Preprints, 10th IFAC International Symposium on
Advanced Control of Chemical Processes
Shenyang, Liaoning, China, July 25-27, 2018

System Reconfiguration and Fault-Tolerant for Distributed Model Predictive Control Using Parameterized Network Topology

Guannan Xiao *. Fei Liu *

**Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Institute of Automation, Jiangnan University, Wuxi, 214122, China （TEL: 86-510-85326295-808 ; e-mail: summerxiao94@outlook.com, fliu@jiangnan.edu.cn）*

Abstract: A parameterized network topology based distributed model predictive control (DMPC) framework is proposed in this work, it is mainly applied in system reconfiguration and sensor fault-tolerant control. The Lyapunov stability condition for DMPC with a parameterized network topology is derived. Regarding to system reconfiguration, the parameterized network topology is served as the explicit reconfiguration model. Furthermore, for fault-tolerant control with sensor bias, the parameterized network topology is used to compensate the sensor fault, and a residual generator is designed by states of predictor and consider a time varying threshold for fault detection. The proposed approach is able to handle the system reconfiguration and fault-tolerant control without backup controllers or controllers redesign, and there is no need of the information of fault because of the using of predictor.

Keywords: model predictive control, senor faults, distributed fault-tolerant control, reconfigurable control, cooperative control

1. INTRODUCTION

Model predictive control (MPC) is widely used in process control, because its ability to explicitly deal with input and output constraints while doing optimization online (Mayne et al. (2000)). Facing with the larger-scale systems and stronger interconnections in modern industry, traditional centralized MPC cannot guarantee good control performance for there are large quantities of coupling variables and constraints. Additionally, if large-scale systems are controlled by centralized MPC, the calculation load will increase dramatically (Tippett et al. (2015)). Distributed model predictive control is developed to decentralize one centralized controller into several local controllers based on the decentralized subsystems. In recent years, DMPC has attracted more and more attentions (Negenborn et al. (2014)). According to optimization objects, DMPC is catalogued into two classes: cooperative DMPC (Liu et al. (2009); Jokic and Mircea (2009); Liu et al. (2014); Conte et al. (2016)) and non-cooperative DMPC (Negenborn et al. (2014)). This paper concerns the cooperative DMPC which optimizes a global cost function utilizing distributed optimization algorithms, moreover, a parameterized topology network for distributed control is considered for both processes and controllers.

Compared with the single process, a number of more challenging problems arise in large-scale processes, especially for distributed control structure. The increasing need of personalized and intelligent production have led to interests on the field of system reconfiguration and distributed flexible control, allowing the configurations among some subsystems can be changed, for example, components of raw materials or material concentration are various. Subjecting to it, some results have been published, from the view of holonic manufacturing and supply chain management, Nirav gave the reconfigurable distributed control framework in continuous process control (Chokshi et al. (2008)). Using the idea of hierarchical architectures, Tony Wauters implemented the reconfiguration production in food industry (Wauters et al. (2012)). Tippett and Bao introduced changeable network topologies into dissipativity based distributed model predictive control to realize reconfigurable controller (Tippett et al. (2015)). Research on system reconfiguration flexible control is still limited, more efforts are needed.

System reconfiguration is also used in fault tolerant control (FTC), while different the changeable processes, the reconfiguration for FTC is often for controller constructions but not process constructions. General control laws in DMPC are under the condition that there is no fault in actuators or sensors, but actually in many cases, there may be some faults or bias on them, which will affect stability and performances of system quickly through the interactions among subsystems. So fault-tolerant control has been studied to adapt the change in the system, existing FTC strategies can be concluded into two kinds: the passive FTC and activate FTC. Passive FTC considers all the possible faults as disturbances or uncertainties in controller designing so that it has strong conservative. While the active FTC first need to detect the faults and design controller to compensate the residuals, relying on fault diagnosis and control reconfiguration. Most existing distributed FTC studies are concentrated on handling actuator faults. David et al. (2010) dealt with the actuator

faults in distributed model predictive control by designing Lyapunov based back up controllers to reconfigure the distributed controller. Alexey similarly used different reconfigurable alternative controllers, and choose the suitable one by comparing the performances for compensation (Zakharov et al. (2015)). While for the case that there is a sensor bias in distributed control systems, little work has been reported before (He et al. (2017)). Some previous work considered sensor bias including in estimation and system control (He et al. (2017); Xu et al. (2017); Han et al. (2017); Manimozhi et al. (2017); Boem et al. (2017); Yin and Liu (2017)). In Xu et al. (2017), an active FTC with robust estimation and MPC was proposed for bounded sensor faults. Assuming that the sensor bias models are known, a Riccati matrix based network FTC controller was derived (Han et al. (2017)). Backup controllers are set in He et al. (2017) with amplitude of sensor bias is estimated by least-square method. There is no sensor fault tolerant control in distributed model predictive control system of my knowledge, and because of the prediction property of MPC, it can provide extra information for fault tolerant.

This work considers a parameterized network topology, and both controllers for distributed system reconfiguration and sensor FTC can be concluded as problems that this method is applicable. Replace fixed values in topology with variable parameters, then the parameterized network topology forms. The main contributions of this paper are: (1) derive the Lyapunov based distributed model predictive controller with parameterized network topology; (2) The controller with parameterized network is applied for system reconfiguration without redesign; (3) For FTC strategy, a time varying threshold is set according to actual situation to improve sensitivity of fault detection and sensor bias is compensated by parametrized topology.

The remaining part of this paper is organized as follows: The problem is formulated in Section 2, the reconfiguration DMPC is stated in Section 3, the FTC algorithm is introduced in Section 4 and Section 5 shows numerical examples, Section 6 concludes this paper.

2. LYAPUNOV BASED DISTRIBUTED MODEL PREDICTIVE CONTROL

A large-scale system containing M linear subsystems is stated as:

$$\begin{cases} x_i(k+1) = \sum_{j\in\mathcal{N}_i} A_{ij}x_j(k) + B_i u_i(k) \\ y_i(k) = \sum_{j\in\mathcal{N}_i} C_{ij}x_j(k) \end{cases} \quad i = 1,\ldots,M \quad (1)$$

where $\mathcal{N}_i$ represents the neighbor subsystems of i th subsystem which have effect on i th subsystem, $x_i \in \mathbb{R}^{n_i}$ is the state of each subsystem, $u_i \in \mathbb{R}^{p_i}$ is the control variable of ith system, and $y_i \in \mathbb{R}^{q_i}$ is the output of each subsystem effected by interconnected states of neighbor systems. The matrices A_{ij}, B_i, and C_{ij} are constant matrices.

For model predictive control, the model for prediction can be derived as:

$$\begin{cases} \tilde{x}_i = \sum_{j\in\mathcal{N}_i} \tilde{A}_{ij}x_j(k) + \tilde{B}_i\tilde{u}_i(k) \\ \tilde{y}_i = \sum_{j\in\mathcal{N}_i} \tilde{C}_{ij}x_j(k) \end{cases} \quad (2)$$

where $\tilde{x}_i = \{x_i(k+1), x_i(k+2), \cdots, x_i(k+N)\}$ represents the predicted states according to system model in optimal domain N. Similarly, $\tilde{y}_i = \{y_i(k), y_i(k+1), \cdots, y_i(k+N-1)\}$. $\tilde{A}_{ij}$, $\tilde{B}_i$, and $\tilde{C}_i$ are written as:

$$\tilde{A}_{ij} = \begin{bmatrix} A_{ij} & 0 & \cdots & \cdots & 0 \\ A_{ij}^2B_i & A_{ij} & 0 & \ddots & \vdots \\ A_{ij}^3B_i & A_{ij}^2B_i & \cdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ A_{ij}^NB_i & A_{ij}^{N-1}B_i & \cdots & \cdots & A_{ij} \end{bmatrix}$$

$$\tilde{B}_i = \begin{bmatrix} B_i & \cdots & \cdots & B_i \end{bmatrix}^T$$

$$\tilde{C}_i = \begin{bmatrix} C_i & C_{ij}A_{ij} & \cdots & C_{ij}A_{ij}^N \end{bmatrix}^T$$

The cost function of cooperative DMPC is defined by (3).

$$J = \min_u \sum_{j\in M} J_j$$
$$J_j = \sum_{i=1}^{N} \begin{bmatrix} (x_j(k+i) - x_s)^T P (x_j(k+i) - x_s) \\ + u_j(k+i-1)^T Q u_j(k+i-1) \end{bmatrix} \quad (3)$$

where J is the global cost function of cooperative distributed model predictive control problem constructed by every subsystem cost function, to solve a cooperative optimization, the distributed optimization need to be utilized.

Based on the Lyapunov theorem, distributed model predictive control stability can be analyzed. For each subsystem i, if there exists a Lyapunov function $V_i(x_i(k))$ and $\mathcal{K}$ class functions β_{1i}, β_{2i}, β_{3i} then the system is stable (Jokic and Mircea (2009)):

$$\begin{aligned} &\beta_{1i}(x_i) \le V(x_i) \le \beta_{2i}(x_i) \\ &V(x_i(k+1)) - V(x_i(k)) \le -\beta_{3i}(x_i(k)) \end{aligned} \quad (4)$$

3. SYSTEM RECONFIGURATION OF DISTRIBUTED MODEL PREDICTIVE CONTROL

System reconfiguration in this work is considered as system interconnection structure change including the connection rate and interconnection states (interconnected or un-interconnected). The interconnection of subsystems can be represented as a network topology, as connection form in (1), a state-to-state topology is derived. For the control construction described in Fig.1, Large-scale process and distributed controllers both have their respective network topology, they may be the same or different.

For example, assume a large system contains three subsystems and each process has three measurable states, and process is modeled as (1), then this kind of the network

topology of process can be written as (5), and element 1 means having connections with two states while 0 is opposite.

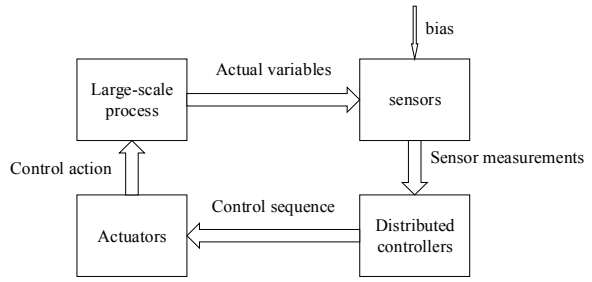


Fig. 1. Large-scale system structure.

The topology matric in (5) shows that except for the internal states of each subsystem, one subsystem is infected by other subsystems. This topology (5) describes the relationship between each state in different subsystems. But if all described by 0 and 1 as general, it can only provide the information whether the two states are interconnected or not, the relationship about connection intension cannot be represented.

$$T_p = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{5}$$

System reconfiguration network topology is explicitly described by parameters, i.e. substitute the 0-1 binary variables in network with variable parameters expressing the different connecting intensions. This description provides a way to show system change explicitly in topology. For parameterized topology, (5) can be rewritten as:

$$T_p\left(\alpha(t)\right) = \begin{bmatrix} \mathbf{1}_{n1\times n1} & \boldsymbol{\alpha}_{12}(t) & \boldsymbol{\alpha}_{13}(t) \\ \boldsymbol{\alpha}_{21}(t) & \mathbf{1}_{n2\times n2} & \boldsymbol{\alpha}_{23}(t) \\ \boldsymbol{\alpha}_{31}(t) & \boldsymbol{\alpha}_{32}(t) & \mathbf{1}_{n3\times n3} \end{bmatrix} \tag{6}$$

where $\boldsymbol{\alpha}_{ij}(t)$ is the topology matric of subsystem i and subsystem j constructed by parameters $\alpha_{kl}(t)$, the information is from j to i. The subscript index k and l of $\alpha(t)$ are sequence number of states in each sub-process.

With the parameterized topology (6), the system (1) can be rewritten as:

$$\begin{cases} X(k+1) = T_P A_P X(k) + B_P U(k) \\ Y(k) = T_P C_P X(k) \end{cases} \tag{7}$$

where $X(k) = \left[x_1(k) \quad x_2(k) \quad \cdots \quad x_m(k)\right]^T \in \mathbb{R}^{n_1+n_2+\ldots+n_m}$,

$Y(k) = \left[y_1(k) \quad y_2(k) \quad \cdots \quad y_m(k)\right]^T \in \mathbb{R}^{q_1+q_2+\ldots+q_m}$

$U(k) = \left[u_1(k) \quad u_2(k) \quad \cdots \quad u_m(k)\right]^T \in \mathbb{R}^{p_1+p_2+\ldots+p_m}$

Divide (7) into M subparts, each subsystem is:

$$\begin{cases} x_i(k+1) = T_{Pi}\left(\alpha(k)\right) A_{Pi} x(k) + B_{Pi} U \\ y_i(k) = T_{Pi}\left(\alpha(k)\right) C_{Pi} x(k) \end{cases} \tag{8}$$

where T_{Pi}, A_{Pi}, B_{Pi} and C_{Pi} are the sub-matrices associating with i th subsystem in T_P, A_P, B_P and C_P.

Furthermore, to simplify computation, remove the zero items in matric T_P so that consider the subsystems having interconnections of states only, i.e. the neighbor systems. Then the system model with parameterized topology can be reorganized as (9).

$$\begin{cases} x_i(k+1) = \mathbf{1}_{n_i\times n_i} A_{ii} x_i(k) + \sum_{j\in\mathcal{M}_i} \boldsymbol{\alpha}_{ij}(k) A_{ij} x_j(k) + B_i u_i(k) \\ y_i(k) = \sum_{j\in\mathcal{M}_i} \boldsymbol{\alpha}_{ij}(k) C_{ij} x_j(k) + C_{ii}(k) x_i(k) \end{cases} \tag{9}$$

where $\boldsymbol{\alpha}_{ij}$ is parameter matric without zero items. (11) is the same form as (1). Assume that the network topology of controller is same as process, the model controller used to determine the control input is also as (9), so the stable DMPC controller can be derived in **Theorem 1**.

Theorem 1: For each subsystem i with time-varying parameterized topology, if there exists a Lyapunov function $V_i\left(x_i(k)\right)$ and $\mathcal{K}$ class functions β_{1i}, β_{2i}, β_{3i} satisfying the following conditions for stability:

$$\begin{aligned} &\beta_{1i}\left(x_i\right) \le \boldsymbol{\alpha}_i(k) V\left(x_i\right) \le \beta_{2i}\left(x_i\right) \\ &\boldsymbol{\alpha}_i(k+1) V\left(x_i(k+1)\right) - \boldsymbol{\alpha}_i(k) V\left(x_i(k)\right) \le -\beta_{3i}\left(x_i(k)\right) \end{aligned} \tag{10}$$

Remark 1: For reconfigurable control with (9), both the process and controller topology need to be changed. The most important here is the controller design is parameterized by network time varying parameters which system reconfiguration is obtained without a new controller design.

4. FAULT-TOLERANT CONTROL WITH RECONFIGURABLE SYSTEM

In this section, distributed sensor fault tolerant control strategy using method introduced in Section 3 is proposed to maintain stability and better performance than faulty system.

4.1 Fault detection

In order to detect the faults in sensors, the residual should be generated first. For the residual generation algorithm, because of the predictive property of MPC, the states and outputs from predictor are introduced to be compared with sensor measurements.

The predictor for (1) is showed as (11), where $\hat{x}_i(k)$, $\hat{u}_i(k)$, $\hat{y}_i(k)$ are the predictive variables, $\hat{u}_i(k)$ is the control inputs calculated from DMPC controller and $\hat{x}_i(k+1)$, $\hat{u}_i(k)$ can both be saved according to the controller. So

different from the general MPC, the amplitude of $\hat{x}_i(k+1)$ should be saved, which may increase the storage space needed of system.

$$\begin{cases} \hat{x}_i(k+1) = \sum_{j\in\mathcal{N}_i} A_{ij}x_j(k) + B_i\hat{u}_i(k) \\ \hat{y}_i(k) = \sum_{j\in\mathcal{N}_i} C_{ij}x_j(k) \end{cases} \tag{11}$$

If the sensor measurement is $x_i'(k+1)$ at $k+1$ sampling time under the control input $\hat{u}_i(k)$, the residual of states x_i can be represented as following:

$$r_i(k+1) = \left| x_i'(k+1) - \hat{x}_i(k+1|k) \right| \tag{12}$$

If $r_i(k+1)$ is equal to zero, there is no sensor bias, while if $r_i(k+1)$ is not zero, there may be sensor bias. But consider that there may be some measurement noises leading $x_i'(k+1)$ can't match $\hat{x}_i(k+1)$ correctly, a threshold r_{ih} is needed. The threshold means the minimal residual that can be accepted for fault detection. According to the simulation, if a fixed threshold is applied to the fault detection, a part of sensor bias can't be identified if the amplitude of states is near to zero. So in order to improve the sensitivity of fault detection, a time varying threshold $r_{ih}(k)$ is defined.

$$r_{ih}(k) = \mu\hat{x}_i(k) \tag{13}$$

where μ is a rate that satisfies $0<\mu<1$ associated with the amplitude of noise or it can be determined by the actual situations.

The principle of fault detection is:
If $r_i(k) < r_{ih}(k)$, there is no fault;
If $r_i(k) \geq r_{ih}(k)$, there is fault detected, translate to the FTC.

In this way, some influence of noises on the process can be rejected when detect faults and sensitivity of fault detection can be improved.

4.2 Fault compensation with system reconfiguration

Sensor bias is the fault on the amplitude of measurements that sensor send to controller, but there is no bias on actual process, only sensor measurements correction are put in controller design model network topology while the process network is unchanged.

Controller is designed using the model (1) and state measurements, substitute the sensor measurements $x_i'(k+1)$, the actual model used for controller is represented in (14).

The reconfigurable FTC controller as (9) is defined in (15).

The parameter γ_j is attached to sensor measurements $x_j'(k)$, it is the correction parameter for $x_j'(k)$ when there

$$\begin{cases} \hat{x}_i(k+1) = \sum_{j\in\mathcal{N}_i} A_{ij}x_j'(k) + B_i\hat{u}_i(k) \\ \hat{y}_i(k) = \sum_{j\in\mathcal{N}_i} C_{ij}x_j'(k) \end{cases} \tag{14}$$

$$\begin{cases} \hat{x}_i(k+1) = \sum_{j\in\mathcal{N}_i} \gamma_j A_{ij}'x_j(k) + \hat{B}_i\hat{u}_i \\ \hat{y}_i(k) = \sum_{j\in\mathcal{N}_i} \gamma_j C_{ij}'x_j(k) \end{cases} \tag{15}$$

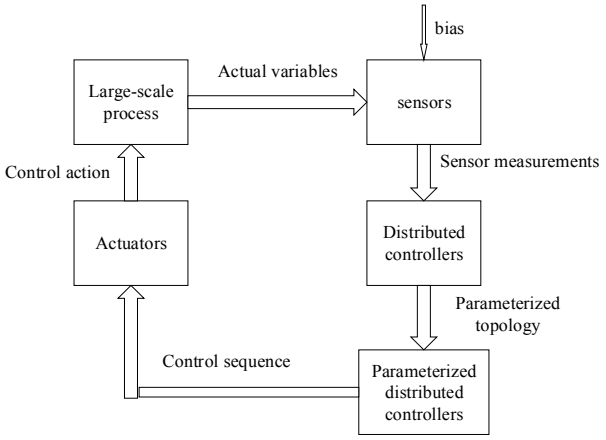


Fig. 2. Parameterized DMPC structure.

is bias on it, substituting $x_j'(k)$ to $\gamma_j x_j'(k)$ will near to the true value. If a fault is detected on the state $x_j'(k)$, then let predictive state $\hat{x}_j(k)$ as the true value, so the correction parameter γ_j is calculated as:

$$\gamma_j = 1 - \frac{x_j'(k) - \hat{x}_j(k)}{x_j'(k)} \tag{16}$$

The fault-tolerant control algorithm containing fault detection and fault compensation is described in Algorithm 1:

Algorithm 1: **Fault detection and fault compensation**

Initialization of parameters

Step 1: At time k, receive the sensor measurement $x_j'(k)$

Step 2: Calculate the residual $r_j(k)$

Fault detection:

$r_i(k) < r_{ih}(k)$: no sensor fault, $\gamma_j(k) = 1$

$r_i(k) \geq r_{ih}(k)$: sensor fault detected, update as (16)

Step 3: Design FTC controller as (15)

Step 4: Save $\hat{x}_j(k)$, $k = k+1$, go back to step1

5. NUMERICAL SIMULATION

Consider a numerical example constructed by two linear subsystems. The two subsystems are connected through the state x_{22} of subsystem 2, except for the subsystem 1, it also influences subsystem 1 as the second state of subsystem1. The setpoints are all set to zero and sampling time is $T = 1s$.

$$A_1 = \begin{bmatrix} -0.0667 & 0 \\ 0 & -0.0667 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0.1171 & 0.1152 \\ 0.1443 & 0.1461 \end{bmatrix},$$

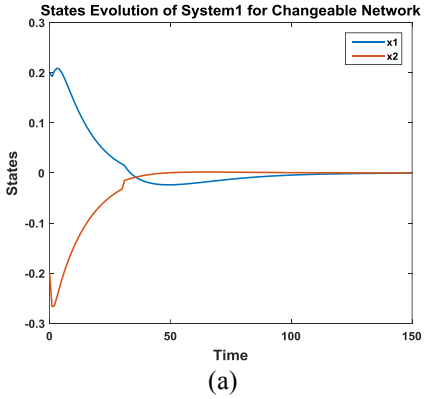
$$C_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} -0.0667 & 0 \\ 0 & -0.0667 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} 0.2108 & 0.0230 \\ 0.2597 & 0.0292 \end{bmatrix}, \quad C_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

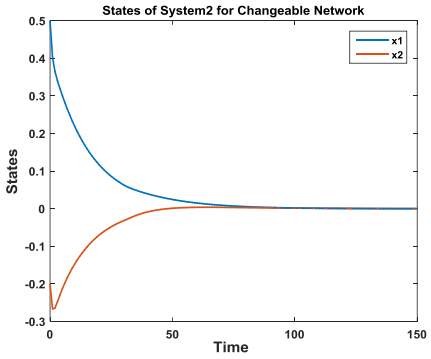
For system reconfiguration, when $k < 30$, the state x_{22} interconnects with the subsystem 1 by whole amplitude, when $k \geq 30$, the connection decreases to 50% of the amplitude of state x_{22}. Fig.3(a)-(b) show the state evolution of subsystem 1 and subsystem 2. From Fig.3a and Fig.3c it can be seen that when sampling instant is 30, the construction of subsystem 1 changes, states and control inputs have an obvious shake, but after that, the system is stable at the setpoint.

For fault-tolerant control, consider that when there is a sensor bias of the sensor of state 1 in subsystem 1. The parameter of threshold $r_{ih}(k)$ is set as $\mu = 0.1$ and with (13) and output measurements, it is time varying. The control process shows in Fig 4. The evolutions of state 1 are compared between FTC and without FTC is showed in Fig 5. It can be seen that states with fault-tolerant controller are driven to setpoint faster than without it. And the residual for actual system and system without sensor bias is showed in Fig 6. Control with sensor bias has an obvious residual compared with corrected control with sensor bias, which shows the FTC controller has some effect on sensor bias restrain.

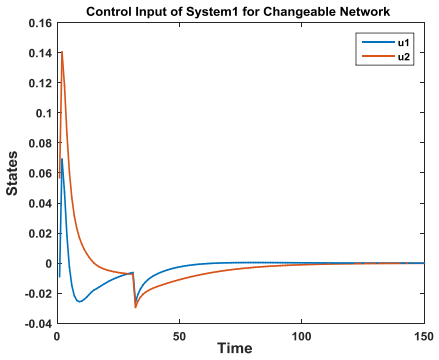
Remark 2: Though the two subsystems are less for large-scale system, they can still reflect the characteristics of the algorithm here. The reconfiguration situation in process control can be considered as a change of feed concentration



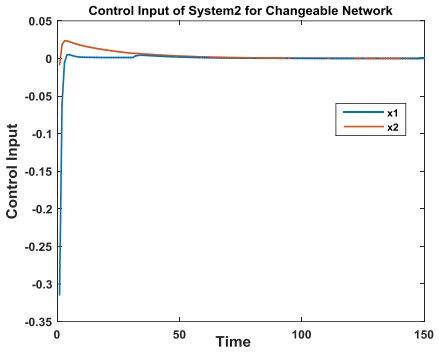
(a)



(b)

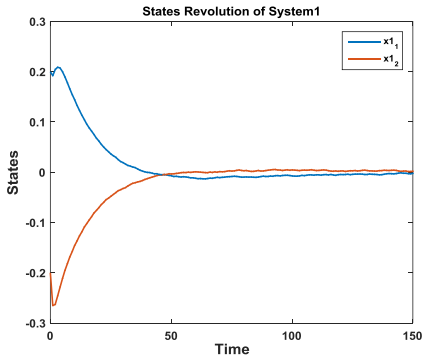


(c)

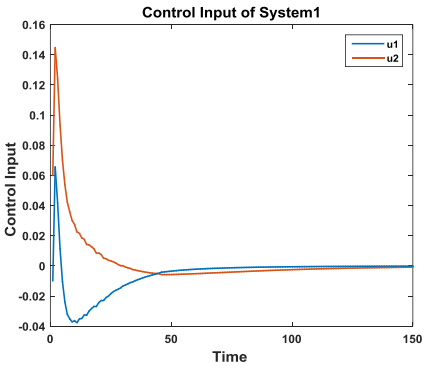


(d)

Fig. 3. State (a-b) and control (c-d) evolution evolution for reconfigurable system.



(a)



(b)

Fig. 4. State (a) and control (b) evolution of subsystem 1 for fault-tolerant control.

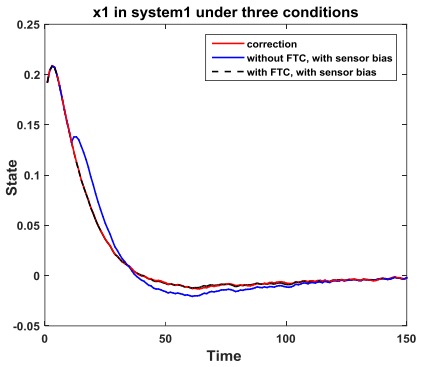


Fig. 5. State comparisons for three situations: correct system, sensor fault with FTC, sensor fault without FTC.

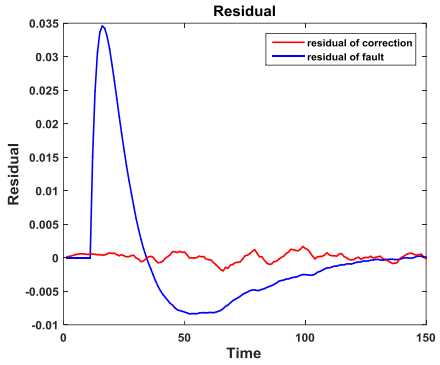


Fig. 6. Residual comparison between FTC system and without FTC system.

or component of interconnected process. For fault-tolerant control, the parameterized topology is like a multiplier of the controller to compensate for the bias.

6. CONCLUSION

A parameterized topology for large-scale processes and controllers network is introduced in this work, and controllers for both system reconfiguration and sensor FTC can be designed with it. For system reconfiguration, this allows distributed controllers adjust parameters online without redesign. For FTC with sensor bias, with compensation brought by parameterized topology, there is no need for backup controllers and redesign. This way has taken full advantages of properties for distributed control and large-scale system network.

ACKNOWLEDGEMENT

This work is supported by The National Natural Science Foundation of China (NSFC 61773183).

REFERENCES

Boem, F., Ferrari, R.M., Keliris, C., Parisini, T. and Polycarpou, M.M. (2017). A distributed networked approach for fault detection of large-scale systems. *IEEE Transactions on Automatic Control*, *62*(1), pp.18-33.

Chilin, D., Liu, J., de la Peña, D.M., Christofides, P.D. and Davis, J.F. (2010). Detection, isolation and handling of actuator faults in distributed model predictive control systems. *Journal of Process Control*, *20*(9), pp.1059-1075.

Chokshi, N. and McFarlane, D. (2008). A distributed architecture for reconfigurable control of continuous process operations. *Journal of Intelligent Manufacturing*, *19*(2), pp.215-232.

Conte, C., Jones, C.N., Morari, M. and Zeilinger, M.N. (2016). Distributed synthesis and stability of cooperative distributed model predictive control for linear systems. *Automatica*, *69*, pp.117-125.

Han, S.Y., Chen, Y.H. and Tang, G.Y. (2017). Sensor Fault and Delay Tolerant Control for Networked Control Systems Subject to External Disturbances. *Sensors*, *17*(4), p.700.

He, X., Wang, Z., Liu, Y., Qin, L. and Zhou, D. (2017). Fault-Tolerant Control for an Internet-Based Three-Tank System: Accommodation to Sensor Bias Faults. *IEEE Transactions on Industrial Electronics*, *64*(3), pp.2266-2275.

Jokic, A. and Lazar, M., 2009, June. On decentralized stabilization of discrete-time nonlinear systems. In *American Control Conference, 2009. ACC'09.* (pp. 5777-5782). IEEE..

Liu, J., Muñoz de la Peña, D. and Christofides, P.D. (2009). Distributed model predictive control of nonlinear process systems. *AIChE Journal*, *55*(5), pp.1171-1184.

Liu, S. and Liu, J. (2014). Distributed lyapunov‐based model predictive control with neighbor‐to‐neighbor communication. *AIChE Journal*, *60*(12), pp.4124-4133.

Manimozhi, M. and Saravanakumar, R. (2017). Sensor and actuator bias estimation using multi model approach. *Computers & Electrical Engineering*, *57*, pp.118-133.

Mayne, D.Q., Rawlings, J.B., Rao, C.V. and Scokaert, P.O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, *36*(6), pp.789-814.

Negenborn, R.R. and Maestre, J.M. (2014). Distributed model predictive control: An overview and roadmap of future research opportunities. *IEEE Control Systems*, *34*(4), pp.87-97.

Tippett, M.J. and Bao, J. (2015). Reconfigurable distributed model predictive control. *Chemical Engineering Science*, *136*, pp.2-19.

Wauters, T., Verbeeck, K., Verstraete, P., Berghe, G.V. and De Causmaecker, P. (2012). Real-world production scheduling for the food industry: An integrated approach. *Engineering Applications of Artificial Intelligence*, *25*(2), pp.222-228.

Xu, F., Olaru, S., Puig, V., Ocampo‐Martinez, C. and Niculescu, S.I. (2017). Sensor‐fault tolerance using robust MPC with set‐based state estimation and active fault isolation. *International Journal of Robust and Nonlinear Control*, *27*(8), pp.1260-1283.

Yin, X. and Liu, J. (2017). Distributed Output‐Feedback Fault Detection and Isolation of Cascade Process Networks. *AIChE Journal*.

Zakharov, A., Zattoni, E., Yu, M. and Jämsä-Jounela, S.L. (2015). A performance optimization algorithm for controller reconfiguration in fault tolerant distributed model predictive control. *Journal of Process Control*, *34*, pp.56-69.