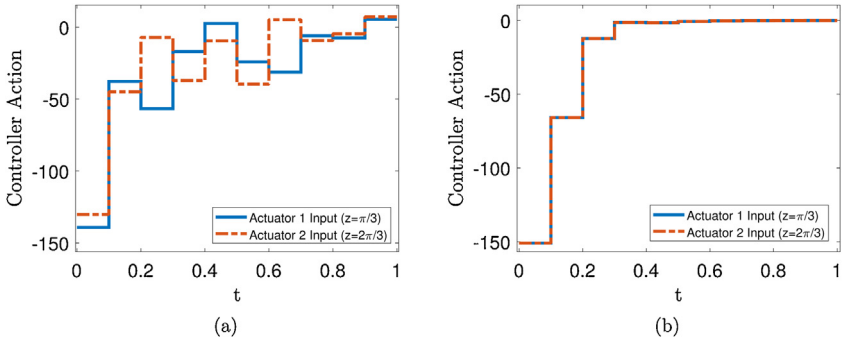

Fig. 10 – Control action profiles $u^*(t)$ under LMPC using (a) FNN model, and (b) higher-order discretization of the parabolic PDE, for the initial condition $x_0(z) = 2$.

of the nonlinear PDE system and can be used to stabilize the system in the framework of LMPC.

Remark 9. To use the FNN model that was developed for predicting one integration time step in MPC with multi-step prediction horizon, we execute FNN predictions successively, and each time use the previous predicted state as the initial condition for the current prediction. In this way, we are able to predict future states for the entire MPC prediction horizon, which will be used in the MPC cost function and constraints.

6. Conclusion

In this work, we developed machine-learning-based MPC for diffusion-reaction processes. We first derived dominant spatial empirical eigenfunctions of the nonlinear PDE system by applying K-L expansion to process solution time-series data. These eigenfunctions were then used as basis functions within a Galerkin's model reduction framework to derive the temporal evolution of a small number of modes capturing the dominant dynamics of the PDE system. Subsequently, FNN models were trained using extensive open-loop simulations in an operating region to approximate the reduced-order dynamics of the PDE system, with a sufficiently small modeling error between the FNN model and the reduced-order model. The FNN model was then used in an LPMC formulation to provide state predictions and achieve closed-loop stability. Finally, a diffusion-reaction process was used to illustrate the effectiveness of FNN modeling in predictive control of PDE systems.

Declaration of competing interest

The authors report no declarations of interest.

Acknowledgements

Financial support from the National Science Foundation and the Department of Energy is gratefully acknowledged.

References

Akesson, B., Toivonen, H., 2006. A neural network model predictive controller. J. Process Control 16, 937–946.
Ali, J.M., Hussain, M.A., Tade, M.O., Zhang, J., 2015. Artificial intelligence techniques applied as estimator in chemical process systems – a literature survey. Expert Syst. Appl. 42, 5915–5931.
Baker, J., Christofides, P.D., 2000. Finite-dimensional approximation and control of non-linear parabolic PDE systems. Int. J. Control 73, 439–456.
Balas, M.J., 1979. Feedback control of linear diffusion processes. Int. J. Control 29, 523–534.
Chollet, F., et al., 2015. Keras. https://keras.io.
Christofides, P.D., 2001. Nonlinear and Robust Control of PDE Systems: Methods and Applications to Transport-Reaction Processes. Birkhäuser, Boston.
Curtain, R.F., Pritchard, A.J., 1978. Infinite Dimensional Linear Systems Theory: Lecture Notes in Control and Information Sciences. Springer-Verlag, Berlin-Heidelberg.
Dubljevic, S., El-Farra, N.H., Mhaskar, P., Christofides, P.D., 2006. Predictive control of parabolic PDEs with state and control constraints. Int. J. Robust Nonlinear Control 16, 749–772.
Dufour, P., Toure, Y., Blanc, D.L.P., 2003. On nonlinear distributed parameter model predictive control strategy: on-line calculation time reduction and application to an experimental drying process. Comput. Chem. Eng. 27, 1533–1542.
Han, H., Wu, X., Qiao, J., 2013. Real-time model predictive control using a self-organizing neural network. IEEE Trans. Neural Netw. Learn. Syst. 24, 1425–1436.
Hassanpour, H., Corbett, B., Mhaskar, P., 2020. Integrating dynamic neural network models with principal component analysis for adaptive model predictive control. Chem. Eng. Res. Des. 161, 26–37.
Holmes, P., Lumley, J.L., Berkooz, G., Rowley, C.W., 2012. Turbulence, Coherent Structures, Dynamical Systems and Symmetry. Cambridge University Press.
Huang, B., Kadali, R., 2008. Dynamic Modeling, Predictive Control and Performance Monitoring: A Data-Driven Subspace Approach. Springer-Verlag, London.
Hunt, K., Sbarbaro, D., Zbikowski, R., Gawthrop, P., 1992. Neural networks for control systems – a survey. Automatica 28, 1083–1112.
Lao, L., Ellis, M., Christofides, P.D., 2014. Economic model predictive control of parabolic PDE systems: addressing state estimation and computational efficiency. J. Process Control 24, 448–462.
Mhaskar, P., El-Farra, N.H., Christofides, P.D., 2006. Stabilization of nonlinear systems with state and control constraints using Lyapunov-based predictive control. Syst. Control Lett. 55, 650–659.
Park, H., Cho, D., 1996. The use of the Karhunen-Loeve decomposition for the modeling of distributed parameter systems. Chem. Eng. Sci. 51, 81–98.
Shardt, Y.A.W., Huang, B., 2011. Closed-loop identification condition for armax models using routine operating data. Automatica 47, 1534–1537.
Shardt, Y.A.W., Huang, B., Ding, S.X., 2015. Minimal required excitation for closed-loop identification: some implications for data-driven, system identification. J. Process Control 27, 22–35.
Sirovich, L., 1987. Turbulence and the dynamics of coherent structures. I. Coherent structures. Q. Appl. Math. 45, 561–571.

Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Progr. 106, 25–57.
Wang, T., Gao, H., Qiu, J., 2016. A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control. IEEE Trans. Neural Netw. Learn. Syst. 27, 416–425.
Wu, Z., Rincon, D., Luo, J., Christofides, P.D., 2021. Machine learning modeling and predictive control of nonlinear processes using noisy data. AIChE J. 67, e17164.
Wu, Z., Rincon, D., Christofides, P.D., 2019a. Real-time adaptive machine-learning-based predictive control of nonlinear processes. Ind. Eng. Chem. Res. 59, 2275–2290.
Wu, Z., Tran, A., Rincon, D., Christofides, P.D., 2019b. Machine learning-based predictive control of nonlinear processes. Part i: theory. AIChE J. 65, e16729.