2016 American Control Conference (ACC)
Boston Marriott Copley Place
July 6-8, 2016. Boston, MA, USA

A Distributed Computation Scheme for Real-time Control and Estimation of PDEs

Satadru Dey, Yongqiang Wang, and Beshah Ayalew

***Abstract*—Real-time estimation/control of Partial Differential Equation (PDE) systems, especially for large-scale applications, generally involves high computational burdens. In this paper, we propose a distributed computation scheme, which can leverage available and otherwise idle computing resources to cooperatively solve the high-dimensional controller/estimator implementations for fine-grained management of such PDE systems. Such a real-time distributed computation scheme requires communication among the computing resources which is subject to uncertainties due to imperfections of the communication network. Given this scenario, the proposed approach: 1) includes a modeling framework in the controller/estimator implementation that explicitly addresses network uncertainties, 2) uses a diagonalization-based scheme where the approximated ODE form is transformed into the diagonal form before implementation in order to minimize the communication requirement, and 3) includes a filtering solution to suppress the effect of communication uncertainties. The proposed scheme is illustrated via a real-time state estimation of individual battery cells in vehicle battery packs using a network of vehicular computing units. Simulation results are included to illustrate the effectiveness of the scheme.**

I. Introduction

Distributed Computing Systems (DCS) are becoming very useful in several engineering applications. The concept of DCS can be boiled down to cooperative execution of a computational problem by a set of computing resources. However, the execution process generally has some limitations such as weak links with the central coordination, communication constraints (communication bandwidth, communication delays, unreliable communication link) etc [1]. The advantage of DCSs lies in the reduction of computation time for large-scale computations given that the communication imperfections are taken care of [2]. In this paper, we proposed a computation approach that can be deployed over DCS for real-time estimation and control of Partial Differential Equation (PDE) systems.

Several spatially distributed and large-scale systems can be modeled by PDEs such as: fluid flow systems [3], manufacturing processes [4], large structures [5], etc. Controller and estimator implementation for PDEs in real-time hardware is often done in the Ordinary Differential Equation (ODE) form [6]. Due to the inherent infinite-dimensional nature of the PDEs, generally the dimension of the ODE system that closely approximates the PDEs is very high. Although several model reductions are available, such reduced models may neglect critical modes of the system, leading to control/observation spillover [7]. However, high-dimensional ODE implementations that retain fine-grained information need processors with high-end computational specifications to solve the large matrix sums and products needed at each iteration step. These specifications are generally cost-prohibitive for *real-time* control/estimation applications.. In this paper, we propose a potential solution to this high computation requirement by exploiting a distributed computation framework, which will essentially reduce the computation time, so that the high-dimensional controllers/estimators can actually meet hard real-time constraints. We illustrate the proposed scheme in the context of linear, parabolic, boundary controlled and boundary measured PDEs.

In our problem, the objective is to solve a computation task of high computational burden by partitioning the overall computation tasks and distributing them over multiple available computing resources. In the literature, few works exist that try to exploit the distributed computing for offline control design or analysis problems like optimization of polynomials [8] and large-scale robust stability [9]. Here, our proposed distributed computing approach addresses an online or real-time computation problem. Note that, the computational burden mentioned here arises from the high dimension (in terms of number of states) of the control/estimation model, not from the structural complexities of the model (such as nonlinearities).

As there are communications among the computing resources, the communication uncertainties may potentially affect the performance of the controller or estimator significantly. The finite difference discretization is one of the natural and widely used forms in the implementation of the PDEs. However, the communication requirement among several computing resources could be high in such natural discretization which in turn introduces significant communication uncertainties in the controller/estimator. The proposed distributed scheme minimizes the effect of such uncertainties in three ways: 1) by using a diagonal form (obtained by transforming the original natural finite difference discretization form) that requires significantly less

*Research supported by U.S. Department of Energy GATE Program.

S. Dey* is with the Department of Civil and Environmental Engineering, University of California, Berkeley, CA 94720, USA. (*Corresponding author; fax: 864-283-7208; e-mail: satadru86@berkeley.edu).

Y. Wang is with Dept. of Electrical and Computer Engineering , Clemson University, Clemson, SC 29634, USA (e-mail: yongqiw@clemson.edu).

B. Ayalew is with the Applied Dynamics and Control Group, Dept. of Automotive Engineering, Clemson University, Greenville, SC 29607, USA (e-mail: beshah@clemson.edu).

communication among the computing resources as compared to the original natural discretization form, 2) by considering an explicit model of network uncertainties in the controller/estimator design phase to enhance the performance and, 3) by including a filtering solution to suppress the effect of network uncertainties. The proposed scheme is illustrated on a state estimation problem of individual battery cells in vehicular battery pack.

The paper is organized as follows. Section II provides the problem formulation, discusses a computation scheme for PDEs using the natural discretization via finite-difference method and then proposes the diagonalization-based distributed computation scheme. Section III discusses the application of the proposed scheme for battery PDE state estimation along with simulation studies in Section IV. Section V summarizes the conclusion of the work.

II. Distributed Computation Scheme for PDEs

A. Problem Formulation

Consider a linear, parabolic, boundary controlled and boundary measured PDE as described below:

$$\frac{\partial c(x,t)}{\partial t} = D\frac{\partial^2 c(x,t)}{\partial x^2}, 1 \geq x \geq 0, t \geq 0 \quad (1)$$

with the following boundary conditions:

$$c(0,t) = 0, \frac{\partial c(1,t)}{\partial x} = Ku, t \geq 0 \quad (2)$$

and measurement equation:

$$y = c(1,t), t \geq 0 \quad (3)$$

where c is some dependent variable which is a function of both time $t \in [0,\infty)$ and space $x \in [0,1]$, $D \in R^+$ and $K \in R$ are known scalar coefficients, $u \in R$ is the scalar control input acting on the boundary and $y \in R$ is the boundary measurement. Such PDEs have broad applications in diffusion problems [10].

B. PDE Observer Computation Scheme via Natural Finite Difference Discretization Form

The PDE described in (1)-(3) can be approximated by a set of ODEs using the method of line technique where the spatial derivatives are approximated using central finite difference methods. The spatial domain is discretized in N nodes $[0, \Delta, 2\Delta, \dots, 1]$ where $\Delta = 1/N$. Correspondingly, the dependent variable c is discretized into a set of variables each of which corresponds to each node as $[c_0, c_1, \dots, c_N]$. Then, the first and second order spatial derivatives are approximated using finite central difference methods as:

$$\frac{\partial c_i}{\partial x} \approx \frac{c_{i+1} - c_{i-1}}{2\Delta}$$
$$\frac{\partial^2 c_i}{\partial x^2} \approx \frac{c_{i+1} - 2c_i + c_{i-1}}{\Delta^2} \quad (4)$$

Based on the discretization (4), the PDE is converted to a set of ODEs as given below:

$$\dot{c}_1 = -2ac_1 + ac_2$$
$$\dot{c}_j = ac_{j-1} - 2ac_j + ac_{j+1}$$
$$\dot{c}_N = 2ac_{N-1} - 2ac_N + 2\frac{K}{\Delta}u \quad (5)$$

where $j = 2, .., N-1$ and $a = D/\Delta^2$. Now, using (5), the ODE state-space model can be formed as below:

$$\dot{X} = AX + Bu$$
$$Y = CX \quad (6)$$

where $X = [c_1, \dots, c_N]^T \in R^N$ is the state vector , $u \in R$ is the control, $Y \in R$ is the boundary measurement, $A \in R^{N\times N}$ is a system matrix, $B = [0,0,..,2K/\Delta]^T \in R^{N\times 1}$, $C = [0,0,..,1] \in R^{1\times N}$. The structure of the matrix is given as follows:

$$A = a\begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 2 & -2 \end{bmatrix} \quad (7)$$

Now, we seek to design an observer for estimating the states of this reduced ODE system (6). The most common form of the observer structure in estimation theory is the copy of the system along with an output error injection term. In context to our system (6), the observer structure can be written as:

$$\dot{\hat{X}} = A\hat{X} + Bu + L\tilde{Y}$$
$$\hat{Y} = C\hat{X} \quad (8)$$

where $\hat{X}$ and $\hat{Y}$ are the estimated state vector and the estimated output, $\tilde{Y} = Y - \hat{Y}$ is the output error and $L \in R^{N\times 1}$ is the observer gain vector to be designed.

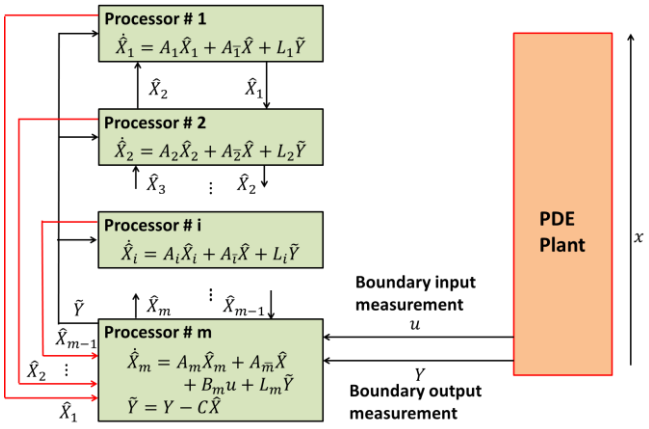

Figure 1: Implementation schematic of the PDE observer using distributed computing scheme via natural discretization (tri-diagonal form). The measurement of boundary output and boundary control input is received by processor m which is assigned the last partition of the computation. Moreover, processor m is receiving the states from the other processors and reconstructing the full-state vector.

When the dimension of system (8) is high, the computational load for solving it becomes unacceptable for many individual low-cost processors. Therefore, we consider a scenario where we have m number of computing nodes or processors available to distribute the computation of the observer dynamics (8). The natural way to allocate the computation to the processors is to partition the overall observer dynamics into m chunks and assign each chunk to one processor. The partitioned dynamics can be written as:

$$\dot{\hat{X}}_i = A_i\hat{X}_i + A_{\bar{i}}\hat{X} + B_i u + L_i\tilde{Y} \quad (9)$$

where $i = 1, \dots, m$, $\hat{X} = [\hat{X}_1^T, \hat{X}_2^T, \dots, \hat{X}_m^T]^T$, $B = [B_1^T, B_2^T, \dots, B_m^T]^T$, $L = [L_1^T, L_2^T, \dots, L_m^T]^T$, A_i is the i-th block on the diagonal of matrix A, and $A_{\bar{i}}$ is constructed by the corresponding rows in matrix A after replacing A_i elements with 0. Essentially, the term $A_{\bar{i}}\hat{X}$ represents the interaction of each partitioned block with their adjacent neighboring blocks. This interaction is due to the tri-diagonal nature of the original system matrix A. Moreover, note that B_i^T for $i = 1, .., m-1$ are zero vectors of appropriate dimensions due to the boundary controlled nature of the PDE plant. The implementation of this distributed computation structure is shown in Fig. 1.

C. Modeling Communication Uncertainty

Note that, in the implementation structure in Fig. 1, the processors are communicating among each other through wired or wireless communication networks. Communication networks can be highly unpredictable. Hidden terminal problems may also exist, causing transmission collisions and thus message losses in the networks [11]. In fact, for many purposes (such as real-time control), it is sometimes advantageous to discard old, non-transmitted messages when a new message becomes available as the newer message is fresher than non-transmitted old messages and is more valuable for real-time applications. In memoryless channels, message losses result in unreliable communication links with a successful transmission rate $0 < p_{i,j} < 1$. That is, with probability $p_{i,j}$, a message from processor i can be successfully sent to processor j, and with probability $1 - p_{i,j}$ the message will be lost. If we represent the channel status as a random variable $\theta_{i,j} \in \{0, 1\}$, then $P\{\theta_{i,j} = 1\} = p_{i,j}$. Considering these network related uncertainties, the observer dynamics can be written as:

$$\begin{aligned} \dot{\hat{X}} &= \theta_x A\hat{X} + Bu + \theta_{\tilde{Y}} L\tilde{Y} \\ \hat{Y} &= C\hat{X} \\ \bar{X} &= \theta_f \hat{X} \end{aligned} \tag{10}$$

where $\bar{X}$ is the full state vector reconstructed at processor m and $\theta_x, \theta_{\tilde{Y}}, \theta_f \in R^{N\times N}$ are time-varying diagonal matrices whose elements $\theta_{x,\mathrm{jj}}, \theta_{\tilde{Y},\mathrm{jj}}$ and $\theta_{f,\mathrm{jj}}$ are stochastic and $\theta_{x,\mathrm{jj}}, \theta_{f,\mathrm{jj}}, \theta_{\tilde{Y},\mathrm{jj}} \in \{0,1\}$ where $j = 1, .., N$. These stochastic elements represent the packet drops in communication. Specifically, θ_x represents the packet loss in the transmission of boundary nodes for each processor (as each processor interacts with two other processors). Then, $\theta_{\tilde{Y}}$ represents the packet loss in the transmission of output error from processor m to all other processors. Finally, θ_f represents the packet loss in the transmission of the partitioned states to processor m for reconstruction of the full state-vector. Note that, the last chunk of the computation is given to the processor which is receiving the measurement information from the plant directly (refer to Fig. 1). This assignment reduces the communication related uncertainties in the Bu term and the $C\hat{X}$ term.

However, this particular distributed computation approach requires exchanging observer state information ($\hat{X}_i$) between processors (due to the presence of $A_{\bar{i}}\hat{X}$ term in (9)), which might be problematic due to two reasons:

1) At each time step k, the calculation of ODE in one processor depends on the observer state information $\hat{X}_i$ from neighboring processors. If the calculation of ODEs in different processors are not in pace, which is highly possible due to differences in computational power and conditions in different processors, then one processor conducting the k-th step ODE iteration needs to wait until neighboring processors finish the k-th step calculation and pass results to it. In other words, calculation paces between different processors need to be precisely synchronized, which is not an easy job, if at all possible.

2) State information needs to be exchanged every ODE iteration step which means substantial communication overhead.

Therefore, the exchange of estimated state information $\hat{X}_i$ should be minimized and even prevented via alternative computation schemes. Next we propose a new approach which can prevent exchanging the state information.

D. PDE Observer Computation Scheme via Diagonal Form

Here, we propose to systematically explore the characteristics of the dynamical systems under consideration and decompose the system according to the internal coupling between the states. Given that the internal coupling between states is embedded in the original system matrix A, it can be revealed by analyzing the structure of matrix A. We propose to use following real eigen-decomposition of system matrix A to probe the structure of internal coupling:

Step 1: Diagonalize A matrix with $A = P^{-1}\Lambda P$ with similarity transformation matrix P which is invertible, where Λ is the real Jordan form [22] having a block diagonal form $\Lambda = diag\{\Lambda_1, \Lambda_2, \dots, \Lambda_m\}$.

Step 2: The new transformed system dynamics become:

$$\begin{aligned} \dot{Z} &= \Lambda Z + \bar{B}u \\ Y &= \bar{C}Z \end{aligned} \tag{11}$$

where $Z = PX$, $\bar{B} = PB$ and $\bar{C} = CP^{-1}$.

Step 3: The observer structure can be chosen as:

$$\begin{aligned} \dot{\hat{Z}} &= \Lambda\hat{Z} + \bar{B}u + L_Z\tilde{Y} \\ \hat{Y} &= \bar{C}\hat{Z} \end{aligned} \tag{12}$$

where $\hat{Z}$ and $\hat{Y}$ are the estimated state vector and the estimated output, $\tilde{Y}$ is the output error and $L_Z \in R^{N\times 1}$ is the observer gain vector to be designed.

Step 4: Now, considering the existence of m processors, the observer dynamics (12) can be decomposed into m independent subsystems as given below:

$$\begin{aligned} \dot{Z}_1 &= \Lambda_1 Z_1 + \bar{B}_1 u + L_{Z1}\tilde{Y} \\ \dot{Z}_2 &= \Lambda_2 Z_2 + \bar{B}_2 u + L_{Z2}\tilde{Y} \\ &\vdots \\ \dot{Z}_m &= \Lambda_m Z_m + \bar{B}_m u + L_{Zm}\tilde{Y} \end{aligned} \tag{13}$$

where $i = 1, .. m$, $\bar{B}_i$ and L_{Zi} is the corresponding i-th block in $\bar{B}$ and L_Z.

Step 5: Distribute the m independent subsystems in (13) to m available processors. Each subsystem Z_i cannot be separated and must be allocated to one processor. Since subsystems are completely decoupled from each other, there is no need to exchange any estimated state information $\hat{Z}_i$ between processors. All estimated states $\hat{Z}_1, \hat{Z}_2, \ldots, \hat{Z}_m$ are transmitted to one particular processor where the original state will be reconstructed as $\hat{X} = P^{-1}\hat{Z}$.

The distributed computing scheme for this diagonalized discretization is shown in Fig. 2.

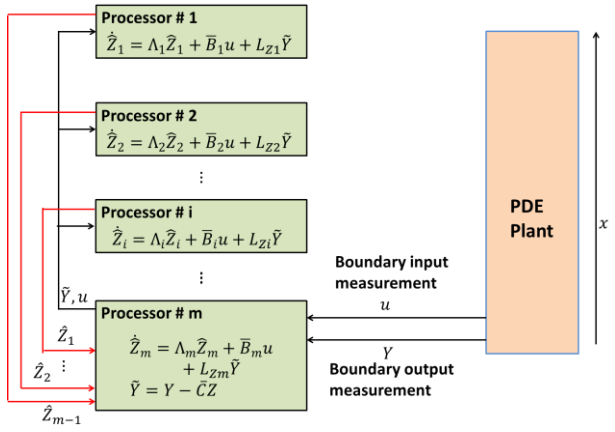

Figure 2: Implementation schematic of the PDE observer using distributed computing scheme via diagonal form. The measurement of boundary output and boundary control input is received by processor *m* which is assigned the last partition of the computation. Moreover, processor *m* is receiving the states from the other processors and reconstructing the full state vector.

Now, considering the network induced uncertainties, we can write the observer dynamics as:

$$\dot{\hat{Z}} = \Lambda\hat{Z} + \theta_u\bar{B}u + \theta_{\tilde{Y}}L_Z\tilde{Y}$$
$$\hat{Y} = \theta_f\bar{C}\hat{Z} \qquad (14)$$

where $\theta_u, \theta_{\tilde{Y}}, \theta_f \in R^{N\times N}$ are time-varying diagonal matrices whose elements $\theta_{u,\mathrm{jj}}$, $\theta_{\tilde{Y},\mathrm{jj}}$ and $\theta_{f,\mathrm{jj}}$ are stochastic and $\theta_{u,\mathrm{jj}}, , \theta_{\tilde{Y},\mathrm{jj}}, \theta_{f,\mathrm{jj}} \in \{0,1\}$ where $\mathrm{j} = 1, .., \mathrm{N}$. These stochastic elements represent the packet drops in communication. Specifically, θ_u represents the packet loss in the transmission of input information from processor *m* to other processors, $\theta_{\tilde{Y}}$ represents the packet loss in the transmission of output error from the processor *m* to all other processors, and θ_f represents the packet loss in the transmission of the partitioned states to the processor *m* for reconstruction of the full state-vector.

In the above formulation, the uncertainties in θ_u, $\theta_{\tilde{Y}}$, and θ_f will significantly complicate the problem and may even prevent an analytical treatment. To make the problem tractable, we propose the following design which will significantly reduce the complexity of the problem. First note that both u and $\tilde{Y}$ are transmitted from the processor *m* to the rest of the processors. Therefore, we can transmit u and $\tilde{Y}$ in the same packets together, which means that their transmission statuses will be the same. Therefore, by this design, we always have $\theta_u = \theta_{\tilde{Y}} = \theta_{u,\tilde{Y}}$. Secondly, if the message $\hat{Z}_i$ is lost, then it is impossible to get an accurate estimate of $\tilde{Y}$. So in this case we can choose not to generate $\tilde{Y}$ and wait for the next $\hat{Z}_i$. In this way, $\tilde{Y}$ is generated only when all $\hat{Z}_i$ are successfully transmitted. This reduces the number of input injections to observers, which, however, can be acceptable if the process dynamics is slow. Therefore, by using this formulation, the observer design will be conducted under the multi-rate sampling framework, but the uncertainty in θ_f is completely avoided. Then, the observer dynamics becomes:

$$\dot{\hat{Z}} = \Lambda\hat{Z} + \theta_{u,\tilde{Y}}\{\bar{B}u + L_Z\tilde{Y}\}$$
$$\hat{Y} = \bar{C}\hat{Z} \qquad (15)$$

Note that, there are three major advantages of the diagonal formulation as compared to the natural tri-diagonal discretization. First, as discussed before, the diagonal form avoids the synchronization requirement of the processors. Second, diagonal form reduces the communication overhead. Third, diagonal form observer (15) has only one uncertain element ($\theta_{u,\tilde{Y}}$) in its dynamics while the natural discretization form observer (10) has at least two different uncertain elements (θ_x and $\theta_{\tilde{Y}}$). Therefore, from the observer design viewpoint, the diagonal form has a clear advantage.

Remark 1: The proposed approach is based on a system transformation which in turn requires computing the matrix inversion (P^{-1}). However, this matrix inversion can be computed offline and can be stored a priori in the processors. Therefore, the transformation would not add extra real-time or online computational cost.

Now, to suppress the effect of uncertainties, we include a Kalman filter for state observer design in the presence of packet-loss [12]. With the plant model (11) and observer structure (15), the error dynamics can be written as:

$$\dot{\tilde{Z}} = \Lambda\tilde{Z} + \{1 - \theta_{u,\tilde{Y}}\}\bar{B}u - \theta_{u,\tilde{Y}}L_Z\tilde{Y}$$
$$\tilde{Y} = \bar{C}\tilde{Z} \qquad (16)$$

Then the observer gain L_Z can be designed by Kalman filtering method to suppress the effect of uncertainties [13].

Remark 2: In this distributed computation scheme, the allocation of the computational load to each processor should be done before the estimation algorithm starts. The main objective is that the computational load assigned to each processor should be determined according to their computational capacity. One possible way to accomplish this objective is the use of consensus protocols [23].

Remark 3: Although 1D parabolic PDE is used for illustration, this scheme can be extended to other forms of PDEs. However, the main condition for applicability of the scheme is: the PDE under consideration can be approximated as an ODE model with system matrices transformable into diagonal form.

Remark 4: Apart from the uncertainties arising from the communication problems in the distributed computation, the Kalman filter can also handle the sources of uncertainties, e.g. unmodeled dynamics and parametric deviations.

III. Application of the Distributed Computation Scheme to Automotive Battery State Estimation

Here, we discuss Lithium-ion battery state estimation in

automotive applications as an illustration of the above distributed computation scheme for PDEs. Li-ion batteries are one of the most prominent energy storage solutions in PHEV and EV applications [14]. In this paper, we use a reduced electrochemical PDE model called Single Particle Model (SPM) [15-17] to illustrate the idea of the proposed distributed PDE computation scheme.

Note that, the SPM consists of a diffusion PDE that describes the distributed Li-ion concentration in a battery cell. This PDE can be approximated as a set of ODEs using the finite difference discretization methods discussed in previous sections. The dimension of the subsequent ODE model can be large in case of finer discretization which is required to retain the accuracy of the original PDE model to a sufficient extent. Moreover, when the SPM modeling approach is extended to battery packs in a vehicle consisting of hundreds of battery cells, the overall dimension of the ODE models can be significantly large and may be computationally cumbersome for a single Battery Management System (BMS) processor in the vehicle. For example, for a battery pack with 100 cells and each cell being modeled by a discretized SPM with 50 nodes, the overall ODE dimension for the pack would be 5000. Under this circumstance, the proposed distributed computation scheme can be useful to speed up the computation without adding extra computation hardware in the vehicle. As discussed in the previous sections, a set of idle vehicular computing processors can be utilized for the distributed computation scheme for vehicular applications. In this paper, we adopt the following state-observable SPM [18] that captures the negative electrode Li-ion diffusion dynamics (17) and the nonlinear voltage output map (18).

$$\begin{aligned}
&\frac{\partial c_s^-}{\partial t} = \frac{D_s^-}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial c_s^-}{\partial r}\right)\\
&\left.\frac{\partial c_s^-}{\partial r}\right|_{r=0} = 0, \qquad \left.\frac{\partial c_s^-}{\partial r}\right|_{r=R^-} = \frac{-I}{a_s^- F D_s^- A L^-} \qquad (17)\\
&V = \frac{\bar{R}T}{\alpha^+ F}\sinh^{-1}\left(\frac{I}{2a_s^+ A L^+ i_0^+}\right)\\
&-\frac{\bar{R}T}{\alpha^- F}\sinh^{-1}\left(\frac{I}{2a_s^- A L^- i_0^-}\right)\\
&+U^+(k_1 c_{sM} + k_2) - U^-(c_{sM}) - R_f I \qquad (18)
\end{aligned}$$

where $c_s^- = c_s^-(r,t)$ and c_{sM} are the Li-ion concentration and boundary/surface concentration, I is the battery cell input current which is a boundary actuation/control and $V = V(c_{sM}, I)$ is the output voltage which is boundary measurement. The rest of the nomenclature can be found in [19]. Note that, the PDE model state $c_s^-(r,t)$ in (16) can be used to compute bulk State-of-Charge (SOC). The PDE model described in (17) and (18), has a nonlinear output function V. However, as noted in [18], the output function $V(c_{sM}, I)$ has a one-to-one correspondence with respect to the surface concentration state c_{sM} and therefore V can be inverted to obtain the c_{sM} information. Consequently, the PDE in (17) can effectively be considered as boundary-measured. The ODE approximation using natural finite difference discretization can be found in [20]. We can transform the natural finite difference discretization form to the diagonal form by the methodology described in the previous section.

IV. Simulation Results

In this section, we present the simulation results of the distributed implementations of the observer design based on the SPM. The battery SPM parameters are taken from [21] for the simulations. In this case study, we assumed four individual vehicular processors are participating in the estimator implementation. Each battery pack contains 300 battery cells. For illustration, we have considered a 30-node discretization of the each battery cell model. In case of all four battery packs needing their SOC to be computed for each cell, the total state variable dimension of the system shoots up to 36000. To illustrate the estimation performance, we have chosen a representative battery cell SOC evolution. For this particular battery cell with 30 states, the four processors are assigned with 10, 10, 5 and 5 states. To justify the effectiveness of the diagonalization-based computation scheme, we compare it with the natural discretized form. Moreover, to implement the packet-loss scenario we assumed a packet dropout probability of 0.02.

In the scenario simulated, a 3A discharge scenario is considered. The output voltage estimation performance is shown in Fig. 3. The estimated bulk SOC and the estimation error are shown in Fig. 4. It is clear that the distributed scheme with the diagonalization form performs better than the natural discretization form.

In the next study, we evaluate the performance of both schemes by varying the probability of the packet-loss. The result is shown in Fig. 5. It is evident that with a higher probability of packet-loss the performance of the natural discretization scheme degrades whereas the diagonalization based scheme is able to maintain robustness to the increased packet dropout probability.

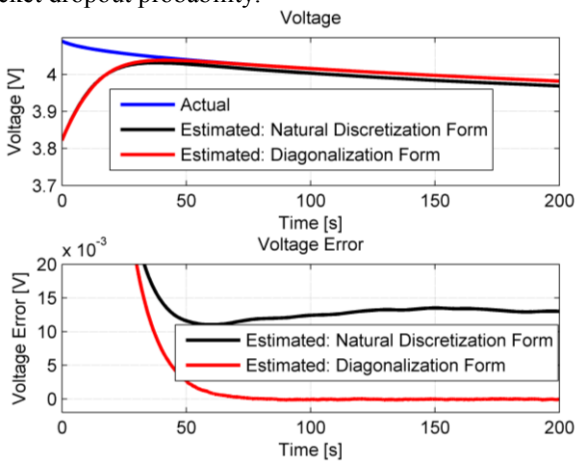

Figure 3: Voltage estimation performance for constant input current discharge

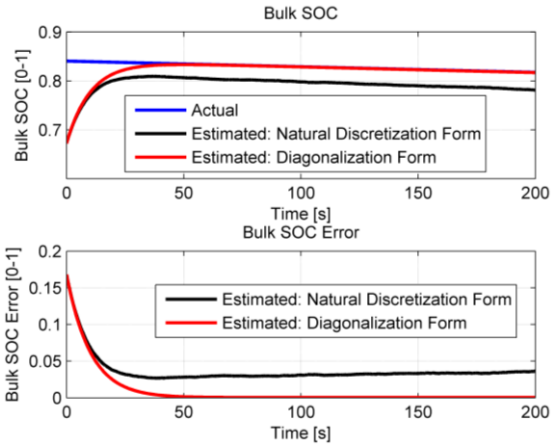

Figure 4: Bulk SOC estimation performance for constant input current discharge

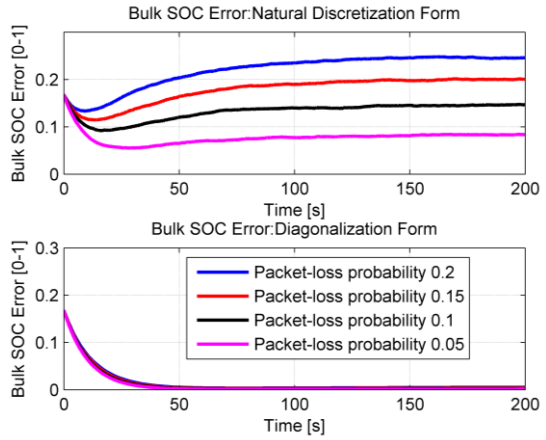

Figure 5: Bulk SOC error of the schemes under different packet-loss probabilities.

V. Conclusion

In this paper, a distributed computing scheme is presented for estimation/control of systems modeled by PDEs. The purpose of the distributed computation scheme is to eliminate the need for a single expensive computational resource for implementing the high-dimension approximated ODEs in the control/estimator design. The scheme distributes the computation load to available computing resources to meet the afore-mentioned objective. One of the major issues in a real-time distributed computation scheme lies in the uncertainties induced by the imperfections of communication networks which can lead to degraded performance of the designed estimator. The proposed computation scheme offers robustness to such uncertainties by using a diagonalization-based approach to reduce the communication requirement among the computational resources. Further, an explicit model of network uncertainties is considered in the estimator design stage to enhance the robustness. The effectiveness of the scheme is demonstrated by simulation.

References

[1] D. P. Bertsekas, and J. N. Tsitsiklis, "Some aspects of parallel and distributed iterative algorithms—a survey," *Automatica*. vol. 27, no.1, pp.3-21, 1991.

[2] H. Attiya, and J. Welch. *Distributed computing: fundamentals, simulations, and advanced topics*. Vol. 19. John Wiley & Sons, 2004.

[3] Aamo, Ole Morten, and Miroslav Krstic. *Flow control by feedback: stabilization and mixing*. Springer Science & Business Media, 2002.

[4] W. Hong, Y. T. Lee, and H. Gong, "Thermal analysis of layer formation in a stepless rapid prototyping process," *Applied Thermal Engineering*, vol. 24, no. 2, pp. 255–268, 2004.

[5] S. S. Ge, T. H. Lee, G. Zhu, and F. Hong,"Variable structure control of a distributed-parameter flexible beam," *Journal of Robotic Systems*, vol. 18, no. 1, pp. 17–27, 2001.

[6] Z. Hidayat, R. Babuska, B. De Schutter, and A. Nunez, "Observers for linear distributed-parameter systems: A survey," In *IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pp. 166-171, 2011.

[7] M. J. Balas, and C. R. Johnson, "Adaptive control of distributed parameter systems: The ultimate reduced-order problem," In *18th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes*, vol. 2, 1979.

[8] M. M. Peet, and Y. V. Peet, "A parallel-computing solution for optimization of polynomials," In *American Control Conference (ACC)*, 2010.

[9] R. Kamyar, M. M. Peet, and Y. Peet, "Solving Large-Scale Robust Stability Problems by Exploiting the Parallel Structure of Polya's Theorem," *IEEE Transactions on Automatic Control*, vol. 58, no. 8 pp. 1931-1947, 2013.

[10] T. S. Ursell, "The Diffusion Equation A Multi-dimensional Tutorial," California Institute of Technology, Pasadena, Tech. Rep., 2007.

[11] L. Zhang, H. Gao, and O. Kaynak, "Network-induced constraints in networked control systems—a survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, 403-416, 2013.

[12] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceedings of IEEE*, vol. 95, no. 1, pp. 138–162, 2007.

[13] F. L. Lewis. *Optimal estimation: with an introduction to stochastic control theory*. New York, Wiley, 1986.

[14] A. G. Boulanger, A. C. Chu, S. Maxx, and D. L. Waltz, "Vehicle electrification: Status and issues," *Proceedings of the IEEE,* vol. 99, no. 6, pp. 1116-1138, 2011.

[15] N. A. Chaturvedi, R. Klein, J. Christensen, J. Ahmed, and A.Kojic, "Algorithms for advanced battery-management systems," *IEEE Control Systems Magazine*, vol. 30, no. 3, pp. 49-68, 2010.

[16] S. Santhanagopalan, and R. E. White, "Online estimation of the state of charge of a lithium ion cell," *Journal of Power Sources*, vol. 161, no. 2, pp. 1346-1355, 2006.

[17] D. D. Domenico, A. Stefanopoulou, and G. Fiengo, "Lithium-ion battery state of charge and critical surface charge estimation using an electrochemical model-based extended Kalman filter," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 132, no. 6, pp. 061302, 2010.

[18] S. J. Moura, N. A. Chaturvedi, and M. Krstic, "PDE estimation techniques for advanced battery management systems—Part I: SOC estimation," In *2012 American Control Conference (ACC)*, pp. 559-565, 2012.

[19] S. Dey, B. Ayalew, and P. Pisu, "Nonlinear Robust Observers for State-of-Charge Estimation of Lithium-Ion Cells Based on a Reduced Electrochemical Model," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 5, pp. 1935-1942, 2015.

[20] S. Dey, B. Ayalew, and P. Pisu, "Nonlinear Adaptive Observer Design for a Lithium-Ion Battery Cell Based on Coupled Electrochemical-Thermal Model," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 11, pp. 111005, 2015.

[21] K. A. Smith, C. D. Rahn, and C. Wang, "Model-based electrochemical estimation and constraint management for pulse operation of lithium ion batteries," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 3, pp. 654-663, 2010.

[22] D. S. Bernstein. *Matrix mathematics: theory, facts, and formulas*. Princeton University Press, 2009.

[23] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," In *Proceedings of the IEEE*, vol. 95, no.1, pp. 215-233, 2007.