Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 461513, 10 pages
http://dx.doi.org/10.1155/2015/461513

Research Article

A Simulation Perspective: Error Analysis in the Distributed Simulation of Continuous System

Yao-fei Ma and Xiao Song

School of Automation Science and Electrical Engineering, Beihang University, Beijing, China

Correspondence should be addressed to Yao-fei Ma; mayaofeibuaa@163.com

Received 8 January 2015; Revised 8 April 2015; Accepted 15 April 2015

Academic Editor: Yun-Bo Zhao

To construct a corresponding distributed system from a continuous system, the most convenient way is to partition the system into parts according to its topology and deploy the parts on separated nodes directly. However, system error will be introduced during this process because the computing pattern is changed from the sequential to the parallel. In this paper, the mathematical expression of the introduced error is studied. A theorem is proposed to prove that a distributed system preserving the stability property of the continuous system can be found if the system error is limited to be small enough. Then, the compositions of the system error are analyzed one by one and the complete expression is deduced, where the advancing step T in distributed environment is one of the key factors associated. At last, the general steps to determine the step T are given. The significance of this study lies in the fact that the maximum T can be calculated without exceeding the expected error threshold, and a larger T can reduce the simulation cost effectively without causing too much performance degradation compared to the original continuous system.

1. Introduction

Engineering problems involve dealing with physical systems. Since most physical laws are described by differential equations, simulation in engineering is in fact related to numerical resolution of differential equations. This is called continuous system simulation and even the components (i.e., the subsystems or submodels) of it are generally described in time discretization manner [1].

The large engineering system has sustained requirement on distributed simulation for two reasons. First, the continuous growth on system scale and complexity results in intense demand on computing capacity, which can be mitigated in distributed environment by scattering the computing load to networked nodes. Second, the system itself is sometimes geographically fragmented; thus, the distributed structure is needed to be consistent with its topology. Except those constructed in distributed manner from scratch, there are still many classic continuous systems that were constructed in the nondistributed way, in that the system was designed and tested without considering the possible scenario of distributed simulation. The demand to transform such system into the distributed one emerges.

The classic continuous system is characterized by computing its components in pipeline, where a computing sequence is set up and the components are computed one by one within a step; the sequence is determined by the component's input/output characteristics and the data dependence. By contrast, the computation on each component in the distributed environment will start simultaneously and advance in a parallel manner; the updating data is exchanged periodically through the network for synchronization.

The difference between two computing patterns can be formulized as follows. Suppose a system containing n components: $M = \{m_1, m_2, \ldots, m_n\}$. A single component can be represented as $m_i = \{x_i, y_i, f_i\}$, where x_i is the input, y_i is the output, and f_i is the model function. We have $y_i = f_i(x_i)$. Assuming a computing sequence $Q = \{q_1, \ldots, q_i, \ldots, q_n\}$ has been determined in the nondistributed environment, the pipeline computing within one step can be described as

$$x_{q_1}^{k\times h} = I^{k\times h},$$

$$y_{q_1}^{k\times h} = f_{q_1}\left(x_{q_1}^{k\times h}\right),$$

$$x_{q_i}^{k\times h} = y_{q_{i-1}}^{k\times h},$$

$$y_{q_i}^{k\times h} = f_{q_i}\left(x_{q_i}^{k\times h}\right) = f_{q_i}\left(y_{q_{i-1}}^{k\times h}\right),\qquad i = 2,\dots,n, \tag{1}$$

where $k = 0,1,2,\dots,h$ is the step size and the subscript q_i refers to the index in Q. In the pipeline computing, the first component is assumed to get the input from a signal source I, and the others get inputs following the data dependence among themselves.

By contrast, the one-step computing in distributed environment can be described as

$$\begin{aligned} x_1^{k\times T} &= I^{k\times T}, \\ y_1^{k\times T} &= f_i\left(x_1^{k\times T}\right), \\ x_i^{k\times T} &= y_{i-1}^{(k-1)\times T}, \\ y_i^{k\times T} &= f_i\left(x_i^{k\times T}\right) = f_i\left(y_{i-1}^{(k-1)\times T}\right), \\ & \qquad i = 2,\dots,n, \end{aligned} \tag{2}$$

where T is the advancing step of the distributed system. It should be noted that the computing sequence is not held any more in this case.

There are two differences comparing these two computing processes. First, the computing is synchronous in the nondistributed environment. For each component, the input is updated to time step k firstly, and then the output at k is computed (the data dependence among components is assumed to be consistent with the computing sequence). On the other hand, the computation is parallel in the distributed environment and each component starts simultaneously. The input of a component will still hold the value of step $(k - 1)$ when the output at k needs to be computed. Second, the advancing steps of two environments, h and T, may be different. In the nondistributed environment, h is determined by the equation solver; either fixed or variable step is employed. However, the determination of T needs to consider the bandwidth of the underlying network, signal frequency, human perception limitation, and so forth. It is often the case where the whole system advances with a fixed T and $T \geq h$.

A formal approach capable of partitioning a system into parallel parts and properly handling the above two issues is the key to build a distributed simulation from a nondistributed one. This problem, denoted as the *Partitioning Problem*, was early studied in the simulation of complex mechanical systems [2, 3] and then became a research focus in the field of discrete event simulation [2, 4, 5] and the decentralized, large-scale control system [6, 7]. However, the proposed approaches all involved extra efforts to reformulize or reengineer the simulated system and did not consider the possible difference on h and T either. These drawbacks set up barriers to apply these approaches broadly.

In [4], anovel partitioning approach is proposed using the system's topology. This approach is easy to perform since most classic simulations have block-diagram structures, but it is "lossy" because the disorder of the component's input/output data cannot be avoided, thus leading to the error on state trajectory of the resulting distributed system. To reduce the error, a portioning rule was proposed to eliminate the possible cumulative delays caused by improper partitioning, which contribute a lot to the overall error.

This paper will study the mathematical expression of this error. The correlation between the error and the advancing step T in distributed environment is revealed. Then, a maximum T can be calculated according to the error threshold specified by the system engineer. The significance to find the maximum T lies in the fact that a larger T will improve the parallelism of the partitioned system by reducing the synchronization frequency between nodes, without causing too much performance degradation at the same time.

2. Literature Review

The *Partitioning Problem* was early studied in the coupled problem [2, 5] of complex mechanical system simulation, where the fluids, thermal, control, and structure subsystems interacted with each other and formed multi-physical-field system [8]. The partitioning is applied to decompose such system into partitions with physical or computational considerations. The resulting system could separately advance in time over each partition, thus gained high simulation efficiency. However, these studies were focused on the finite element analysis, not for general purpose simulation.

The *Partitioning Problem* is also critical to build the decentralized, large-scale control system [6, 7]. A graph-based approach was employed in that the system states were connected as vertexes and the coupling strength (measured by weight factor) between any two of them was evaluated. Small weight connections were more likely to form the partitioning edge by which the system was split into relative independent parts. Another methodwas to use the delay differential equations [9, 10] to describe the dynamic of the distributed system. Networked control scheme [11] was constructed to obtain the convergence and stability in control of the system. However, this approach focuses on the influence of signal delay associated with network latency, rather than the errors caused by the change of computing pattern. In our opinion, such errors will still exist even though the underlying network is perfect and has no delay.

In the Modeling and Simulation (M&S) domain, DEVS (Discrete Event System Specification) theory casts light into the solving of *Partitioning Problem*. The basic idea was to transform a continuous system into the DEVS form by quantizing the system states. A DEVS system is comprised of a set of connected components (called "atomic model" or "coupled model"). The output of each component will hold unchanged until the values of the states it maintained exceed some predefined quantized level. As a result, the time driving continuous system is transformed into an event driving system; thus, it can be decoupled and partitioned easily [12–14]. The transformed system is suitable for asynchronous, distributed environment in nature; however, the "illegitimacy" phenomenon, where the states may transit for unlimited times within a limited period, often causes the simulation to fail to converge correctly.

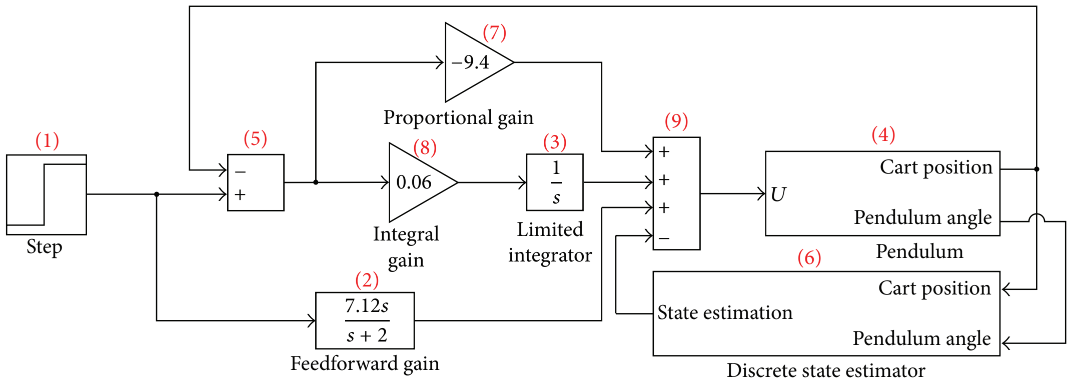


FIGURE 1: An inverted pendulum system. The states are "cart position" and "pendulum angle." A computing sequence is maintained (indicated by red numbers) in the nondistributed environment.

As an extension of DEVS, the QSS approach [15–17] was proposed to resolve the "illegitimacy" problem by using a special "hysteretic quantization" method, where the output of each component (or model, as called in DEVS system) will not transit to the next quantization level during its descending period until the change has exceeded certain threshold. QSS provides a general approach for the Partitioning Problem since each QSS model interacts with discrete, states updating events. However, the QSS model needs to be particularly designed to allow states transiting between quantization levels. The changes of states are triggered by unpredicted threshold crossing events in QSS. In other words, the advancing "step" of QSS system is unpredicted and time-varying. This characteristic makes QSS unsuitable for applications such as the Hardware-In-the-Loop (HIL) or Man-In-the-Loop (MIL) simulation, where the system has to advance with some fixed step T to align the hardware's working frequency or to take care of the needs of human perception.

The approaches mentioned above all involve extra efforts to reformulize or reengineer the simulated system. For example, in the graph-based approach, the detailed analysis of the coupling strength between system states requests the engineer to have full knowledge of system dynamics. The QSS approach also needs the system components to be modified to follow the DEVS formulation. Additionally, the possible difference between h and T was not considered either. These drawbacks set up barriers to apply these approaches broadly.

By contrast, the partitioning approach proposed in [4] took the advantage of the system's structure characteristics; no extra work is needed except for decomposing the system according to the data transferring route. However, the incurred error needs to be further studied considering the possible correlation with T.

The content of this paper is organized as follows: in Section 3, the previous work is briefly reviewed. In Section 4, the mathematical expression of the errors is given with theorems. In Section 5, a series of steps are concluded to determine the maximum T.

3. Problem Description

Normally, a continuous system can be represented by a block diagram as Figure 1 shows [4]. This assembling approach based on basic components is commonly seen in the construction of complex system.

The components within this system can be classified into two categories: the Direct-Feed-Through (DFT) component and the Non-Direct-Feed-Through (NDFT) component. The output of DFT component is directly associated with its input; that is, the output at time t is determined by the input at the same time. On the other hand, the output of NDFT component is only determined by its current state and has nothing to do with the current input. It implies that, when computing a DFT component, the components it depends on should be computed firstly to maintain the correct data dependence. The NDFT component has no such requirement since its current output does not rely on the current input.

With parallel computing in the distributed environment, more deductions can be deduced from the foregoing. First, if part of the system deployed on a separate node formed a NDFT component (both DFT and NDFT component can have recursive structure), its output would be delayed for T when updating to other nodes. Second, if two or more DFT components were cascaded, they should not be divided into different nodes; otherwise, the output delay would accumulate from the first DFT component. For example, if the components (5), (7), and (9) in Figure 1 were deployed to different nodes, the output of component (9) will be $3T$ delayed compared to the original system, component (7) is $2T$ delayed, and component (5) is T delayed.

These outcomes have been observed in the experiments of [4], and a partitioning rule was proposed to reduce the accumulated delays. The rule is simple: the distributed DFT component should be avoided; each partitioned part should be ensured to form a NDFT component. The system performance was improved by applying this rule, as shown in Figure 2.

Although the partitioning rule made the distributed system more close to the original continuous system, there

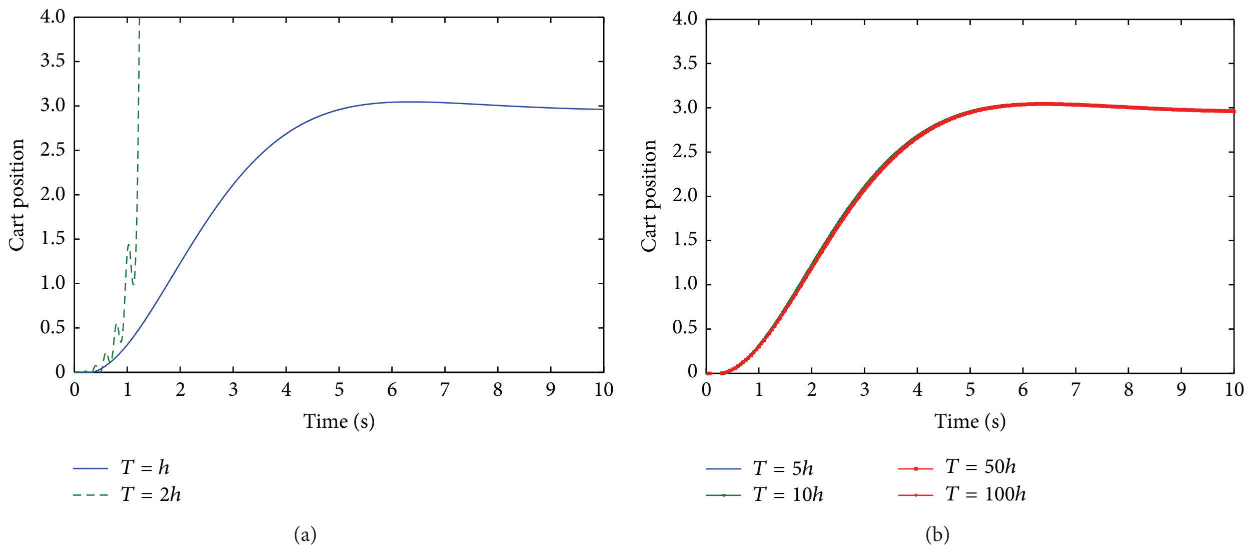


FIGURE 2: (a) The comparison of the trajectories ("cart position"). (a) The distributed system became unstable when $T = 2h$ without applying the partitioning rule. (b) By applying this rule, the distributed system was stable even when $T = 100h$.

is still one step (T) delay among each node. This delay cannot be eliminated completely because of the parallelism nature of distributed environment, leading to the system error (denoted as ΔE). To understand the influence of T to ΔE, the mathematical expression of ΔE needs to be formulized.

To simplify the analysis of ΔE, an abstract control system containing two components is employed here: one component is the *Controller* and the other is the *Plant*. This system is assumed to be *Asymptotically Stable*. Denoting the original continuous system as CS and the distributed system as DS, the controller and plant can be described as differential equations in CS [15]:

$$\text{Controller (CS)}: \begin{cases} \dot{x_c}(t) = f_c(x_c(t), u_c(t)) \\ y_c(t) = g_c(x_c(t), u_c(t)) \\ u_c = y_p, \end{cases} \tag{3a}$$

$$\text{Plant (CS)}: \begin{cases} \dot{x_p}(t) = f_p(x_p(t), u_p(t)) \\ y_p(t) = g_p(x_p(t)) \\ u_p = y_c, \end{cases} \tag{3b}$$

where $x_c(t)$ and $x_p(t)$ are system states, $y_c(t)$ and $y_p(t)$ are outputs, $u_c(t)$ and $u_p(t)$ are inputs, $f_c(*)$ and $f_p(*)$ are state functions, and $g_c(*)$ and $g_p(*)$ are output functions. In DS, the controller and the plant are described as

$$\text{Controller (DS)}: \begin{cases} \dot{x_c}(t) = f_c(x_c(t), (u_c)_d(t)) \\ y_c(t) = g_c(x_c(t), (u_c)_d(t)) \\ (u_c)_d(t) = (y_p)_d(t-T), \end{cases} \tag{4a}$$

$$\text{Plant (DS)}: \begin{cases} \dot{x_p}(t) = f_p(x_p(t), (u_d)_q(t)) \\ y_p = g_p(x_d(t)) \\ (u_p)_d(t) = (y_c)_d(t-T), \end{cases} \tag{4b}$$

where $(*)_d$ is the discretization operator. This operator indicates the operand updates its value to the external world following the advancing step T, just like being discretized.

Perturbation Analysis is used here to analyze the composition of ΔE. Perturbation analysis treats ΔE as the outcome of disturbances to CS such as the output delay and discretization. The difference between CS and DS can be determined without formulizing the differential equations of both systems and then comparing the eigenvalues. First of all, the DS system is expressed in the form of perturbation equations:

$$\begin{aligned} \dot{x_p} &= f_p(x_p, (g_c)_d(x_c(t-T), g_p(x_p(t-T)))) \\ &= f_p(x_p, g_c(x_c + \Delta x_c, g_p(x_p + \Delta x_p) + \Delta y_p) \\ &\quad + \Delta y_{cd}), \end{aligned} \tag{5a}$$

$$\begin{aligned} \dot{x_c} &= f_c(x_c, (g_p)_d(x_p(t-T))) \\ &= f_c(x_c, g_p(x_p + \Delta x_p) + \Delta y_{pd}), \end{aligned} \tag{5b}$$

where

(i) $\Delta x_c = x_c(t-T) - x_c$, the error of controller state caused by time delay;

(ii) $\Delta x_p = x_p(t-T) - x_p$, the error of plant state caused by time delay;

(iii) $\Delta y_p = g_p(x_p(t-T)) - g_p(x_p)$, the error of plant output caused by time delay;

(iv) $\Delta y_{cd} = (g_c)_d - g_c$, the error of controller output caused by discretization;

(v) $\Delta y_{pd} = (g_p)_d - g_p$, the plant output error caused by discretization.

The system error is represented as

$$\Delta E = \left\| \left(\Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd} \right) \right\|. \quad (6)$$

To find out the detailed expression of ΔE, a preparation theorem is proposed as follows.

4. The Mathematical Analysis

4.1. A Preparation Theorem. Based on the assumption of *Asymptotically Stable*, the functions of $f_p(*)$, $g_p(*)$, $f_c(*)$, and $g_c(*)$ in (3a), (3b), (4a), and (4b) are further assumed to be *Lipschitz Continuous* over a region D, where D is a nonsaturation region defined by x_p and x_c:

$$D = \left\{ x_p, x_c \mid x_p \in D_{x_p}, x_c \in D_{x_c}, g_p\left(x_p\right) \in D_{y_p}, g_c\left(x_c, g_p\left(x_p\right)\right) \in D_{y_c} \right\}. \quad (7)$$

For convenience, the time symbol t is omitted unless necessary. D_{x_p} and D_{x_c} are nonsaturation regions for x_p and x_c, respectively, that is, the state spaces of the controller and the plant. D_{y_p} and D_{y_c} are nonsaturation regions for y_p and y_c, respectively, that is, the output spaces of the controller and the plant. With this assumption, a preparation theorem [18] is given.

Theorem 1 (preparation theorem). *For an asymptotically stable CS as shown in (3a) and (3b), if (a) the functions $f(*)$ and $g(*)$ are continuously differentiable and (b) a Lyapunov function V is defined over an open region D containing the original point (assumed to be the equilibrium point), then a DS can be obtained from CS, in that all start positions lying in an arbitrary interior region (denoted as D_1) of D are attracted to another arbitrary interior region (D_2) of D_1 in finite time. D_1 and D_2 are defined by sections of V crossed by two level surfaces.*

This theorem is critical because it guarantees that DS can be derived from a CS given that CS is stable. The assumption that the original point is the equilibrium is easy to be satisfied by translation transformation of the state variables.

Proof. Define an auxiliary function:

$$\alpha\left(x, \Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd}\right) = \frac{\partial V}{\partial x_p} \cdot f_p\left(x_p, g_c\left(x_c + \Delta x_c, g_p\left(x_p + \Delta x_p\right) + \Delta y_p\right) + \Delta y_{cd}\right) + \frac{\partial V}{\partial x_c} \cdot f_c\left(x_c, g_p\left(x_p + \Delta x_p\right) + \Delta y_{pd}\right). \quad (8)$$

We have

$$\alpha\left(x, 0, 0, 0, 0, 0\right) = \dot{v}(x), \quad (9)$$

where $x = (x_p, x_c)$ and $v(x)$ is the Lyapunov function of CS.

Define the second auxiliary function $\beta(*)$ over a region D_3:

$$\beta\left(\Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd}\right) = \sup_{x \in D_3} \left(\alpha\left(x, \Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd}\right)\right), \quad (10)$$

where $D_3 = D_2 - D_1$. $\beta(*)$ is continuous since $\alpha(*)$ is continuous. Then, we have

$$\alpha\left(x, \Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd}\right) < \beta\left(\Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd}\right) \quad (11a)$$

and then

$$\beta\left(0, 0, 0, 0, 0\right) = \sup_{x \in D_3} \left(\alpha\left(x, 0, 0, 0, 0, 0\right)\right) = \sup_{x \in D_3} \left(\dot{v}(x)\right). \quad (11b)$$

A positive real number s can be found since $\dot{v}(x)$ is the negative definition to satisfy:

$$\dot{v}(x) < -s. \quad (12a)$$

Combining (11b), then

$$\beta\left(0, 0, 0, 0, 0\right) < -s. \quad (12b)$$

As a result, a region D_4 can always be found in the vicinity of the origin of $\beta(*)$:

$$D_4 = \left\{ \left(\Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd}\right) \mid \left\| \left(\Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd}\right) \right\| = \Delta E < \rho \right\}, \quad (13)$$

where ρ is a positive real number defining the radius of D_4. Another positive real number s_1 ($0 < s_1 < s$) can be found to satisfy (14) over D_4:

$$\beta\left(\Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd}\right) < -s_1. \quad (14)$$

Considering inequality (11a), we have

$$\alpha\left(x, \Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd}\right) < -s_1, \quad x \in D_3. \quad (15)$$

Consider (9), and we have

$$\dot{v}(x) < -s_1, \quad x \in D_3. \quad (16)$$

To integrate both sides of (16) in $[0, t]$,

$$V\left(x(t)\right) - V\left(x_0\right) < -s_1 * t, \quad (17a)$$

$$V\left(x(t)\right) < V\left(x_0\right) - s_1 * t, \quad (17b)$$

where $x_0 \in D_3$ is the start point of state trajectory. Inequation (17b) means the state trajectory in DS is strictly constrained by a diminishing function in D_3. Considering the fact that $\dot{v(x)} < 0$, we have

$$V(x_0) < V_{D_1}, \tag{18a}$$

$$V(x(t)) < V_{D_1} - s_1 * t, \tag{18b}$$

where V_{D_1} is the value of $V(x)$ where the state trajectory crosses the level surface that defines region D_1. Inequations (18a) and (18b) mean that the state trajectory of DS will stay inside D_1. Considering $\dot{v(x)} < 0$, we also have

$$V(x_0) > V_{D_2}, \tag{19}$$

where V_{D_2} is the value of $V(x)$ where the state trajectory crosses the level surface that defines region D_2. Then, the solution trajectory of DS will go from the start point to D_2 within finite time:

$$\Delta t < \frac{V(x_0) - V_{D_2}}{s_1}. \tag{20}$$

Inequation (20) means a DS can be found, whose trajectory will converge to the equilibrium point of CS eventually within finite time, given CS being asymptotically stable and the system error being constrained as (13) shows. □

4.2. The Mathematical Expression of System Error. The mathematical expression of ΔE is studied by analyzing each component of it.

4.2.1. Δx_c. Δx_c is the variation of the controller's state value within time interval T. According to the definition in (5a) and (5b),

$$\begin{aligned}\Delta x_c &= x_c(t-T) - x_c \\ &= \int_{t-T}^{t} f_c(x_c(\tau), u_c(\tau)) \cdot d\tau,\end{aligned} \tag{21a}$$

$$\|\Delta x_c\| = \left\| \int_{t-T}^{t} f_c(x_c(\tau), u_c(\tau)) \cdot d\tau \right\|. \tag{21b}$$

Integrating both sides of (21b),

$$\begin{aligned}\|\Delta x_c\| &= \left\| \int_{t-T}^{t} f_c(x_c(\tau), u_c(\tau)) \cdot d\tau \right\| \\ &\le \int_{t-T}^{t} \|f_c(x_c(\tau), u_c(\tau))\| \cdot d\tau.\end{aligned} \tag{22}$$

$f_c(*)$ is bounded in $[t - T, t]$ since it is continuously differentiable (one of the conditions in Theorem 1). Then, we have

$$\|\Delta x_c\| \le \int_{t-T}^{t} M_{f_c} \cdot d\tau = M_{f_c} \cdot T, \tag{23}$$

where M_{f_c} is the upper bound of $f_c(*)$ within $[t - T, t]$.

4.2.2. Δx_p. Δx_p is the variation of the plant's state value within time interval T. Following a similar approach in the analysis of Δx_c, we have

$$\|\Delta x_p\| \le M_{f_p} \cdot T, \tag{24}$$

where M_{f_p} is the upper bound of $f_p(*)$ within $[t - T, t]$.

4.2.3. Δy_p. Δy_p is the variation of the plant's output within time interval T. According to (5a) and (5b), we have

$$\begin{aligned}\Delta y_p &= g_p(x_p(t-T)) - g_p(x_p) \\ &= g_p(x_p + \Delta x_p) - g_p(x_p),\end{aligned} \tag{25a}$$

$$\|\Delta y_p\| = \|g_p(x_p + \Delta x_p) - g_p(x_p)\|. \tag{25b}$$

Considering $g_p(*)$ is *Lipschitz Continuous* (Theorem 1), we have

$$\begin{aligned}\|\Delta y_p\| &= \|g_p(x_p + \Delta x_p) - g_p(x_p)\| \\ &\le M_{g_p} \cdot \|(x_p + \Delta x_p) - (x_p)\| \\ &= M_{g_p} \cdot \|\Delta x_p\|.\end{aligned} \tag{26a}$$

Considering (24), we have

$$\|\Delta y_p\| \le M_{g_p} \cdot M_{f_p} \cdot T, \tag{26b}$$

where M_{g_p} is the *Lipschitz Const* of $g_p(*)$.

4.2.4. Δy_{pd}. Δy_{pd} is caused by discretization on the plant's output, where the discrete interval is T. Δy_{pd} is bounded by the changing rate of the output, as shown in Figure 3.

According to the definitions in (5a) and (5b), we have

$$\|\Delta y_{pd}\| = \|(g_p)_d - g_p\| \le \Delta y_p. \tag{27}$$

Considering (26b),

$$\|\Delta y_{pd}\| \le M_{g_p} \cdot M_{f_p} \cdot T. \tag{28}$$

4.2.5. Δy_{cd}. Similarly, we have

$$\|\Delta y_{cd}\| \le M_{g_c} \cdot M_{f_c} \cdot T. \tag{29}$$

Combining (6), (23), (24), (26b), (28), and (29),

$$\begin{aligned}\Delta E &= \|(\Delta x_c, \Delta x_p, \Delta y_p, \Delta y_{cd}, \Delta y_{pd})\| \\ &\le \sqrt{M_{f_c}^2 (1 + M_{g_c}^2) + M_{f_p}^2 \cdot (1 + 2M_{g_p}^2)} \cdot T,\end{aligned} \tag{30}$$

where $\sqrt{M_{f_c}^2 (1 + M_{g_c}^2) + M_{f_p}^2 \cdot (1 + 2M_{g_p}^2)} \cdot T$ defines the upper bound of ΔE. Once an expected ΔE is given (denoted as ΔE^*), $T_{\max}$ can be determined:

$$T_{\max} = \frac{\Delta E^*}{\sqrt{M_{f_c}^2 (1 + M_{g_c}^2) + M_{f_p}^2 \cdot (1 + 2M_{g_p}^2)}} \tag{31}$$

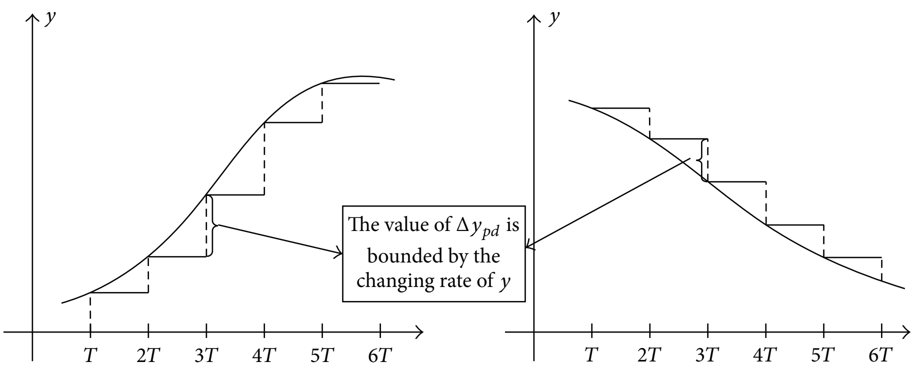

y
y
The value of Δy_{pd} is bounded by the changing rate of y
T
2T
3T
4T
5T
6T

FIGURE 3: Δy_{pd} is bounded by the changing rate of the output value.

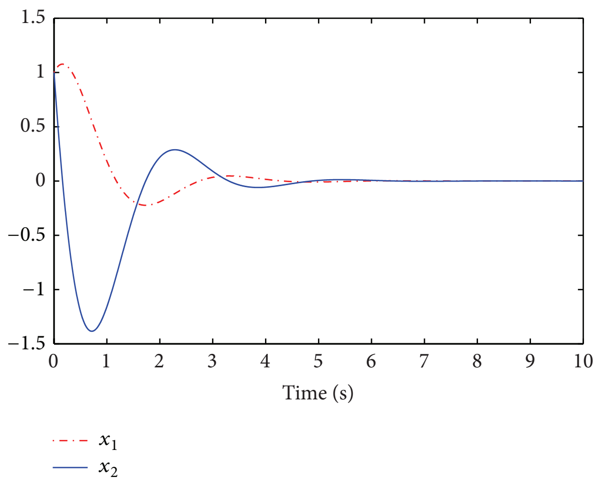


FIGURE 4: The trajectories of x_1, x_2, where $k = 5$, $R = 5$, $m = 1$, and the initial value $x_1^{t=0} = x_2^{t=0} = 1$.

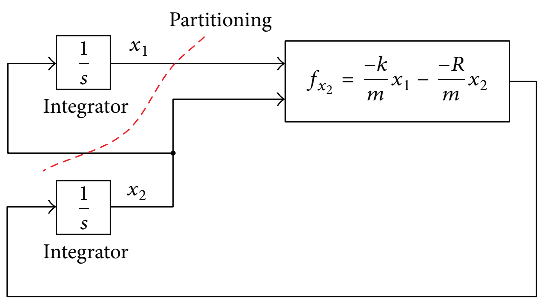


FIGURE 5: This simple system is partitioned into two parts by the dash line to form a distributed system.

for any $T < T_{\max}$, and the overall error will be less than ΔE^*. Equation (31) also indicates the way to partition CS will influence $T_{\max}$: the more the parts partitioned (supposing each part is deployed on a node), the greater the denominator of the right part of (31), thus the smaller that of the allowable $T_{\max}$.

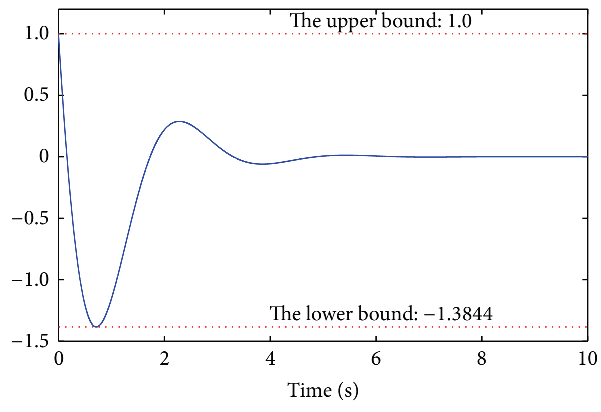


FIGURE 6: The plot of $\dot{x}_1$. $M_{f_{x_1}}$ is determined by the bound of it in this example.

5. Experiment

A distributed simulation is constructed in this section to show the determination of T. The original continuous system is a point mass system with fraction as follows:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= \frac{-k}{m}x_1 - \frac{R}{m}x_2, \end{aligned} \tag{32}$$

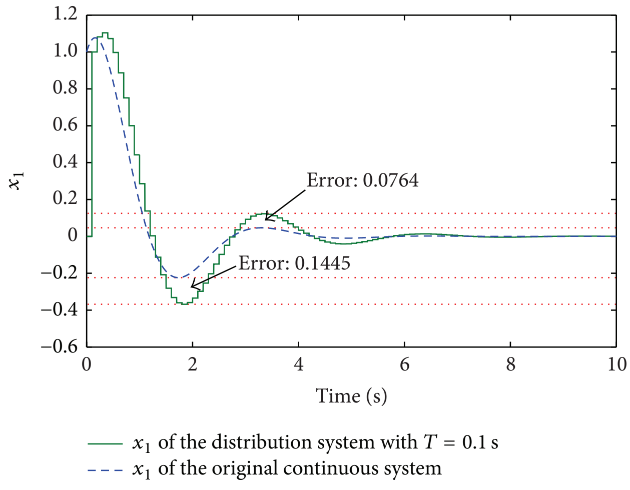
where k is spring const, m is mass, and R is friction coefficient. This system is asymptotically stable; the trajectories of x_1 and x_2 are shown in Figure 4.

A distributed system can be partitioned from the original system, where the structure has the block-diagram representation (see Figure 5).

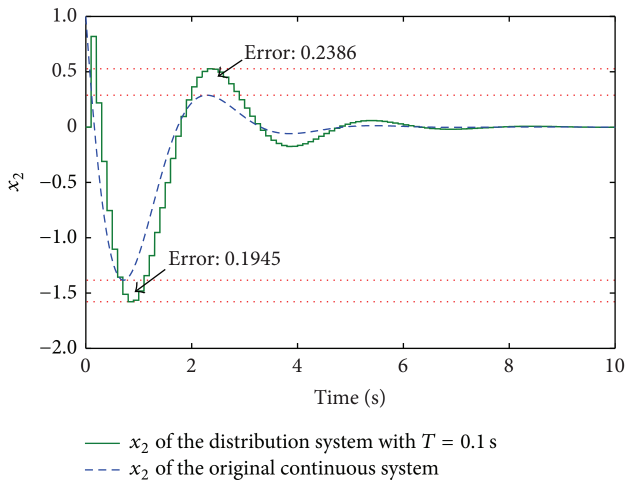
The differential equations of these two parts are

$$\text{Component I: } \begin{cases} \dot{x}_1 = f_{x_1} = x_2 \\ y_1 = g_{x_1} = x_1, \end{cases} \tag{33a}$$

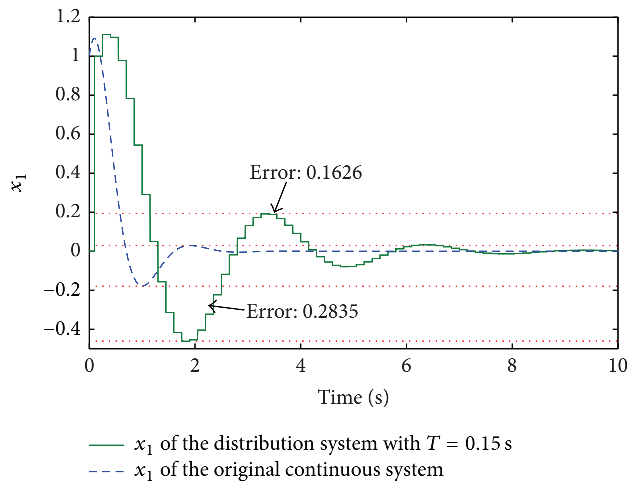
$$\text{Component II: } \begin{cases} \dot{x}_2 = f_{x_2} = \dfrac{-k}{m}x_1 - \dfrac{R}{m}x_2 \\ y_2 = g_{x_2} = x_2, \end{cases} \tag{33b}$$



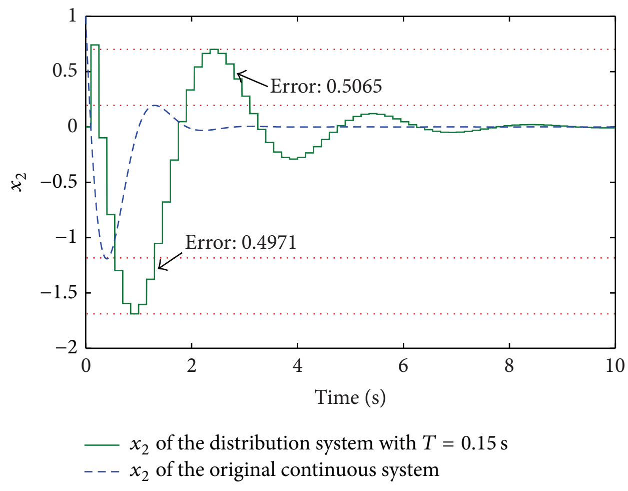
(a)



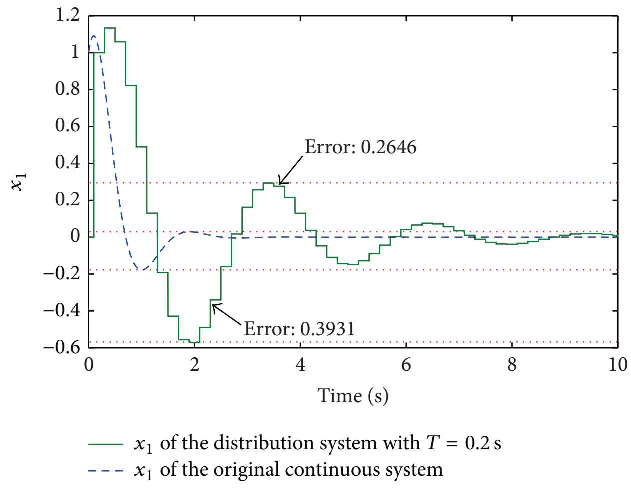
(b)



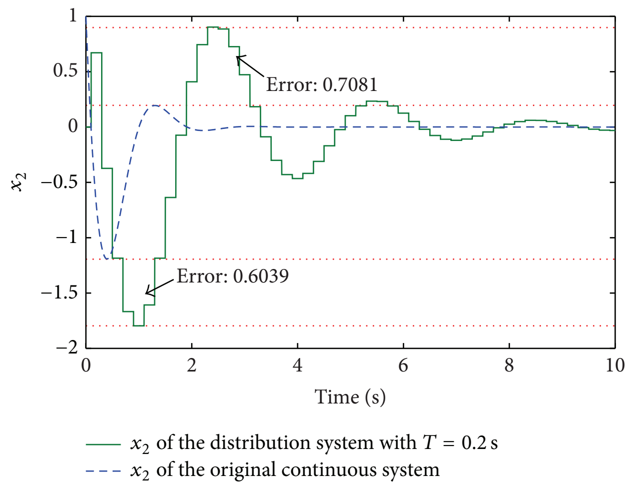
(c)



(d)



(e)



(f)

FIGURE 7: The comparison of x_1 and x_2. In the distributed simulation, T is set to be 0.1 s, 0.15 s, and 0.2 s, respectively; the trajectories of x_1 and x_2 are compared in each case.

where y_1, y_2 are the output of components I and II, respectively. These two parts are deployed on two nodes, forming the simple distributed system.

Equation (31) indicates that the parameters reflecting system dynamic need to be determined firstly, which are as follows:

$M_{f_{x_1}}$: the upper bound of function f_{x_1} within any time interval $[t - T, t]$;

$M_{g_{x_1}}$: the *Lipschitz Const* of function g_{x_1};

$M_{f_{x_2}}$: the upper bound of function f_{x_2} within any time interval $[t - T, t]$;

$M_{g_{x_2}}$: the *Lipschitz Const* of function g_{x_2}.

From the engineering perspective, these parameters can be directly observed from the simulation results of the original system. In this case, $M_{f_{x_1}}$ is determined by the lower bound of $\dot{x_1}$ as Figure 6 shows.

As a result, $M_{f_{x_1}} = -1.3844$. According to the definition of *Lipschitz Const*, $M_{g_{x_1}}$ is limited by the upper bound of $\dot{g_{x_1}}$, that is, the upper bound of $\dot{x_1}$ in this case. Then, we have $M_{g_{x_1}} = -1.3844$. Following similar procedures, we have $M_{f_{x_2}} = -1.7796$, $M_{g_{x_2}} = -1.7796$. According to (31), $T_{\max}$ is estimated as follows:

$$T_{\max} \approx \sqrt{M_{f_{x_2}}^2 \left(1 + M_{g_{x_2}}^2\right) + M_{f_{x_1}}^2 \cdot \left(1 + 2M_{g_{x_1}}^2\right)} \approx 0.211 \cdot \Delta E. \quad (34)$$

The distributed system described in (33a) and (33b) is simulated. The trajectories of the system states are compared to those of the original continuous system, as Figure 7 shows. The threshold value of ΔE is 0.5, and then $T_{\max}$ is estimated to be 0.1055 s according to (34). Three configurations, where $T = 0.1$ s, $T = 0.15$ s, and $T = 0.2$ s, are tested in the form of distributed simulation. As expected, only the case $T = 0.1$ s satisfies that both x_1 and x_2 do not exceed the threshold value of ΔE.

However, the actual error does not exceed the threshold value immediately when T is configured to be greater than 0.1 s, as shown in Figures 7(c) and 7(d). The reason is that $T_{\max}$ is not strictly estimated since many associated parameters use their boundary values.

6. Conclusion

The mathematical expression of ΔE helps us to gain the insight of system error produced in the construction of the distributed system using the partitioning approach. Giving an expected threshold of ΔE, a proper advancing step T of the distributed system can be determined. A larger T, compared to the integral step in the nondistributed system, will reduce the data-exchange frequency between simulation nodes, leading to the reduction of demands on system timing performance and network bandwidth. Then, the simulation cost is saved eventually. In fact, this approach can also play a role in multicore or multi-CPU parallel computing environments.

However, for systems that may become unstable after system partitioning, this approach is not so convenient since a *Lyapunov function* of the continuous system needs to be found firstly. The parameters defining specific regions satisfying (13)–(15) also need to be determined. More work needs to be done to improve this approach's availability.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, vol. 8 of *Springer Series in Computational Mathematics*, Springer, Berlin, Germany, 2nd edition, 1993.

[2] E. Hinton, P. Bettess, and R. W. Lewis, Eds., *Numerical Methods in Coupled Problems*, Pineridge Press, Swansea, UK, 1981.

[3] R. W. Lewis, Ed., *Numerical Methods in Coupled Problems*, John Wiley & Sons, London, UK, 1984.

[4] Y.-f. Ma, X. Song, J.-y. Wang, and Z. Xiao, "A practical infrastructure for real-time simulation across timing domains," *Mathematical Problems in Engineering*, vol. 2015, Article ID 163845, 12 pages, 2015.

[5] R. W. Lewis, Ed., *Numerical Methods in Coupled Problems*, Wiley, London, UK, 1984.

[6] C. Ocampo-Martinez, S. Bovo, and V. Puig, "Partitioning approach oriented to the decentralised predictive control of large-scale systems," *Journal of Process Control*, vol. 21, no. 5, pp. 775–786, 2011.

[7] K. Salahshoor and S. Kamelian, "A new weighted graph-based partitioning algorithm for decentralized nonlinear model predictive control of large-scale systems," *International Journal of Computer Applications*, vol. 40, no. 14, pp. 7–14, 2012.

[8] C. A. Felippa, K. C. Park, and C. Farhat, "Partitioned analysis of coupled mechanical systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 190, no. 24-25, pp. 3247–3270, 2001.

[9] W. Michiels and S.-I. Niculescu, *Stability and Stabilization of Time-Delay Systems: An Eigenvalue-Based Approach*, vol. 12, SIAM, 2007.

[10] J.-B. Hu, G.-P. Lu, S.-B. Zhang, and L.-D. Zhao, "Lyapunov stability theorem about fractional system without and with delay," *Communications in Nonlinear Science and Numerical Simulation*, vol. 20, no. 3, pp. 905–913, 2015.

[11] S. Kamelian and K. Salahshoor, "A novel graph-based partitioning algorithm for large-scale dynamical systems," *International Journal of Systems Science: Principles and Applications of Systems and Integration*, vol. 46, no. 2, pp. 227–245, 2015.

[12] A. Adegoke, H. Togo, and M. K. Traore, "A unifying framework for specifying DEVS parallel and distributed simulation architectures," *Simulation-Transactions of The Society for Modeling and Simulation International*, vol. 89, no. 11, pp. 1293–1309, 2013.

[13] F. Bergero, E. Kofman, and F. Cellier, "A novel parallelization technique for DEVS simulation of continuous and hybrid systems," *SIMULATION*, vol. 89, no. 6, pp. 663–683, 2013.

[14] S. Jafer, Q. Liu, and G. Wainer, “Synchronization methods in parallel and distributed discrete-event simulation,” *Simulation Modelling Practice and Theory*, vol. 30, pp. 54–73, 2013.

[15] E. Kofman, “Quantization-based simulation of differential algebraic equation systems,” *Simulation-Transactions of the Society for Modeling and Simulation Internatinal*, vol. 79, no. 9, pp. 363–376, 2003.

[16] E. Kofman, “Discrete event simulation of hybrid systems,” *SIAM Journal on Scientific Computing*, vol. 25, no. 5, pp. 1771–1797, 2004.

[17] E. Kofman, J. S. Lee, and B. P. Zeigler, “DEVS representation of differential equation systems: review of recent advances,” in *Proceedings of the 13th European Simulation Symposium*, pp. 591–595, October 2001.

[18] E. Kofman, *Discrete event based simulation and control of continuous [Ph.D. thesis]*, Universidad Nacional de Rosario, Santa Fe, Argentina, 2003.