Proceedings of BS2013:
13th Conference of International Building Performance Simulation Association, Chambéry, France, August 26-28

STRUCTURED BUILDING MODEL REDUCTION TOWARD PARALLEL SIMULATION

Justin R. Dobbs and Brandon M. Hencey
Cornell University, Ithaca, NY

ABSTRACT

Building energy model reduction exchanges accuracy for improved simulation speed by reducing the number of dynamical equations. Parallel computing aims to improve simulation times without loss of accuracy but is poorly utilized by contemporary simulators and is inherently limited by inter-processor communication. This paper bridges these disparate techniques to implement efficient parallel building thermal simulation. We begin with a survey of three structured reduction approaches that compares their performance to a leading unstructured method. We then use structured model reduction to find thermal clusters in the building energy model and allocate processing resources. Experimental results demonstrate faster simulation and low error without any interprocessor communication.

INTRODUCTION

Emerging software tools promise to streamline the architectural process by bridging conceptual design to energy simulation. These tools save time by translating a CAD model directly to a building energy model (BEM), but they can lengthen simulation time by conveying insignificant dynamics and redundant states to the BEM. Meanwhile, simulators have not kept pace as computers have become increasingly parallel. The result has been excessive runtimes that discourage the use of simulation in early-stage design.

Efforts to reduce simulation runtime have been based on either model order reduction or parallel simulation. Model reduction, historically a manual process requiring experience and discretion, is poised to become accessible to more users through automation (Dobbs and Hencey 2012). Nonetheless, differences among various reduction schemes make choosing the best method for a given situation difficult. Parallel simulation methods suffer from startup or communication overhead or benefit certain scenarios that exclude real-world, nonlinear building energy simulations.

Our contribution is two-fold. First, we begin with a survey of two broad classes of model reduction—structured and unstructured—and detail three structured approaches: state deletion, state aggregation, and structured balanced truncation. Although all three are considered structured, their properties and applications differ considerably. Removal of states from a physical state space (Papachristodoulou et al. 2010) is simple but gives poor error performance. State aggregation applies narrowly to resistor-capacitor thermal networks but offers insight into the building's thermal structure (Deng et al. 2010; Dobbs and Hencey 2012). Structured balanced truncation reconfigures the state space under the constraint of an externally-imposed macrostructure using an energy-based stability analysis (Sandberg and Murray 2008).

Second, after comparing the performance of these methods on a resistor-capacitor network, we present a parallel simulation scheme that uses structured model reduction in two ways. First, it uses state aggregation to extract the building's thermal structure and then uses that information to allocate processing resources. Second, it invokes structured balanced truncation to create partial reduced-order models that allow each processor to iterate autonomously with an approximation of the others; this circumvents the communication bottleneck. The discussion concludes by comparing the method's speed and error performance for various processor configurations using a two-zone building model. Although the results demonstrate performance worse than that of unstructured balanced truncation, we view this work as a positive step toward parallel simulation of real-world nonlinear building energy models.

MODEL REDUCTION BACKGROUND

Building models natively use physical quantities as states, so it makes sense to classify model order reduction methods by whether the results can be mapped back to the object-oriented building energy model (BEM). Those that make limited changes to the original state space and thus preserve its physical integrity are termed *structured*. In contrast, methods that preserve only input-output equivalence while freely reconfiguring the states are called *unstructured*. The widely used unstructured balanced truncation algorithm offers good performance and analytical error bounds, but the resulting state space is physically

unintuitive. Put another way, an unstructured method is best used within a simulator as a one-way process just before simulation begins and not as a way to simplify the source BEM. Structured methods preserve an intuitive relationship to the input model, but this benefit comes at the expense of model accuracy. We now discuss the details of three structured methods. The first two operate directly on the physical state space, while the third employs balanced truncation bound to an externally imposed clustering topology.

Greedy State Removal

This method removes less important states and replaces references to them with steady-state values. Suppose the building model has been converted to a linear time-invariant system $G = \{A, B, C\}$ with state equation $\dot{x} = Ax + Bu$ and output $y = Cx$. The vector x contains the state during simulation, the vector u is an input stream (such as EnergyPlus weather data), and y is the output observation. For a building thermal model, the states are physical quantities such as temperatures. Now suppose the model contains large thermal capacitances plus a few very small ones that have little influence on the overall response. The dynamics of these states decay very quickly and are of minimal importance, so we can remove them from the model. If we group those states into a vector x_c and call the remaining states $\hat{x}$, then the original system state vector can be described by a concatenation of x_c and $\hat{x}$ permuted to their original positions in the vector x using a boolean matrix T: $x = T\begin{bmatrix}\hat{x}^T & x_c^T\end{bmatrix}^T$. The system matrices can then be rewritten in a partitioned form as

$$\begin{bmatrix}\dot{\hat{x}}\\ \dot{x}_c\end{bmatrix} = \underbrace{\begin{bmatrix}\bar{A}_{11} & \bar{A}_{12}\\ \bar{A}_{21} & \bar{A}_{22}\end{bmatrix}}_{\bar{A}}\underbrace{\begin{bmatrix}\hat{x}\\ x_c\end{bmatrix}}_{T^{-1}x} + \underbrace{\begin{bmatrix}\bar{B}_1\\ \bar{B}_2\end{bmatrix}}_{\bar{B}}u,\quad y = \underbrace{\begin{bmatrix}\bar{C}_1 & \bar{C}_2\end{bmatrix}}_{\bar{C}}\begin{bmatrix}\hat{x}\\ x_c\end{bmatrix}, \tag{1}$$

and the permuted matrices of the full-order system are given by

$$\bar{A} = T^{-1}AT,\quad \bar{B} = T^{-1}B,\quad \bar{C} = CT. \tag{2}$$

Because the states in x_c decay quickly, we can set $\dot{x}_c = 0$ and replace references to those states with steady-state relationships. The reduced-order system is then

$$\hat{G} = \begin{cases}\dot{\hat{x}} = \underbrace{\left(\bar{A}_{11} - \bar{A}_{12}\bar{A}_{22}^{-1}\bar{A}_{21}\right)}_{\hat{A}}\hat{x} + \underbrace{\left(\bar{B}_1 - \bar{A}_{12}\bar{A}_{22}^{-1}\bar{B}_2\right)}_{\hat{B}}u\\ y = \bar{C}_1\hat{x}.\end{cases} \tag{3}$$

The above gives a reduced-order model when the states to be eliminated are already known. Choosing the best states to remove is computationally difficult, but a more tractable sub-optimal solution is offered by Papachristodoulou et al. (2010). Suppose we want to remove n states from the system, where $n < \dim(x)$. (Without an automated way to choose the degree of reduction, the user must choose n by trial-and-error.) We then proceed as given in Algorithm 1, removing physical states in increasing order of their impact on the simulation error. Because the state space was not transformed before reduction, some physical intuition is preserved, and one can delete corresponding BEM objects if a simpler BEM is desired.

Algorithm 1 Greedy state removal (Papachristodoulou et al. 2010)

1. Generate a linear state space model G from the BEM.
2. Initialize $x_c \leftarrow \emptyset$, $\hat{x} \leftarrow x$.
3. Repeat n times:
   (a) Initialize `norms[ ]` $\leftarrow \emptyset$.
   (b) For each state $\hat{x}_i \in \hat{x}$:
      i. Create a temporary state vector $\hat{x}_{\text{temp}} = \hat{x} \setminus \hat{x}_i$ that omits state $\hat{x}_i$. Create a reduced-order system $\hat{G}_{\text{temp}}$ using $\hat{x}_{\text{temp}}$.
      ii. Create an error system $E = G - \hat{G}$ whose output is the error introduced by removing the state.
      iii. Compute the observability Gramian, Q, of the error system, for example using the MATLAB function `gram()`. (Note that Q in this context is not the same as the reduced transition matrix Q given by Equation 4.)
      iv. Compute the eigenvalues of Q. Set `norms[i]` $\leftarrow \bar{\lambda}_Q^{1/2}$, the square root of the largest eigenvalue magnitude.
   (c) Update $\hat{x}$, removing the state corresponding to the smallest norm: $\hat{x} \leftarrow \hat{x} \setminus \hat{x}_k$ where $k = \text{argmin}_i$ `norms[i]`.
   (d) Add $\hat{x}_k$ to the list of omitted states: $x_c \leftarrow x_c \cup \hat{x}_k$.

State Aggregation

Using aggregation on building thermal models has been proposed by Deng et al. (2010). The method requires that all states be combinable through a closed (heat-conservative) resistor-capacitor network whose discrete-time state transition matrix is analogous to a Markov chain. The restriction to RC networks may appear to limit the utility of aggregation, but recent work has shown that state aggregation can reveal the thermal structure of an object-oriented model if an appropriate RC network is used (Dobbs and Hencey 2012). We will use this insight later in the paper to realize parallel simulation. The aggregation procedure is detailed in Algorithm 2.

Algorithm 2 Capacitor aggregation (Deng et al. 2010)

1. Generate a closed resistor-capacitor network from the BEM.
2. Create one supernode (cluster) containing the entire network: $\phi(k) = 1,\ k \in \mathcal{N}$.
3. For each supernode $j \in \mathcal{M}$:
   (a) Extract the continuous transition matrix A.
   (b) Compute the modified discrete transition matrix $\hat{P}$ and its second eigenvector u_2.
   (c) Sort the components of vector u_2 numerically. Each component of u_2 corresponds to a capacitor in the cluster.
   (d) Place all states to one side of the zero-crossing in one cluster, and all others in another cluster. (This divides the supernode into two pieces.)
   (e) Check for disconnected subnetworks within the new supernodes and further subdivide as necessary to ensure full connectivity within each.
   (f) Treating each supernode as the sum of its constituents, compute the reduced-order transition matrix Q followed by the Kullback-Leibler (K. L.) distance between Q and the full-order discrete-time transition matrix P.
4. If no supernodes can be divided, proceed to step 7.
5. Select the non-singleton supernode whose division yields the lowest K. L. divergence.
6. Recurse on step 3.
7. Extract the reduced-order network (see below).

Improvements to the Method. For applications that require a resistor-capacitor network, the equivalent resistance computation scheme greatly impacts the quality of the result. We have improved the accuracy of the equivalent resistance calculation over the reference implementation of Deng et al. (2010). We start with the same reduced-order discrete time transition matrix,

$$Q_{kl}(\phi) = \frac{\sum_{i\in\phi^{-1}),j\in\phi^{-1}(l)} \pi_i P_{ij}}{\sum_{i\in\phi^{-1}(k)} \pi_i}\ k,l \in \mathcal{M}. \qquad (4)$$

The reference implementation approximates the continuous-time reduced-order transition matrix $\bar{A}$ using $\exp(At) \approx I + At$, giving

$$\bar{A}_{kl}(\phi) = \frac{\sum_{i\in\phi^{-1}(k),j\in\phi^{-1}(l)} \pi_i A_{ij}}{\sum_{i\in\phi^{-1}(k)} \pi_i}. \qquad (5)$$

The equivalent resistance matrix follows straightforwardly from $\bar{A}$. Make the partition function ϕ a boolean matrix

$$\phi_{jk} = \begin{cases} 1 & C_j \in \bar{C}_k \\ 0 & \text{otherwise} \end{cases} : \sum_{k\in\mathcal{M}} \phi_{jk} = 1,\ \sum_{\substack{j\in\mathcal{N} \\ k\in\mathcal{M}}} \phi_{jk} = N, \qquad (6)$$

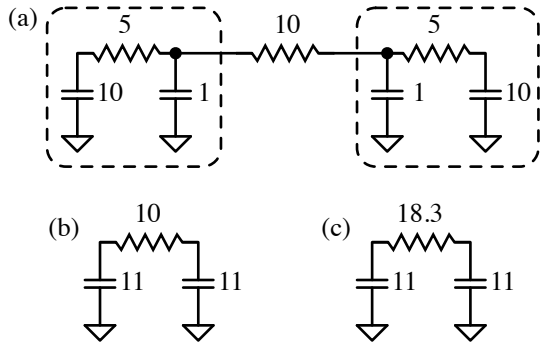

Figure 1: A case that highlights the different results between two equivalent resistance computation methods. (a) Full-order system, with two clusters to reduce; (b) Reduced-order system using the approximation in Deng et al. (2010); (c) Reduced-order system using $\log(Q)$.

meaning $\phi_{jk} = 1$ when capacitor j is a member of the supernode k. Because the capacitance of a supernode is the sum of its members, each element of the reduced-order capacitance vector is $\bar{C}_k = (\phi^T C)_k$. The continuous transition matrix is $\bar{A} = \text{diag}(\bar{C})^{-1}\bar{G}$; rearrange and use $\bar{C} = \phi^T C$ to get

$$\bar{G} = \text{diag}(\phi^T C)\bar{A},\ \ \bar{R}_{ij} = \begin{cases} \bar{G}_{ij}^{-1} & i \neq j \\ 0 & i = j \end{cases}. \qquad (7)$$

Equation 5 effectively ignores resistance within clusters, thereby introducing error. Figure 1(a) depicts a case where the error is significant: a chain with very large capacitors at the ends, small capacitors toward the center, and a large resistance separating the clusters. The influence of the inner capacitors is small, so they can be merged with the outer ones. Thus 20 resistance units separate most of the capacitance and we expect the equivalent reduced-order resistance to be slightly less. The approximation of Equation 5 neglects the resistors within the clusters, however, and yields an approximation of just 10 units, shown in (b).

A more accurate, if computationally more expensive, approach is to take the matrix logarithm. The more accurate transition rate matrix is $\bar{A}' = \log(Q)/\tau$, where the time step $\tau = -\lambda_{N-1}^{-1}(A)$ is the reciprocal of the smallest magnitude non-zero eigenvalue of the full-order transition matrix A. (Selecting τ this way ensures accurate translation of the dominant modes.) The conductance matrix $\bar{G}'$ and more accurate resistance matrix $\bar{R}'$ are then

$$\bar{G}' = \frac{\text{diag}(\phi^T C)\log Q}{\tau},\ \ \bar{R}'_{ij} = \begin{cases} \left(\bar{G}'_{ij}\right)^{-1} & i \neq j \\ 0 & i = j \end{cases}. \qquad (8)$$

Returning to the special case of Figure 1(c), we see that our method yields a reasonable equivalent resistance of

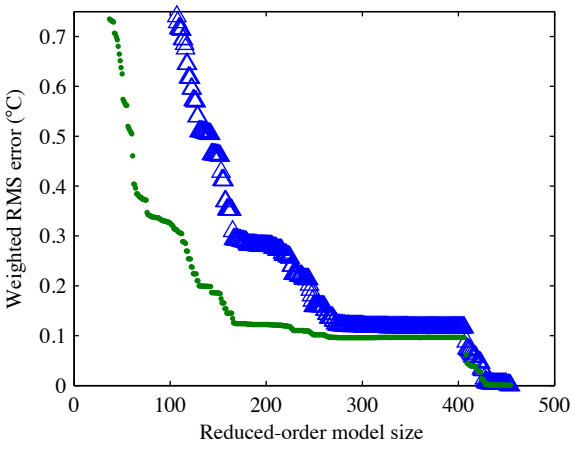

Figure 2: Simulation error for fifteen-room building using approximation Equation 5 (△); using the more accurate $\log(Q)$ *of Equation 8 (·).*

18.3 units. The difference is equally striking in more complex cases. Figure 2 shows a 2:1 difference in simulation error between the two resistance computation methods when applied to a fifteen-room, 454-state building model using the EnergyPlus weather file for Elmira, NY. The weighted root mean square error has been computed using

$$E_{\text{wRMS}} = \sum_{k=1}^{N} \frac{C_k E_k}{\sum_{j=1}^{N} C_j} = E_k \pi_k, \qquad (9)$$

where π_k is the fraction of the system's total thermal capacitance contained in node k, and E_k is the root mean square temperature error for that node across a one-year simulation. This weighting penalizes error in large thermal masses more than smaller ones.

Structured Balanced Truncation

Block-structured reduction methods have been employed extensively for very-large-scale integration (VLSI) circuit simulation to reduce subsystems internally while preserving their interconnection topology (Odabasioglu, Celik, and Pileggi 1997; Yu, He, and Tar 2005). We consider a modified balanced truncation scheme by Sandberg and Murray (2008) in which subsystem blocks are internally balanced and then truncated under assurance of global system stability. The stability guarantee is particularly important when models contain active elements such as HVAC systems. For resistor-capacitor networks, the hierarchical aggregation method presented in Deng et al. (2010) yields a suitable clustering structure to frame the operation.

The procedure begins with a special formulation of the system model. Figure 3(a) depicts the linear fractional transformation (LFT) layout in which each subsystem cluster is disconnected from the others and modeled as a stand-alone system within the lower block-diagonal transfer function matrix G. An interconnection block N connects the subsystems together using feedback. The input-output characteristics of this model exactly match a conventional realization, but the format allows a single set of block-diagonal Gramians to be computed for the entire system. In contrast to conventional unstructured Gramians, which can be obtained by solving an explicit equation, structured Gramians are found by solving linear matrix inequalities (LMIs). The Hankel singular values provide a specific order for states to be removed across the entire model.

Figure 3(b) shows simulation error after structured balanced truncation for a two-room, 43-state RC network. The x axis is the number of clusters imposed on the system. The y axis is the total size of the state space, or the sum of all the subsystem sizes after reduction. The z axis is a weighted sum of root mean square simulation error across all system nodes (Equation 9). Unsurprisingly, the results show that low cluster counts achieve the lowest simulation error; less structure is imposed on the LMI solver. At very low cluster counts, the simulation error approaches that of standard balanced truncation as shown in Figure 4(◇).

We have modified the method of Sandberg and Murray (2008) to avoid deleting the last state in any cluster. Otherwise, the removal of a cluster's last state effectively removes the cluster from existence and severs the thermal communication among neighboring clusters. In the trivial case (not shown in Figure 3(b)) where the number of clusters equals the number of model states, this protection prevents any reduction from taking place, because every cluster is a singleton.

PERFORMANCE COMPARISON

Figure 4 compares the simulation error for the three structured methods to unstructured balanced truncation. We used a 43-state, two-room RC network model generated automatically from CAD data using physically realistic parameters. For each method, we reduced the model incrementally, running a full year simulation at each step using the EnergyPlus weather file for Elmira, NY.

Unstructured balanced truncation (◇) using the MATLAB function `reduce()` yielded the lowest simulation error for the number of states. The greedy state deletion algorithm (•) produced the largest error. Between these extremes, structured balanced truncation, while not as effective as its unstructured counterpart, showed its best performance at low cluster count (▽). Higher cluster counts increased simulation error (△). The cluster configurations were obtained using hierarchical state aggregation (□), which as a standalone model reduction scheme per-

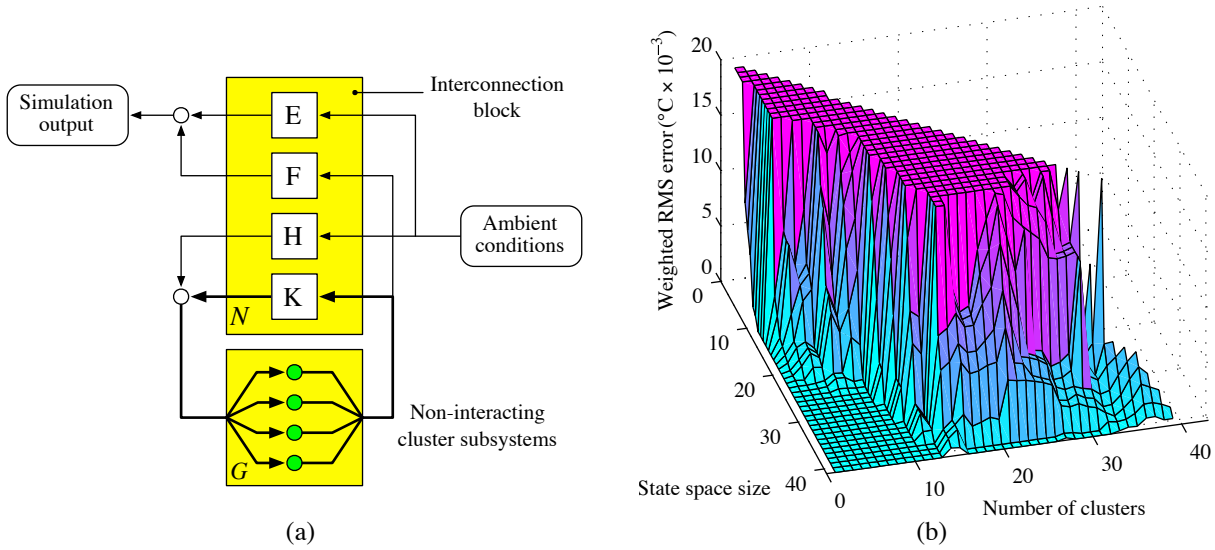

Figure 3: Structured balanced truncation: Interconnection topology, showing the cluster subsystems G and the interconnection block N with feedback path emphasized (a); simulation error versus the total state space size and number of clusters (b). Results are shown for a two-room 43-state model. Higher cluster count implies greater restriction on the reduction. Best performance is achieved with less structure. (Plot is truncated at 20×10^{-3} *°C for clarity.)*

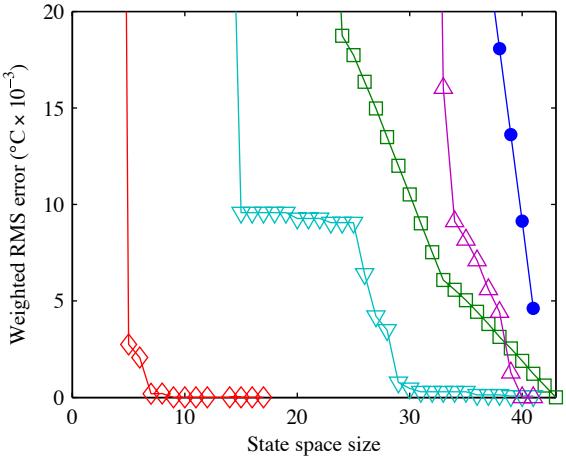

Figure 4: Simulation error comparison for a two-room, 43-state model using unstructured balanced truncation (◇), structured balanced truncation with five clusters (▽), structured balanced truncation with twenty clusters (△), capacitor aggregation (□), and greedy state deletion (●).

formed unimpressively. Clearly, better performance becomes possible as the need for a physically relevant state space is relaxed.

PARALLEL PROCESSING

Simulation speed can be radically improved by using parallel hardware more effectively, but communication overhead limits the gains from adding more processors. We can overcome this bottleneck by decomposing the simulation into tasks that can iterate with little or no cross-communication. Such decoupling may be implemented

1. within large matrix operations,
2. across time,
3. across methods, or
4. across the dynamical model.

Parallelizing linear algebra, as is done by readily available libraries such as LaPACK, can boost the speed of single-threaded simulators when large matrices are used. Unfortunately, the use of large matrices runs counter to software modularity and is ineffective in modular, nonlinear simulation packages.

Decomposition across time, proposed by Garg et al. (2011), splits the simulated period into N consecutive segments and simulates all segments simultaneously. It is particularly appealing for boosting monolithic software such as EnergyPlus using only an external wrapper. For nonlinear simulations, the "warmup period" for computing the starting conditions approaches the actual simulation period, yielding diminishing returns with large processor counts. Transient dynamics are lost across time slice boundaries due to the lack of communication among jobs.

Decomposition across methods combines otherwise disparate software components to work in lockstep (Radosevic, Hensen, and Wijsman 2006), encouraging modularity and data hiding in the software design. It is sensitive to time step choice and, for large models, requires

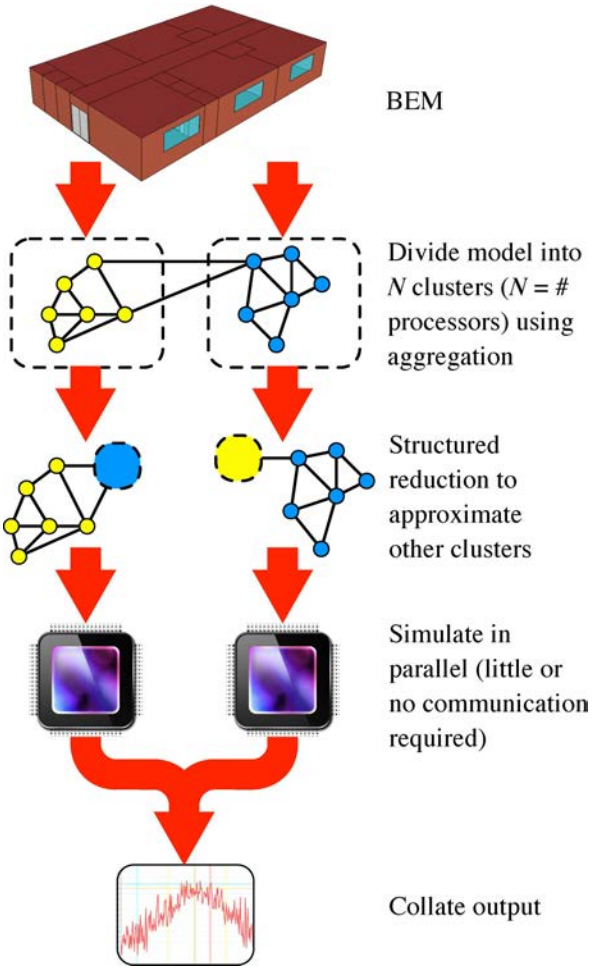

Figure 5: Summary of parallel simulation process using structured model reduction.

considerable data exchange at each iteration to maintain stability. Cross-communication overhead ultimately limits the benefit of this approach.

The above methods take into account properties of the simulation input or algorithms but ignore the structure of the building itself. Dividing the simulation across the model is one way to account for building structure in allocating processors. It has been demonstrated with electric circuit simulation (Frohlich et al. 1998) but not for building simulation in a way that accounts for thermal interactions. Dividing the building along lines of weakest thermal interaction lessens the importance of frequent iteration by reducing the error when communication is skipped. To further reduce cross-communication, each processor also hosts a reduced-order model of the others so that it can iterate independently. The process of partitioning the model, generating the reduced-order surrogate models, and running the simulation is depicted in Figure 5.

The state aggregation algorithm described previously is one way to partition the model to avoid cutting strong thermal connections (Dobbs and Hencey 2012). To decompose the model across M processors, one may terminate the recursive division when M clusters have been found. Each processor is assigned one of the clusters to simulate at full fidelity. The remaining clusters are merged and reduced using structured balanced truncation, and iterating with this reduced-order surrogate model takes the place of iterating with the rest of the system. If the total system size is N states, each processor simulates N/M states plus the reduced-order approximation of the other processors. In this implementation, all models are linear and time-invariant; we view this as an early step toward parallelizing real-world, nonlinear simulations.

Discussion of Results

We have implemented a parallel simulation of the two-room, 43-state building model RC network on two, four, and eight processors, varying the size of the surrogate external models across the admissible range. The EnergyPlus weather file for Elmira, NY has been used for input. Figure 6 shows how states are allocated in the case of two ($\circ$), four ($\triangle$), and eight ($\triangledown$) processors, and Figure 7 shows simulation error and runtime for those configurations. In the transition from two to four processors, the portion of the model assigned to processor #1 (top left) has stronger internal thermal coupling than that running on processor #2 (top right) and therefore is left alone while states from processor #2 are reallocated to the two additional processors. The number of full-fidelity states on the most heavily loaded processor determines the maximum potential simulation speed; in the transition from two to four processors, processor #1 sheds very few states, so the speed gain is only slight. The situation improves substantially with eight processors because the resulting state distribution is more uniform. (A processor running a smaller full-order cluster may run a more complex surrogate model while maintaining the same total load—and the same execution time—as the others.) Simulation error from unstructured balanced truncation ($\diamond$) and state aggregation ($\square$) on a single processor are shown for comparison. Although balanced truncation yields superior performance in this example, the salient feature of our parallel simulation method is its ability to scale with minimal inter-processor communication—an important feature for computer clusters, for example, where communication is expensive.

CONCLUSION

This paper has compared four model reduction methods to examine the tradeoff between structure preservation and simulation accuracy. The results reinforce the well-known result that unstructured methods yield the best per-

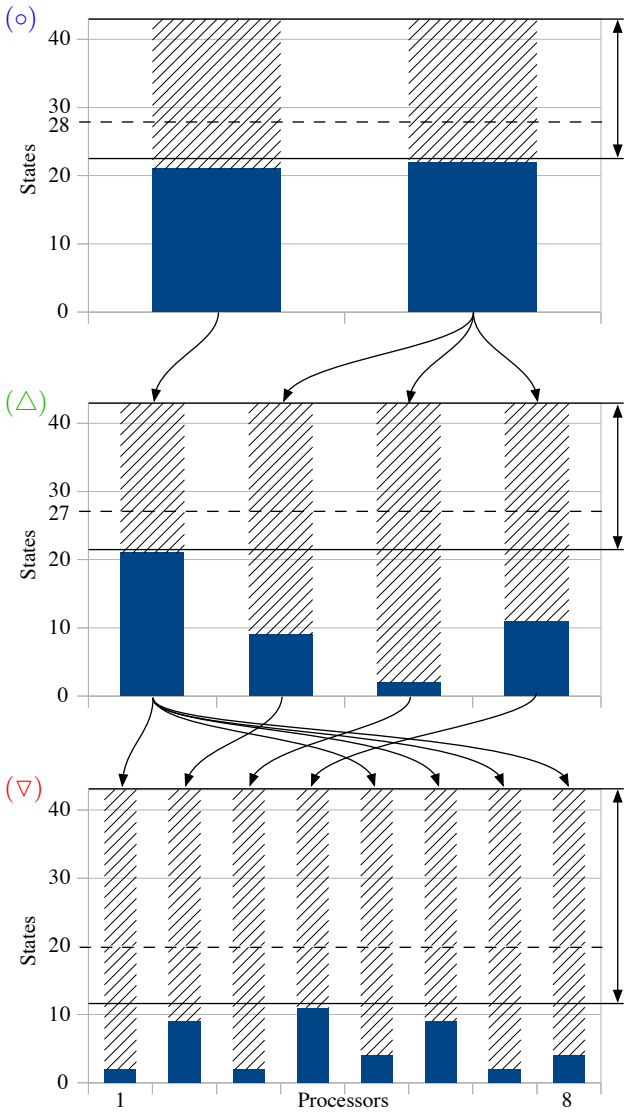

Figure 6: Processor load configuration for 43-state model using two (○), four (△), and eight (▽) processors. The solid bars are unreduced states. The upper portion of each bar is the surrogate model of the other processors. The total number of states on the processors can be varied across the indicated range (↕), where more states yields more accurate reduced-order surrogate models but a slower simulation. The processor load thresholds (dashed) correspond to the dashed error threshold in Figure 7.

formance. Model reduction does not in itself address the poor hardware utilization of contemporary simulators. To that end, we have combined two structured reduction methods into a parallel simulation strategy that allocates processor resources based on thermal structure. The proposed scheme demonstrates good performance with-

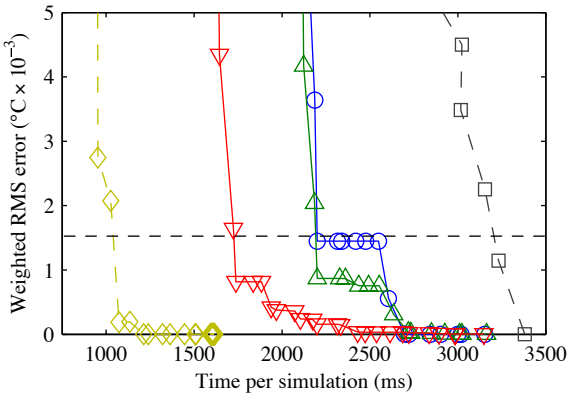

Figure 7: Simulation error vs. run time for 43-state model using two (○), four (△), and eight (▽) processors. The dashed line is placed just above the knee in the simulation error and corresponds to the dashed CPU load levels in Figure 6. Single-threaded simulations of reduced models using aggregation (□) and unstructured balanced truncation (◇) are shown for comparison.

out any cross-processor communication but would benefit from more equal processor loading; in certain cases, an entire processor may be assigned to simulate just one model state for only a slight performance benefit. Distributing states more evenly across processors while still respecting thermal clustering is an obvious area for improvement. Furthermore, although a larger model would have been more illustrative, our attempts to reduce a more complex (454-state) building model with structured balanced truncation were foiled when each of two popular linear matrix inequality solvers exhausted all available computer memory. Addressing computational inefficiencies within the reduction methods themselves, for example by avoiding computationally intractable LMIs, is a worthy direction for future research.

References

Deng, Kun, Prabir Barooah, Prashant G. Mehta, and Sean P. Meyn. 2010. "Building thermal model reduction via aggregation of states." *American Control Conference*. IEEE, 5118–5123.

Dobbs, Justin R., and Brandon M. Hencey. 2012. "Automatic model reduction in architecture: a window into building thermal structure." *Proceedings of the SimBuild, 5th National Conference of IBPSA-USA, Madison, WI*. IBPSA.

Frohlich, N., B.M. Riess, U.A. Wever, and Q. Zheng. 1998. "A new approach for parallel simulation of VLSI circuits on a transistor level." *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on* 45 (6): 601–613.

Garg, Vishal, Kshitij Chandrasen, Jyotirmay Mathur, Surekha Tetali, and Akshey Jawa. 2011. "Development and performance evaluation of a methodology, based on distributed computing, for speeding EnergyPlus simulation." *Journal of Building Performance Simulation* 4 (3): 257–270.

Odabasioglu, A., M. Celik, and L.T. Pileggi. 1997. "PRIMA: Passive reduced-order interconnect macromodeling algorithm." *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*. IEEE Computer Society, 58–65.

Papachristodoulou, A., Y.C. Chang, E. August, and J. Anderson. 2010. "Structured model reduction for dynamical networked systems." *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2670–2675.

Radosevic, M. Trcka, J. L.M. Hensen, and A. J.Th.M. Wijsman. 2006. "Distributed building performance simulation—a novel approach to overcome legacy code limitations." *HVAC&R Research* 12 (sup1): 621–640.

Sandberg, H., and R.M. Murray. 2008. "Model reduction of interconnected linear systems using structured gramians." *Proceedings of the 17th IFAC World Congress*. 8725–8730.

Yu, H., L. He, and SXD Tar. 2005. "Block structure preserving model order reduction." *Behavioral Modeling and Simulation Workshop, 2005. BMAS 2005. Proceedings of the 2005 IEEE International*. IEEE, 1–6.