AIChE

A Linear Model Predictive Control Algorithm for Nonlinear Large-Scale Distributed Parameter Systems

Ioannis Bonis, Weiguo Xie, and Constantinos Theodoropoulos
School of Chemical Engineering and Analytical Science, University of Manchester, Manchester M13 9PL, U.K.

DOI 10.1002/aic.12626
Published online May 2, 2011 in Wiley Online Library (wileyonlinelibrary.com).

This work provides a framework for linear model predictive control (MPC) of nonlinear distributed parameter systems (DPS), allowing the direct utilization of existing large-scale simulators. The proposed scheme is adaptive and it is based on successive local linearizations of the nonlinear model of the system at hand around the current state and on the use of the resulting local linear models for MPC. At every timestep, not only the future control moves are updated but also the model of the system itself. A model reduction technique is integrated within this methodology to reduce the computational cost of this procedure. It follows the equation-free approach (see Kevrekidis et al., Commun Math Sci. *2003;1:715–762; Theodoropoulos et al.,* Proc Natl Acad Sci USA. *2000;97:9840-9843), according to which the equations of the model (and consequently of the simulator) need not be given explicitly to the controller. The latter forms a "wrapper" around an existing simulator using it in an input/output fashion. This algorithm is designed for dissipative DPS, dissipativity being a prerequisite for model reduction. The equation-free approach renders the proposed algorithm appropriate for multiscale systems and enables it to handle large-scale systems.* © 2011 American Institute of Chemical Engineers *AIChE J,* 58: 801–811, 2012
Keywords: model reduction, successive linearization, adaptive predictive control, equation-free, dominant subspace, black-box simulators

Introduction

Distributed parameter systems (DPS) arise naturally in engineering practice. The growth in a number of cutting-edge technology-based industrial sectors ranging from nanotechnology to semiconductor manufacturing and crystallization, polymerization as well as biotechnological systems is paved by developments in modeling, analysis and design of DPS. DPS are based on sets of partial differential equations (PDEs), which are able to accurately express their nonlinear distributed dynamic behavior. In recent years much attention has been given to the simulation of these systems and as a result significant advances are noted, subsequently influencing industrial systems' design. Extending simulation results to design is not straightforward.[1] Of particular interest are large-scale DPS and recently multiscale systems, which couple various spatio-temporal scales. The use of multiscale simulators to perform system-level tasks is not trivial.[2] The same is true for the many, essentially, black-box simulators, which are available as commercial packages or as legacy codes. Equation-free methods can be used to overcome the obstacle of the system equations not being available to the user or at all in closed form.[3] It is clear that the next natural step is to address issues of optimization and control for black-box DPS and for multiscale systems alike.

Correspondence concerning this article should be addressed to C. Theodoropoulos at k.theodoropoulos@manchester.ac.uk.

Optimization technology is used to determine the optimal conditions of a system's operation, whereas optimal control is employed to drive the system to that set of conditions and to ensure that it will not deviate significantly. The field of PDE-constrained optimization has undergone significant advances and is still developing rapidly.[4] Recently we have developed a suite of algorithms for the optimization of large-scale input/output nonlinear DPS and of multiscale systems.[5–7] Here we extend these concepts for predictive control.

The control of DPS is a challenging task. Due to the complexity of the system at hand, the use of conventional algorithms, intended for lumped parameter systems, may be prohibitive. In general, highly distributed control variables are involved in the problem and the control decisions are implemented in a spatially distributed network of actuators. However, it has been observed in many systems of engineering interest that the dominant dynamic behavior can be attributed to a small number of modes, which in general are much less than the (large) number of variables of the system. This notion has been used repeatedly in the context of advanced control methodologies in conjunction with model order reduction.[8,9] Christofides and collaborators have worked extensively on the control of DPS, proposing a set of algorithms for the feedback and robust control of various classes of systems modeled by PDEs.[10] Model reduction-based approaches for model predictive control (MPC) of DPS include modal decomposition,[11] approximate inertial manifold methods,[12,13] Galerkin's method,[14] reduced basis methods following the Lagrange approach,[15] proper orthogonal decomposition (or Karhunen–Loeve expansion)[16–18] and equation-free methods.[19–21]

Research has been also directed towards designing new optimization algorithms, appropriate for this class of problems[22,23] and towards modifying existing control algorithms.[24] Another trend is to replace the nonlinear models with stochastic counterparts or hybrid models.[25] Finally, there is the option of identifying a linear model which captures the behavior of a DPS at the region of operation.[26] The last approach usually incurs the least computational cost, especially if it is used in the context of multiparametric MPC.[27]

The use of linear models for the control of nonlinear systems does not always produce satisfying results. Linear models used in MPC cannot generally describe the behavior of highly nonlinear systems over sizeable regions. Even nonlinear models are only approximations of the reality and include inaccuracies. Using better models for control intuitively leads to better closed loop behaviors. There have been many approaches towards the predictive control of nonlinear systems. A natural extension of MPC is nonlinear MPC (NMPC). Many different formulations have been presented over the years.[28,29] NMPC has the added advantage of employing models of increased accuracy, but also the disadvantage of significantly higher computational cost. Furthermore, the computation of the sequence of future control moves follows as solution of nonlinear optimization problems (NLPs), which are generally nonconvex and might exhibit multiple optima. Another way of coping with nonlinearity is to use multiple models instead of a single one.[30,31] Obviously, in such formulations the optimal determination of model switching policies is as crucial as the accuracy of the models themselves. NMPC based on successive linearization has been proposed as a remedy for the nonconvexity of the optimization subproblem.[32–34] By adaptively linearizing the original nonlinear model on a nominal trajectory, a local linear model is produced, which is used in an MPC algorithm.

In this work, we present a predictive control algorithm for nonlinear DPS, which is based on a combination of on-line model reduction and successive linearizations. At each sampling time, the dominant dynamics of the system are identified and a basis for the low-dimensional corresponding subspace is computed. The dependent variables of the DPS are projected onto this subspace, yielding the state variables which will be considered for the current time interval. A low-order linear approximation of the original nonlinear model is found by performing a linearization using the reduced Jacobian around the current state. This linearization is in the temporal domain, i.e. we consider adaptive successive linearizations along the temporal trajectories of the (nonlinear) right-hand sides of the dynamic PDEs at hand, and not only at stationary points as in previous equation-free control approaches.[19–21] The reduction of the dimensionality of the problem, both in terms of number of state variables and of size of the Jacobian matrix, is performed efficiently on-line, in the least computationally expensive way. The resulting (local) linear model is considered invariant over the prediction horizon; hence it can be used to formulate a quadratic optimization problem, from which the sequence of future system inputs will be computed.

This framework makes the use of black-box simulators possible, even for multiscale systems, as it only requires the availability of an input/output time integrator for the DPS at hand. In cases where explicitly coping with process nonlinearity is considered advantageous, the same principles for model reduction can be exploited for the formulation of a reduced NLP problem in the context of NMPC. We have used such an approach for dynamic optimization[7] which underpins the links with a NMPC formulation.

The rest of the article is organized as follows. Firstly, the conventional MPC method is briefly presented as well as the MPC set-up for our proposed algorithm. Next, we present the proposed algorithm, we address some implementation issues and present a computationally efficient modification. We apply the methodology for the control of tubular reactors, delineating the results. Finally some concluding remarks are given.

MPC formulation

MPC is admittedly the most widely used among the advanced control strategies, receiving attention both in academia and industry. One of the most useful features of this framework is the ability to handle input and output constraints. The central idea is that a model of the system at hand, which can predict future outputs (y) over a given prediction horizon (N_p) subject to initial conditions and inputs, is used to formulate an optimization problem which results in the computation of a sequence of future inputs over a control horizon (N_c).[35–37] This computation is performed at every sampling time, however following the receding horizon approach only the first move in the sequence is actually implemented. The optimization problem

minimizes the anticipated deviation from the set-point as well as the control energy required over the control horizon. For implementation details, see the books of Camacho and Bordons[35] and Wang.[38]

MPC can be formulated in discrete or continuous time, using linear or nonlinear models. In the conventional formulation,[35] a linear, discrete time, state-space model (Eq. 1) is employed.

$$\begin{aligned} x_{k+1} &= A_{\mathrm{d}}x_k + B_{\mathrm{d}}u_k \\ y_k &= Cx_k + Du_k \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^{n_{st}}$ is the incremental state variable vector: $x = \tilde{x}_{nom} - \tilde{x}$, $\tilde{x}$ being the state vector and $\tilde{x}_{nom}$ the corresponding vector of nominal states which will be defined later on in this section, $y \in \mathbb{R}^{n_{out}}$ is the system output, $u \in {}^{n_{in}}$ is the system input, n_{in} is the number of inputs, n_{st} is the number of state variables, n_{out} is the number of outputs, x_k is the state at the k-th sampling interval, ($k = \{1, ..., T_f/T_s+1\}$, where t_f is the simulation time and t_s the sampling time) and A_d, B_d, C, D are the matrices involved in the state space representation of the system ($A_d \in \mathbb{R}^{n_{st} \times n_{st}}$, $B_d \in \mathbb{R}^{n_{st} \times n_{in}}$, $C \in \mathbb{R}^{n_{out} \times n_{st}}$ and $D \in \mathbb{R}^{n_{out} \times n_{in}}$). These state and output matrices are assumed to be both controllable and observable. The optimization problem solved at a sampling instance t, is[39]:

$$\begin{aligned} U = \arg\min_{U} \Bigg[& \sum_{k=1}^{N_{\mathrm{p}}} \left\| \left(\hat{y}(t+kT_{\mathrm{s}}|t) - r(t+kT_{\mathrm{s}})\right) \right\|_{Q_{\mathrm{w}}}^2 \\ & + \sum_{k=1}^{N_{\mathrm{c}}} \left\| \left(u(t+kT_{\mathrm{s}}|t) - u(t+(k-1)T_{\mathrm{s}}|t)\right) \right\|_{R_{\mathrm{w}}}^2 \Bigg] \\ s.t. \quad & \hat{x}(t+(k+1)T_{\mathrm{s}}|t) = A_{\mathrm{d}}\hat{x}(t+kT_{\mathrm{s}}|t) \\ & + B_{\mathrm{d}}u(t+kT_{\mathrm{s}}|t) \quad k = 1, 2, \ldots N_{\mathrm{p}} \\ & \hat{y}(t+kT_{\mathrm{s}}|t) = C\hat{x}(t+kT_{\mathrm{s}}|t) + Du(t+kT_{\mathrm{s}}|t) \\ & u^{\mathrm{lower}} \le u \le u^{\mathrm{upper}} \end{aligned} \tag{2}$$

where Q_{w} and R_{w} are weighting matrices of appropriate dimensions, $\hat{x}, \hat{y}$ denote predicted values, $(\cdot \mid t)$ denotes the current time interval t, r is the reference signal and $U = [u(t + \mathrm{Ts}|t)^T \; u(t + \mathrm{Ts}2|t)^T \; \ldots \; u(t + N_c\mathrm{Ts}|t)^T]^T$ the sequence of future inputs, which satisfies (2). As mentioned, only $u(t + \mathrm{Ts}|t)$ is actually implemented and all the other elements of U are discarded.

It is clear that this approach is based on a linear model of the form of Eq. 1. However most engineering models are nonlinear. A general description of a nonlinear model is:

$$\begin{aligned} \dot{\tilde{x}} &= f(\tilde{x}(t), \tilde{u}(t), t), \quad \tilde{x}(t_0) = \tilde{x}_0 \\ \tilde{y} &= g(\tilde{x}(t), \tilde{u}(t), t) \end{aligned} \tag{3}$$

There are many ways of extracting a linear model corresponding to the system (3).[40] One possibility is to perform Jacobian linearization at a nominal point[41] $\{\tilde{x}_{\mathrm{nom}}(t), \tilde{u}_{\mathrm{nom}}(t)\}$ which satisfies Eq. 3. This is typically the set point and/or a steady state of the nonlinear model. We can consider any point in state space as a perturbation of the nominal point:

$$\tilde{u}(t) = \tilde{u}_{\mathrm{nom}} + u(t) \quad \text{and} \tag{4}$$

$$\tilde{x}(t_0) = \tilde{x}_{\mathrm{nom}}(t_0) + x(t_0) \tag{5}$$

for small $\|x(t_0)\|$ and $\|u(t)\|$, with $\|u(t)\| = \sup \|u(t)\|_2$.

For small $\|x\|$ and $\|y\|$ we can define

$$\tilde{x}(t) = \tilde{x}_{\mathrm{nom}}(t) + x(t) \tag{6}$$

$$\tilde{y}(t) = \tilde{y}_{\mathrm{nom}}(t) + y(t) \tag{7}$$

Taking a Taylor expansion of system (3) using incremental variables and assuming that f and g are smooth, we have

$$\begin{aligned} \dot{x}(t) &= \left.\frac{\partial f}{\partial \tilde{x}}\right|_{(\tilde{x}_{\mathrm{nom}}, \tilde{y}_{\mathrm{nom}})} \cdot x(t) + \left.\frac{\partial f}{\partial \tilde{u}}\right|_{(\tilde{x}_{\mathrm{nom}}, \tilde{y}_{\mathrm{nom}})} \cdot u(t) \\ y(t) &= \left.\frac{\partial g}{\partial \tilde{x}}\right|_{(\tilde{x}_{\mathrm{nom}}, \tilde{y}_{\mathrm{nom}})} \cdot x(t) + \left.\frac{\partial g}{\partial \tilde{u}}\right|_{(\tilde{x}_{\mathrm{nom}}, \tilde{y}_{\mathrm{nom}})} \cdot u(t) \end{aligned} \tag{8}$$

Or equivalently:

$$\begin{aligned} \dot{x}(t) &= A_{\mathrm{c}}x(t) + B_{\mathrm{c}}u(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{9}$$

with initial condition $x(t_0) = x_o$ and the system Jacobian, A_{c}, and actuator effect, B_{c}, given by:

$$A_{\mathrm{c}} = \left.\frac{\partial f}{\partial \tilde{x}}\right|_{(\tilde{x}_{\mathrm{nom}}, \tilde{y}_{\mathrm{nom}})} \tag{10}$$

$$B_{\mathrm{c}} = \left.\frac{\partial f}{\partial \tilde{u}}\right|_{(\tilde{x}_{\mathrm{nom}}, \tilde{y}_{\mathrm{nom}})} \tag{11}$$

The system (9) is in continuous time and needs to be transformed in discrete time as per Eq. 1. Transforming the continuous time model to a discrete-time one and considering zero-order hold for the inputs we can compute A_{d} and B_{d} from A_{c}, B_{c} and the sampling time (T_{s}). Note here that the prediction horizon consists of a number of sampling time steps. It can be shown that[42]:

$$A_{\mathrm{d}} = e^{A_{\mathrm{c}}T_{\mathrm{s}}} \tag{12}$$

$$B_{\mathrm{d}} = \int_0^{T_{\mathrm{s}}} e^{A_{\mathrm{c}}\tau} B_{\mathrm{c}} d\tau \tag{13}$$

Note that matrices C and D remain unaltered by the transformation.

The Proposed Algorithm

Model reduction

The basis of this work is model reduction, which is based on dissipation which many engineering systems exhibit.[43] This technique is extending the concepts of the "equation-free" approaches for control applications[19–21,44,45] which are inspired by the Recursive Projection Method.[46] These studies have exploited equation-free methodologies to produce low-order models, which correspond to equilibrium setpoints and can be further used for linear quadratic regulation, pole placement and feedback linearization. In this work the

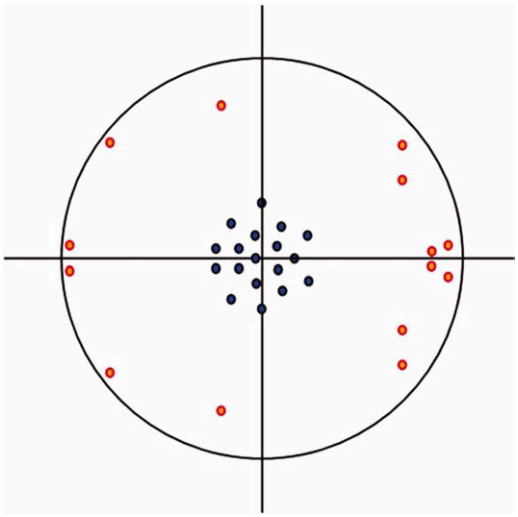

Figure 1. Eigenspectrum of a discretized dynamic system, exhibiting a clear separation of scales.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

linearization is adaptive, it is repeated along the trajectory and is not restricted to an equilibrium point, thus extending the ideas on dynamic optimization presented by Theodoropoulos and Luna-Ortiz.[7]

The dynamic behavior of nonlinear systems given by Eq. 3 is simulated using time integrators of the form

$$\dot{\tilde{x}}_i = F(\tilde{x}_{i-1}, \tilde{u}_{i-1}, t_{i-1}) \tag{14}$$

Function $F(.)$ in Eq. 14 can represent a black box simulator, which the user can only run for a given set of initial conditions, with given independent (control) variables profiles, for a set time. For most engineering systems there exists a "commodity" simulator, which one can use for accurate simulations. Typically, the interaction of such a code with the user is limited.

The eigenspectrum, i.e. the set of eigenvalues of the scheme presented in Eq. 14 is shown in Figure 1. Eigenvalues outside the unit circle would correspond to the unstable modes of the system (14). The model reduction technique presented here is efficient if the assumption holds that only a relatively small number of eigenvalues, m, are close to the unit circle, either inside or outside. This is important, as those are considered as the dominant modes for the system. Let K_δ a disk smaller than the unit circle, so that its center is the origin (0,0) and its radius is equal to (1-δ), where $\delta > 0$ is a positive parameter, and the eigenvalues of the system are ordered, so that

$$|\mu_1| \geq \ldots \geq |\mu_m| > 1 - \delta \geq |\mu_{m+1}| \geq |\mu_{\mathrm{n_{st}}}| \tag{15}$$

In this formulation, $\mu_1,\ldots,\mu_m$, which are outside K_δ, are (heuristically) the dominant modes of the system. Let $\mathbf{P}$ denote the maximal invariant subspace of the Jacobian, corresponding to those modes and $\mathbf{Q}$ its orthogonal complement, so that:

$$\mathbf{P} \oplus \mathbf{Q} = \mathbb{R}^{n_{\mathrm{st}}}, \quad \text{or} \quad \mathbb{R} = P\mathbb{R}^{n_{\mathrm{st}}} + Q\mathbb{R}^{n_{\mathrm{st}}} \tag{16}$$

where P and Q are the orthonormal projectors onto the subspaces $\mathbf{P}$ and $\mathbf{Q}$ respectively, $P,Q \in \mathbb{R}^{n_{st} \times n_{st}}$.

As any valid projector, P satisfies $P^2 = P$. And since $Q = I - P$ from (16), also $PQ = 0$. This way, any vector $x \in \mathbb{R}^{n_{st}}$ can be decomposed in two unique components: $x = Px + Qx = p + q$, $p \in \mathbf{P}$ and $q \in \mathbf{Q}$. In this approach, we assume that $\mathbf{P}x$ adequately approximates x.

Let $Z \in \mathbb{R}^{n_{st} \times m}$ be an orthonormal basis for $\mathbf{P}$. The projectors P and Q can be computed from:

$$P = ZZ^{\mathrm{T}}, \quad Q = I - ZZ^{\mathrm{T}}, \quad \text{with} \quad Z^{\mathrm{T}}Z = I \tag{17}$$

In the formulation that is presented in this section, the projectors P and Q, which are high dimensional, need not be computed explicitly. The basis Z is computed using matrix-free methods, which do not explicitly require the Jacobian. At every sampling time, Arnoldi method or subspace iterations are employed to compute an orthonormal basis for the dominant subspace of the system.

The clustering of the eigenvalues in Figure 1 denotes separation of temporal scales in the dynamics of the system under examination. Two clusters stand out: one being close to the origin and the other outside of K_δ. The "spectral gap" between these clusters need not be as prominent in practice, for the assumption of the existence of a dominant subspace to hold. The spectral gap is tightly related to the dissipativity of the system. The use of spectral decomposition methods in general in hyperbolic systems of PDEs is not advised, since the number of modes required for the extraction of an accurate model of ODEs may be large.[10]

Let it be noted that there might be some collateral benefits from the truncation of some modes. Using conventional techniques which work with full models (e.g., an implementation of the conventional NMPC algorithm), additional issues might arise due to the fact that multiscale systems exhibit stiffness if the characteristic temporal scale of the system is smaller than the one of the fastest scale.[47] Thus, the use of conventional simulation methods within the NMPC framework might fail, depending on the characteristics of the system at hand and the examined region in the parameter space.

The proposed algorithm disregards the evolution of the fast modes in the computations performed in every time interval. That is shown to be of minimal importance for closed-loop stability but does affect the controller's predictive capability for some classes of systems.[11] It is important to note here that the chosen prediction horizon needs to be small enough so that the linearization around the current state is valid for the whole horizon. Moreover, the sampling time needs to be large enough so that the fast modes decay (i.e., they become enslaved to the dominant/slow modes) hence they can be safely ignored. It should be, however, mentioned that the sampling time is normally much smaller than the prediction horizon. Possible inaccuracies are compensated by the adaptive nature of the successive linearization scheme.

Successive linearization MPC

The ideas for model reduction described in the previous section are applied to produce a linear low-order model which can be used for predictive control. The basis of this approach is successive linearization (SL). The idea is that linearization based on the system's Jacobian is performed every time interval, so that the nonlinear model can be approximated by a

Table 1. Pseudo-Code for Algorithm 1

1. Initialize the control problem: choose horizons N_p, N_c, weights, Q_w and R_w and set initial conditions for $\tilde{x}$ and $\tilde{u}$
2. For every time interval:
   i. Apply the 1st control decision to the system, as calculated from the previous time step
   ii. Measure the output of the system and estimate the current state
   iii. Formulate a new controller and compute next control action as in (linearization) Table 2 (Algorithm 2)
   iv. Solve the optimization subproblem (2) and compute the next N_c control decisions

linear one. The Jacobian of the system is defined in Eq. 10. However in the proposed formulation this matrix will not be calculated; only products of the approximate Jacobian F_x with vectors will be used. Different notation is used to stress that F_x implies the use of the integrator, rather than explicit use of the system equations as in A_c. This linearization can be performed on a nominal trajectory. Whereas in the conventional formulation of MPC presented in section 2 there is only one linear model, SL results in the computation of a sequence of linear models. One advantage of SL over NMPC is that the resulting optimization problem solved online is convex.

In the proposed method, the linear models are of reduced dimension and are calculated online, following the procedure described in (linearization) Algorithm 2. On initialization, only the horizons N_p, N_c, as well as the weights, Q_w and R_w are set along with the initial conditions for $\tilde{x}$ and $\tilde{u}$. At every sampling time, the invariant subspace corresponding to the dominant modes of the system is identified using the nonlinear model (3), as described in the previous section. Only matrix (Jacobian)-vector products are needed, which can be efficiently calculated with numerical directional perturbations:

$$F_x v \approx \frac{1}{2\varepsilon}(F(\tilde{x}+\varepsilon v,\tilde{u},t)-F(\tilde{x}-\varepsilon v,\tilde{u},t)) \tag{18}$$

where $\varepsilon \in \mathbb{R}^+$ is the perturbation size, $v \in \mathbb{R}^{n_{st}}$ the vector which needs to be multiplied with the Jacobian. The dynamic simulator available is used to perform these perturbations by perturbing the initial conditions and running the simulator for a certain reporting time. The states $F(.)$ computed are used in the central finite difference scheme as in Eq. 18.

The orthonormal basis $Z \in \mathbb{R}^{n_{st}\times m}$ which results from this procedure is used for projecting the state vector onto the dominant subspace. A new, reduced state vector is defined:

$$\xi = Z^T x \tag{19}$$

where $\xi \in \mathbb{R}^m$. The basis is also used for the computation of a reduced Jacobian, $H = Z^T F_x Z$. Firstly, the product $F_x Z \in \mathbb{R}^{n_{st}\times m}$ is computed by columns as in Eq. 18. m perturbations are performed, one for every column of Z. In the j-th (of m) perturbations, the vector v is actually the j-th column of Z. Secondly, $F_x Z$ is pre-multiplied with Z^T, for the desired reduced Jacobian to be formed. It can be shown, that the reduced Jacobian of the continuous-time system is $A_{c,red} = I - H$. The sensitivity matrix describing the actuator effect, $B_{c,red}$, is computed directly with numerical perturbations on the inputs using the dynamic simulator. The projection onto the dominant subspace is $B_{c,red} = Z^T B_c$. Note here that the low-order matrices $A_{c,red}$ and $B_{c,red}$ are time-dependent since they are recalculated at each sampling time to provide a linearized description of the system in continuous time. to obtain the corresponding description in discrete time, we can rewrite Eqs. 12 and 13 at each sampling time j:

$$A_k = e^{A_{c,red}T_s} \tag{20}$$

$$B_k = \int_0^{T_s} e^{A_{c,red}\tau} B_{c,red} d\tau \tag{21}$$

where subscript k denotes that those matrices are valid for the time-interval $[k, k+1]$, For a more compact notation, these subscripts in matrices A, B, C and D will be omitted for the rest of the manuscript. Equations 20 and 21 define a reduced model, which is the projected onto the **P** counterpart of the model (1):

$$\begin{aligned} \xi_{k+1} &= A\xi_k + Bu_k \\ y_{k+1} &= CZ^T\xi_k + Du_k \end{aligned} \tag{22}$$

Note that in the proposed algorithm, linearization is performed only once per sampling time, at the current state, whereas in the SL approach, linearization along a nominal trajectory is considered, the current state typically being the first point of this trajectory. Thus, in the current formulation, matrices A and B are considered time-invariant within each sampling time, but they are recalculated before each optimization problem. The evolution of the matrices throughout the prediction horizon is neglected.

The reduced linear model (22) can be used to formulate an optimization problem in every timestep, which is the one shown in Eq. 2 projected onto the dominant subspace of the system corresponding to the current sampling interval:

$$\begin{aligned} U = \arg\min_U \Bigg[& \sum_{k=1}^{N_p} \|(\hat{y}(t+kT_s|t) - r(t+kT_s))\|^2_{Q_w} \\ & + \sum_{k=1}^{N_c} \|(u(t+kT_s|t) - u(t+(k-1)T_s|t))\|^2_{R_w} \Bigg] \\ s.t. \quad & \hat{\xi}(t+(k+1)T_s|t) = A\hat{\xi}(t+kT_s|t) \\ & + Bu(t+kT_s|t), k = 1,2,\ldots N_p \\ & \hat{y}(t+kT_s|t) = CZ^T\hat{\xi}(t+kT_s|t) + Du(t+kT_s|t) \\ & u^{lower} \le u \le u^{upper} \end{aligned} \tag{23}$$

The dimension of the problem is reduced, as the number of equality constraints is reduced. This leads to big savings in

Table 2. Pseudo-Code for (Linearization) Algorithm 2

1. Compute a basis Z the dominant subspace **P** of the current state by employing Arnoldi method using directional perturbations on the nonlinear model (14) as per Eq. (18)
2. Compute the reduced deviation state variable from Eq. (19).
3. Calculate the matrices A, B, C, D for the reduced discrete state space model (22) using directional perturbations to the direction of Z.
4. Formulate the underlying QP problem (23) using the updated model

Table 3. Pseudo-Code for Algorithm 3

1. Initialize the control problem: choose horizons N_p, N_c, weights, Q_w and R_w and set initial conditions for $\tilde{x}$ and $\tilde{u}$
2. For every time interval:
   i. Apply the 1st control decision to the system, as calculated from the previous time step
   ii. Measure the output of the system and estimate the current state
   iii. *If* conditions (24) and (25) are met:
   Formulate a new controller and compute next control action as in (linearization) Algorithm 2
   Else:
   use the model from the previous sampling time
   iv. Solve the optimization subproblem (2) and compute the next N_c control decisions

computational time and memory usage. The proposed algorithm is outlined in Table 1. This algorithm is based on the derivation of a linear, discrete state-space model as explained by Algorithm 1 (Table 1).

The quadratic subproblem defined in Eq. 23, is a reduced form of the standard finite-horizon linear MPC formulation. It can be shown that if the (full-scale) QP subproblem is feasible then its solution satisfies the reduced QP subproblem as well. However, it is known that the QP subproblem may, in general, be infeasible if the control horizon chosen is not large enough.[48] Even if this is the case, the minimal-time or the soft constraint approaches can be used to handle it.[49]

A modification for the reduction of computational cost

The most computationally expensive step of the algorithm is updating the basis for the dominant subspace of the system. As mentioned in section 3.1, iterative matrix-free techniques are used for that computation, which are known to scale well with problem size. In particular, we use Arnoldi method in this work. In this section we present some ways of reducing the computational cost of this part of the algorithm. One way this reduction can be achieved is by "warm-starting". An initial guess for this basis could be $\Delta x = x_k - x_{k-1}$, extending the concept of the power method.[50] After the first iteration is completed, the basis is practically updated rather than recalculated. Hence, a zero order continuation can be used for warm starting: the dominant mode (first column) of the basis from the previous sampling time can be used for the initialization of the Arnoldi method.

The cost of computing Z can be further reduced by decreasing the accuracy to which Arnoldi converges. This is a nontrivial task as it affects the quality of the resulting model used for control. Tolerance for convergence to the dominant subspace can be seen as a tuning parameter. It is advised that some offline tests are performed, which should confirm that even with an increased tolerance, the dynamics of the system under examination are sufficiently captured. Another way of reducing cost is to increase the sampling time. This should also be treated with caution. Firstly, the dominant dynamics of the underlying system could be fast enough to require small time intervals; secondly, since in this work we effectively consider adaptive linearization in a neighborhood of the current state, for the linearization to hold, the time interval should be "small enough".

The heuristics mentioned so far do reduce the computational cost, however computing Z at every sampling interval is in general expansive. A drastic way of reducing cost is to consider conditional updates of the basis Z; instead of updating Z at every sampling time, we can only update it when significant "jumps" in parameter space are noticed:

$$\frac{\left\|x_{\text{upd}} - x_k\right\|}{\left\|x_{\text{upd}}\right\|} \geq \varepsilon_{\text{upd}} \tag{24}$$

where x_{upd} is the state at the last point where Z was updated and ε_{upd} the heuristically defined *affinity* parameter (tolerance). This condition indicates that if the underlying nonlinear system is smooth enough, it exhibits similar dynamics for states, which do not differ significantly.

It would be prudent and relatively inexpensive, to also set a maximum number of sampling intervals, n_{upd}, between two consecutive updates of Z:

$$k - k_{\text{upd}} < n_{\text{upd}} \tag{25}$$

Therefore, we can implement Algorithm 1 as presented in the previous section, with the model of the plant being updated if and only if conditions (24) and (25) are met. This version is presented in Table 3 as Algorithm 3.

Case Studies

In this section we apply the proposed methodology for the control of a tubular reactor with recycle, where a first order, irreversible reaction takes place. The model for the reactor consists of two nonlinear (parabolic) PDEs, which in dimensionless form are[51,52]:

$$\begin{aligned}\frac{\partial x_1}{\partial t} &= \frac{1}{Pe_1}\frac{\partial^2 x_1}{\partial y^2} - \frac{\partial x_1}{\partial y} - Da \cdot x_1 \exp\left(\frac{\gamma x_2}{1+x_2}\right)\\ \frac{\partial x_2}{\partial t} &= \frac{1}{Pe_2}\frac{\partial^2 x_2}{\partial y^2} - \frac{\partial x_2}{\partial y} + CDa \cdot x_1 \exp\left(\frac{\gamma x_2}{1+x_2}\right)\\ &\quad + \beta(x_{2w} - x_2)\end{aligned} \tag{26}$$

With boundary conditions

$$\begin{aligned}\frac{\partial x_1}{\partial y} &= -Pe_1\left[(1-r)x_1^0 + rx_1\big|_{y=1} - rx_1\big|_{y=0}\right]\\ \frac{\partial x_2}{\partial y} &= -Pe_2\left[(1-r)x_2^0 + rx_2\big|_{y=1} - rx_2\big|_{y=0}\right]\end{aligned} \tag{27}$$

at $y = 0$ and

$$\begin{aligned}\frac{\partial x_1}{\partial y} &= 0\\ \frac{\partial x_2}{\partial y} &= 0\end{aligned} \tag{28}$$

at $y = 1$.

Here y is the dimensionless longitudinal coordinate, x_1 is the dimensionless concentration and x_2 is the dimensionless temperature. r is the recycle ratio, Da is the Damköhler number, Pe_1 is the Pecklet number for mass transport and Pe_2 for heat transport, β a dimensionless heat transfer coefficient, C is the dimensionless adiabatic temperature rise, x_{2w} the dimensionless adiabatic wall temperature, γ the

Figure 2. Schematic representation of a tubular reactor with eight cooling zones.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

dimensionless activation energy. A schematic of the reactor is given in Figure 2. The values for the parameters chosen are: are $Da = 0.1$, $Pe_1 = Pe_2 = 7.0$, $\gamma = 10.0$, $\beta = 2.0$, $C = 2.5$, $x^0{}_1 = 0$, $x^0{}_2 = 0$. Such tubular reactors are commonly used in chemical- and bio-processing. For the examples presented, the sampling time was 0.01, Arnoldi method was warm-started as presented in section 3.2, but tolerance was not increased (and was set to 10^{-6}). For the results presented using Algorithm 3, the affinity parameter ε_{upd} was set to 10^{-2} and n_{upd} was 50.

A single output was considered for control, which was the exit temperature from the reactor. The wall temperature, $x_{2\text{w}}$, was the manipulated variable. In specific, we consider 8 cooling zones in the jacket of the reactor, whose temperature can be controlled independently. Hence $x_{2\text{w}}$ varies spatially:

$$x_{2\text{w}}(y) = \sum_{j=1}^{8} \big(H(y - y_{j-1}) - H(y) - y_i\big)u_j(t) \tag{29}$$

With $y_i = i / 8$, $i = 0, 1, \ldots, 8$, $x_{2w_j}(t)$, $j = 1, 2, \ldots, 8$, is the dimensionless temperature of the cooling zone for time t and $H(.)$ is the Heaviside step function:

$$H(y) = \begin{cases} 0, & y<0 \\ 1, & y \geq 0 \end{cases} \tag{30}$$

Equations 26–28 were semidiscretized in space with the Finite Element method. The number of nodes selected was 16, resulting in 32 ODEs (and 32 dependent variables: dimensionless concentrations and dimensionless temperatures). The dynamic simulator for the system was used by the controller as black box. In particular no access to the discretized version of the equations was given to the controller. Only state information was passed to the controller, which was wrapped around the time integrator.

This system exhibits an interesting parametric behavior. It is stable for no recycle ($r = 0$) as shown in Figure 3, but a Hopf bifurcation there exists for $r = 0.5$ and the system exhibits sustained oscillations (Figure 4).

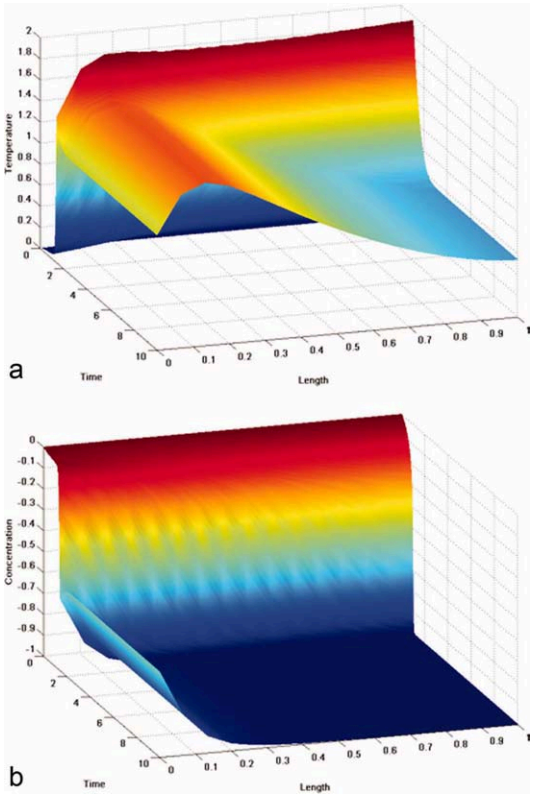

Figure 3. Dynamic profile for the dimensionless (a) temperature and (b) concentration for the open loop reactor behavior with $r = 0$.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

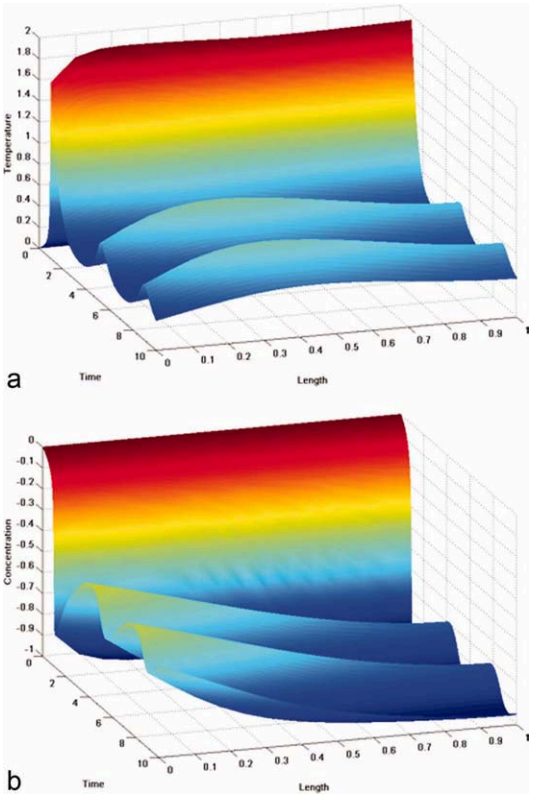

Figure 4. Dynamic profile for the dimensionless (a) temperature and (b) concentration for the open loop reactor with $r = 0.5$.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

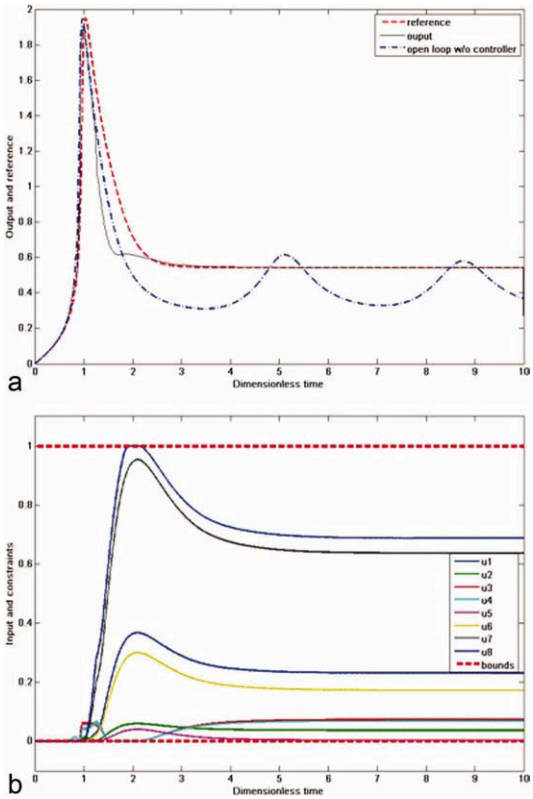

Figure 5. (a) Dynamic behavior of the open-loop system (dash-dotted line), reference output (dashed line) and closed-loop output (solid line) and (b) the corresponding computed cooling profiles.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Stabilization of the reactor

A natural choice for a case study to illustrate the efficiency of the proposed control algorithm, is the stabilization of the reactor for $r = 0.5$. Namely, the objective for this case is for the exit temperature from the reactor, which exhibits sustained oscillations, to be the equal to the case with no recycle. The manipulated variables are the temperatures of the cooling zones, u_j, which have the bounds:

$$0 \leq u_j \leq 1, \quad j = 1, 2, \ldots, 8 \tag{31}$$

The size of the subspace chosen was $m = 4$. The horizons chosen were $N_p = N_c = 7$. The sampling interval was set to 0.01 (dimensionless).

The controller of Algorithm 1 succeeded in effectively stabilizing the system. Figure 5a presents the output behavior for the closed loop system, vs. the reference trajectory and the open loop oscillating behavior, whereas Figure 5b presents the corresponding input values calculated by the controller. In Figure 5, the dynamic closed loop behavior of the state variables is presented. The corresponding closed loop behavior of the states is illustrated in Figure 6. The implementation of Algorithm 3 also gave similar results, as presented in Figure 7, but with a much lower computational cost.

Destabilization of the reactor

As a second case study, we consider destabilizing the reactor, driving the exit temperature for the case with no recycle to exhibit sustained oscillations equal to the case with $r = 0.5$. As in the previous case, the manipulated variables are the temperatures of the 8 cooling zones, with bounds:

$$-1 \leq u_j \leq 1, \quad j = 1, 2, \ldots, 8 \tag{32}$$

In this case the objective is impossible to reach unless the values of the input variables are allowed to change sign at operation time. Due to the complexity of the dynamics, more modes are required to capture it relatively accurately, so the size of the subspace considered in this case was $m = 10$. In addition, longer horizons were needed: $N_p = N_c = 15$. In this case using the conventional MPC formulation presented in section 2, which requires the computation of matrices of the state space representation at a system's steady state is not possible, since such a steady state does not exist.

As in the previous case, both versions of the algorithm succeeded in tracking the set point. Figures 8a and 10a depict the output behavior for the closed loop system vs. the reference trajectory while Figures 8b and 10b present the corresponding input values for the 2 versions of the algorithm. In Figure 9, the dynamic closed loop behavior of the state variables is presented.

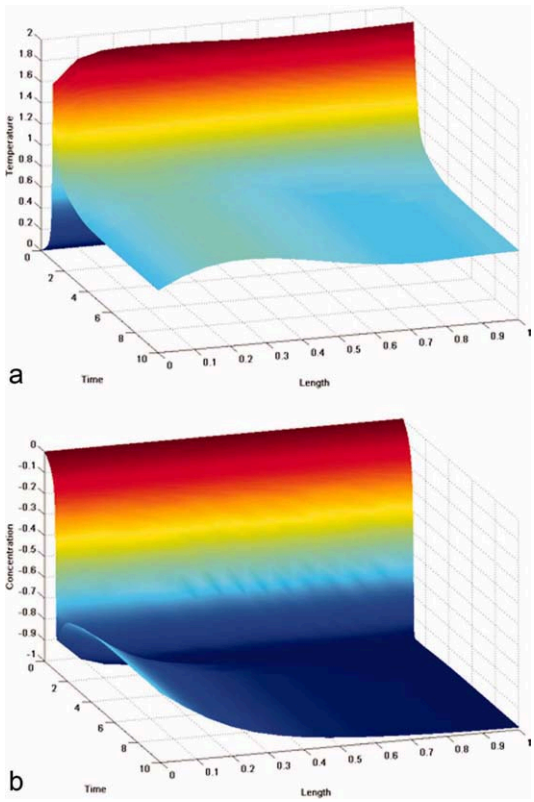

Figure 6. Dynamic profiles for the dimensionless (a) concentration and (b) temperature throughout the reactor for the closed loop system.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

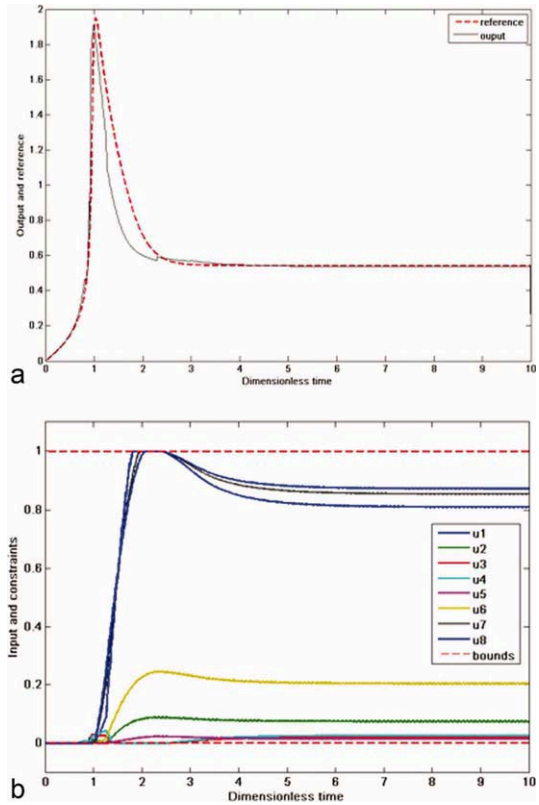

Figure 7. (a) Dynamic behavior of the reference output (dashed line) and the closed loop output (solid line) for the conditional model update case and (b) Corresponding computed cooling zones' profiles.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Computational cost

The applications implementing the proposed algorithms resulted in computational gains of 30–80% in comparison to the successive linearization NMPC. Memory usage was also lower. Algorithm 3, which implements the conditional update of the basis for the dominant dynamics of the system resulted in reduction of computational time varying from 70 to 85% in comparison to the conservative version (Algorithm 1). In absolute times, this translates to 3.6 CPU minutes for Algorithm 3 vs. 21.5 CPU minutes for Algorithm 1 in the stabilization case study. The corresponding times for the destabilization case are 3.0 and 13.2 CPU minutes for implementations of Algorithms 3 and 1 respectively. Those times correspond to an implementation in MATLAB (R2007b) single-threaded code and execution on a workstation based on 2 Intel Xeon 5160 processors (3.00 GHz) and 4 GB of RAM, running a 32-bit distribution of GNU/Linux (kernel 2.6.26).

Conclusions

We have presented a model reduction based linear MPC strategy for nonlinear DPS, based on the "equation-free" notion. The key feature of the proposed algorithm is the online identification of the dominant subspace for the dynamics of the nonlinear system at hand, which is exploited for adaptive linearization of this nonlinear model. Only projections of the dependent (state) variables onto this subspace are considered. Thus, the corresponding sensitivity matrices involved in the Jacobian-based linearization around the current state are low-dimensional projections of the original matrices onto the dominant subspace. Numerical directional perturbations are used for the calculations, resulting in low computational cost and memory requirements. This procedure is repeated for every sampling time and results in the identification of a linear state-space model, which is considered time invariant for the prediction horizon and can be used in the context of linear MPC. The control problem is reformulated for every sampling time in the sense that the model of the process changes. Following the receding horizon approach, only the first of the control moves computed from the optimization problem is implemented.

This approach is based on the successive linearization method, which is a NMPC technique that indirectly compensates for the nonlinearity of the problem. A convex (quadratic) optimization is solved per time interval, hence typical issues that arise in conventional NMPC are overcome, such as the existence of multiple optima. In addition to theoretical advantages, this approach exhibits practical ones as well, such as lower computational cost and memory requirements and simplicity. For those reasons, the proposed control

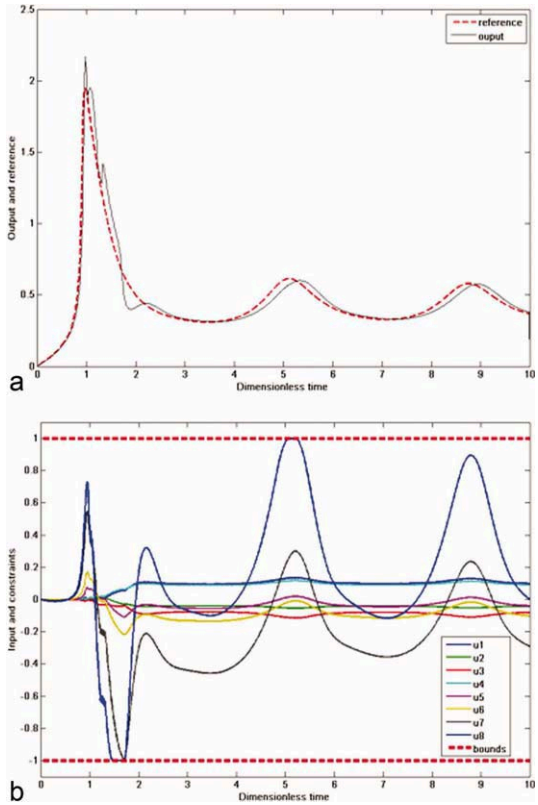

Figure 8. (a) Closed loop output (solid line) and reference (dashed line) for the destabilization case study and (b). the corresponding calculated inputs.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

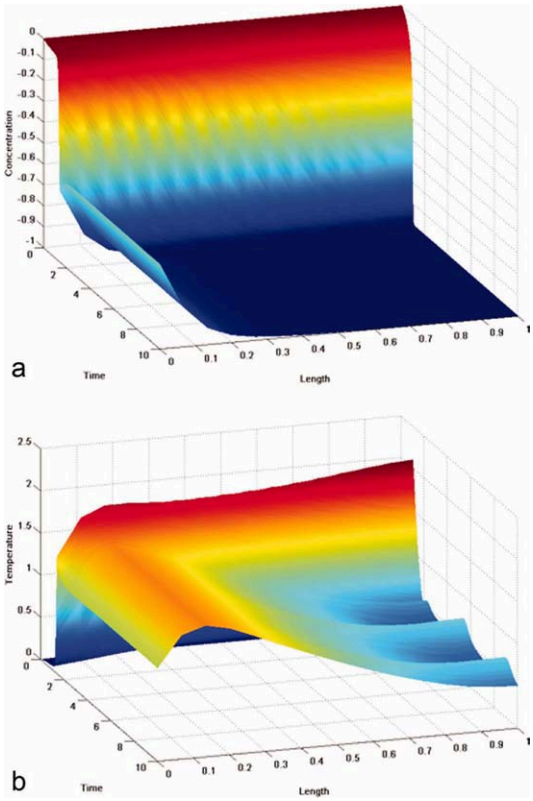

Figure 9. The closed loop dynamic profiles for the dimensionless concentration (a) and dimensionless temperature (b) throughout the reactor for the destabilization case study.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

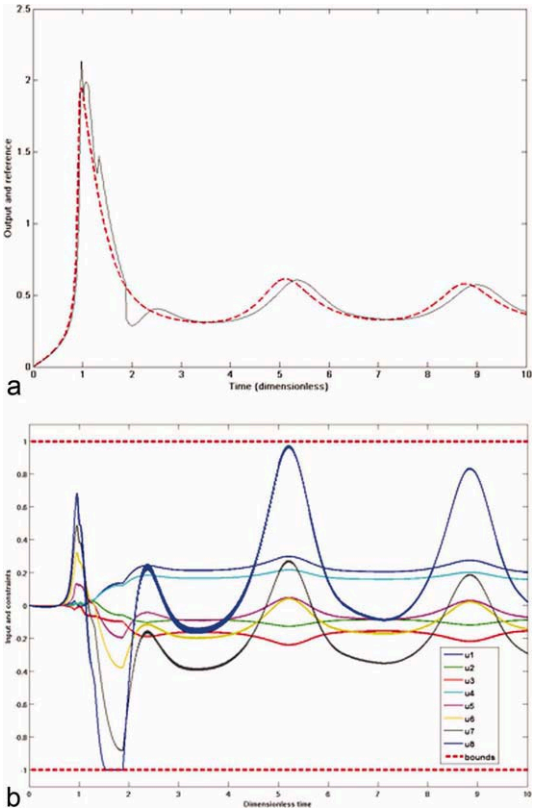

Figure 10. (a) The closed loop (solid line) and reference (dashed line) output for the destabilization case study using Algorithm 3. (b) Corresponding computed cooling profiles.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

method is appropriate for large-scale systems. Furthermore, it does not presuppose providing the equation of the system in closed form (only that the closed form conceptually exists). Only a dynamic simulator for the system at hand is required, which is handled in an input/output fashion. This renders the proposed scheme particularly appropriate for multiscale systems (where the equations for the microscopic simulators do not exist in closed form), and black box codes (commercial or legacy in-house developed), where access to the model is limited in general.

Acknowledgments

This work was supported by the EU programmes CONNECT (COOP-2006-31638) and CAFE (KBBE-2008-212754).

Literature Cited

1. Biegler LT, Wachter A. SQP SAND strategies that link to existing modeling systems. *Large-Scale PDE-Constrained Optim*. 2003: 199–217.
2. Kevrekidis IG, Gear CW, Hyman JM, Kevrekidid PG, Runborg O, Theodoropoulos C. Equation-free, coarse-grained multiscale computation: enabling microscopic simulators to perform system-level analysis. *Commun Math Sci*. 2003;1:715–762.
3. Theodoropoulos C, Qian YH, Kevrekidis IG. "Coarse" stability and bifurcation analysis using time-steppers: a reaction-diffusion example. *Proc Natl Acad Sci USA*. 2000;97:9840–9843.
4. Biegler LT, Ghattas O, Heinkenschloss M, van Bloemen Waanders B. Large-scale PDE-constrained optimization: an introduction. *Large-Scale PDE-Constrained Optim*. 2003:3–13.
5. Bonis I, Theodoropoulos C. A model reduction-based optimisation framework for large-scale simulators using iterative solvers. *Comput Aided Chem Eng*. 2008;25:545–550.
6. Luna-Ortiz E, Theodoropoulos C. An input/output model reduction-based optimization scheme for large-scale systems. *Multiscale Model Simul*. 2005;4:691–708.
7. Theodoropoulos C, Luna-Ortiz E. A reduced input/output dynamic optimisation method for macroscopic and microscopic systems. *Model Reduct Coarse-Grain Appr Multiscale Phenomena*. 2006:535–560.
8. Li HX, Qi C. Modeling of distributed parameter systems for applications—a synthesized review from time-space separation. *J Process Control*. 2010;20:891–901.
9. Theodoropoulos C. *Optimisation and linear control of large scale nonlinear systems: a review and a suite of model reduction-based techniques*. In: Gorban AN, Roose D, editors. *Coping with Complexity: Model Reduction and Data Analysis*, Vol. 75. Berlin: Springer, 2011:37–61.
10. Christofides PD. *Nonlinear and Robust Control of PDE Systems: Methods and Applications to Transport-Reaction Processes*. Boston: Birkhauser, 2001.
11. Dubljevic S, El-Farra NH, Mhaskar P, Christofides P. Predictive control of parabolic PDEs with state and control constraints. *Int J Robust Nonlinear Control*. 2006;16:749–772.

12. Ito K, Kunisch K. Reduced-order optimal control based on approximate inertial manifolds for nonlinear dynamical systems. *SIAM J Numer Anal*. 2008;46:2867–2891.
13. Christofides PD, Daoutidis P. Finite-dimensional control of parabolic PDE systems using approximate inertial manifolds. *J Math Anal Appl*. 1997;216:398–420.
14. El-Farra NH, Armaou A, Christofides PD. Analysis and control of parabolic PDE systems with input constraints. *Automatica*. 2003;39:715–725.
15. Ito K, Ravindran SS. A reduced-order method for simulation and control of fluid flows. *J Comput Phys*. 1998;143:403–425.
16. Armaou A, Christofides PD. Dynamic optimization of dissipative PDE systems using nonlinear order reduction. *Chem Eng Sci*. 2002;57:5083–5114.
17. Shvartsman SY, Kevrekidis IG. Nonlinear model reduction for control of distributed systems: a computer-assisted study. *AIChE J*. 1998;44:1579–1595.
18. Shvartsman SY, Theodoropoulos C, Rico-Martinez R, Kevrekidis IG, Titi ES, Mountziaris TJ. Order reduction for nonlinear dynamic models of distributed reacting systems. *J Process Control*. 2000;10:177–184.
19. Armaou A, Kevrekidis IG, Theodoropoulos C. Equation-free gap-tooth-based controller design for distributed complex/multiscale processes. *Comput Chem Eng*. 2005;29:731–740.
20. Armaou A, Siettos CI, Kevrekidis IG. Time-steppers and "coarse"-control of distributed microscopic processes. *Int J Robust Nonlinear Control*. 2004;14:89–111.
21. Siettos CI, Kevrekidis IG, Kazantzis N. An equation-free approach to nonlinear control: coarse feedback linearization with pole-placement. *Int J Bifurcation Chaos*. 2006;16:2029–2041.
22. Biegler L. Efficient nonlinear programming algorithms for chemical process control and operations. *Syst Model Optim*. 2009:21–35.
23. Zavala VM, Laird CD, Biegler LT. Fast implementations and rigorous models: can both be accommodated in NMPC? *Int J Robust Nonlinear Control*. 2008;18:800–815.
24. Dufour P, Touré Y, Blanc D, Laurent P. On nonlinear distributed parameter model predictive control strategy: on-line calculation time reduction and application to an experimental drying process. *Comput Chem Eng*. 2003;27:1533–1542.
25. Aggelogiannaki E, Sarimveis H. Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models. *Comput Chem Eng*. 2008;32:1225–1237.
26. Panos C, Kouramas KI, Georgiadis MC, Pistikopoulos EN. Dynamic optimization and robust explicit model predictive control of hydrogen storage tank. *Comput Chem Eng*. 2010; 34: 1341–1347.
27. Pistikopoulos EN. Perspectives in multiparametric programming and explicit model predictive control. *AIChE J*. 2009;55:1918–1925.
28. Henson MA. Nonlinear model predictive control: current status and future directions. *Comput Chem Eng*. 1998;23:187–202.
29. Camacho E, Bordons C. Nonlinear model predictive control: An introductory review. *Assess Future Direct Nonlinear Model Predictive Control*. 2007:1–16.
30. Mazinan AH, Sadati N. An intelligent multiple models based predictive control scheme with its application to industrial tubular heat exchanger system. *Appl Intel*. 2009:1–14.
31. Bequette BW. Multiple-model adaptive predictive control of mean arterial pressure and cardiac output. *IEEE Trans Biomed Eng*. 1992;39:765–778.
32. Li WC, Biegler LT. Multistep, Newton-type control strategies for constrained, nonlinear processes. *Chem Eng Res Design*. 1989;67:562–577.
33. Cannon M, Ng D, Kouvaritakis B. Successive linearization NMPC for a class of stochastic nonlinear systems. *Nonlinear Model Predictive Control*. 2009:249–262.
34. Lee JH, Ricker NL. Extended Kalman filter based nonlinear model predictive control. *Ind Eng Chem Res*. 1994;33:1530–1541.
35. Camacho EF, Bordons C. *Model Predictive Control*. London: Springer Verlag, 2004.
36. Garcia CE, Prett DM, Morari M. Model predictive control: theory and practice—a survey. *Automatica*. 1989;25:335–348.
37. Mayne DQ, Rawlings JB, Rao CV, Scokaert PO. Constrained model predictive control: stability and optimality. *Automatica*. 2000; 36:789–814.
38. Wang L. *Model Predictive Control System Design and Implementation using MATLAB*. London: Springer Verlag, 2009.
39. Wills AG, Heath WP. Application of barrier function based model predictive control to an edible oil refining process. *J Process Control*. 2005;15:183–200.
40. Heij C, Ran ACM, van Schagen F. *Introduction to Mathematical Systems Theory. Linear Systems, Identification and Control*. Basel: Birkhauser Verlag, 2007.
41. Datta BN. *Numerical Methods for Linear Control Systems*. California: Elsevier, 2004.
42. Franklin GF, Workman ML, Powell D. *Digital Control of Dynamic Systems*. Boston: Addison-Wesley Longman Publishing, 1997.
43. Alonso AA, Ydstie EB. Process systems, passivity and the second law of thermodynamics. *Comput Chem Eng*. 1996;20:S1119–S1124.
44. Siettos CI, Armaou A, Makeev AG, Kevrekidis IG. Microscopic/stochastic timesteppers and coarse control: a kinetic Monte Carlo example. *AIChE J*. 2003;49:1922–1926.
45. Siettos CI, Maroudas D, Kevrekidis IG. Coarse bifurcation diagrams via microscopic simulators: a state-feedback control-based approach. *Int J Bifurcation Chaos*. 2003;14:207–220.
46. Shroff GM, Keller HB. Stabilization of unstable procedures: the recursive projection method. *SIAM J Num Anal*. 1993;30:1099–1120.
47. Hadjinicolaou M, Goussis DA. Asymptotic solution of stiff PDEs with the CSP method: the reaction diffusion equation. *SIAM J Sci Comput*. 1998;20:781–810.
48. Bemporad A, Morari M, Dua V, Pistikopoulos EN. The explicit linear quadratic regulator for constrained systems. *Automatica*. 2002;38:3–20.
49. Scokaert POM, Rawlings JB. Feasibility issues in linear model predictive control. *AIChE J*. 1999;45:1649–1659.
50. Golub GH, Van Loan CF. *Matrix Computations*. Baltimore: Johns Hopkins University Press, 1996.
51. Jensen KF, Ray WH. The bifurcation behavior of tubular reactors. *Chem Eng Sci*. 1982;37:199–222.
52. Alonso AA, Frouzakis CE, Kevrekidis IG. Optimal sensor placement for state reconstruction of distributed process systems. *AIChE J*. 2004;50:1438–1452.

Manuscript received Oct. 19, 2010, and revision received Mar. 11, 2011.