Taylor & Francis
Taylor & Francis Group

Journal of Building Performance Simulation

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tbps20

Evaluating the performance of an Inexact Newton method with a preconditioner for dynamic building system simulation

Zhelun Chen, Jin Wen, Anthony J. Kearsley & Amanda Pertzborn

To cite this article: Zhelun Chen, Jin Wen, Anthony J. Kearsley & Amanda Pertzborn (2022) Evaluating the performance of an Inexact Newton method with a preconditioner for dynamic building system simulation, Journal of Building Performance Simulation, 15:1, 112-127, DOI: 10.1080/19401493.2021.2007285

To link to this article: https://doi.org/10.1080/19401493.2021.2007285

JOURNAL OF BUILDING PERFORMANCE SIMULATION
2021, VOL. 15, NO. 1, 112–127
https://doi.org/10.1080/19401493.2021.2007285

Evaluating the performance of an Inexact Newton method with a preconditioner for dynamic building system simulation

Zhelun Chen[a], Jin Wen[a], Anthony J. Kearsley[b] and Amanda Pertzborn[c]

[a]Department of Civil, Architectural & Environmental Engineering, Drexel University, Philadelphia, PA, USA; [b]Applied and Computational Mathematics Division, National Institute of Standards and Technology, Gaithersburg, MD, USA; [c]Building Energy and Environment Division, National Institute of Standards and Technology, Gaithersburg, MD, USA

ABSTRACT
Efficiently, robustly, and accurately solving systems of nonlinear differential algebraic equations for dynamic building system simulation is becoming more important due to the increasing need to simulate large-scale problems. The focus of this paper is an investigation of methods for solving the nonlinear algebraic equations encountered in dynamic building system simulations. Dynamic building system simulations employ a myriad of solution techniques, many of which use derivative information. As the problem size grows, so do the memory requirements, leading to challenges in solving large-scale problems. Newton-Krylov methods are a promising candidate for large-scale simulation. The performance of these methods is often improved, sometimes drastically, when employed with a preconditioning technique. Here, a cost-effective preconditioning technique is applied to a Newton-Krylov method and tested on a series of problems. To illustrate the benefits of the approach, comparison is made to the frequently employed Powell's Hybrid Method.

ARTICLE HISTORY
Received 23 October 2020
Accepted 11 November 2021

KEYWORDS
Newton-Krylov method; HVAC simulation; building system; large-scale simulation

1. Introduction

The building sector, including residential and commercial buildings, represents the largest energy-consuming sector in the United States; in 2019 it was responsible for 39% of the country's energy consumption, in comparison to 28% for the transportation sector and 33% for the industrial sector (U.S. Energy Information Administration 2020). Moreover, buildings consume 74% of the electricity in the United States (U.S. Energy Information Administration 2020), which makes the building sector a crucial part of the overall smart grid infrastructure. Given the rapid development of the smart grid and the potential of buildings to store and generate electricity (Goellner et al. 2011) through demand shifting and transactive control, there is a need to improve the dynamic interactions between buildings and the smart grid. The increased size and complexity of these simulations calls for robust, efficient, and accurate dynamic building energy system modelling and simulation.

Unlike traditional dynamic building simulations, which typically focus on one building, dynamic building simulations of smart grid applications simulate large and complex building energy systems and their interactions within building clusters. They are coupled through the smart grid or other means for uses such as district heating/cooling and shared distributed energy resources. This complexity presents a unique and difficult computational challenge: a large system of coupled nonlinear equations need to be solved rapidly and robustly. These equations arise, typically, from the discretization or the approximation of a differential equation (Kearsley 1996) and equations that couple these equations together into a single large system. In cases considered here, a collection of nonlinear equations are assembled into a single large nonlinear system whose solution is sought instead of considering an optimization approach where residuals of the equations are incorporated into a nonlinear program (Boggs, Kearsley, and Tolle 1999).

For example, the simulation model for a heating, ventilation, and air conditioning (HVAC) system in a medium size campus building would not be considered a large-scale building cluster model. Yet, for this seven-story mixed use (classrooms, offices, and laboratories) building with a total area of 7,246 m^2 (78,000 ft^2), a dynamic simulation of its HVAC system, including a primary cooling system (one chiller with one cooling tower), a heating system (two steam-to-water heat exchangers), and three air handling units (AHUs) serving eighty-eight variable

CONTACT Zhelun Chen ✉ zhelun.chen.drexel@gmail.com Department of Civil, Architectural & Environmental Engineering, Drexel University, 3141 Chestnut Street, Curtis 251, Philadelphia, PA, USA

air volume (VAV) terminals, would require the simultaneous solution of a nonlinear system with more than eight hundred state variables at each time step. Given the typical measurement interval of the sensors, the simulation time step is typically in the order of seconds for the simulation for HVAC control development. Considering that many of simulation environments (e.g. HVACSIM+ (Park, Clark, and Kelly 1986) and TRNSYS (Klein et al. 2017)) still employs dense and direct Newton solver, simulating the HVAC operation for one day would consume one or two days on a personal laptop. If multiple buildings of similar size are grouped for a simultaneous solution, the complexity and size of the equation system will increase drastically. Such long computational times are major obstacles to the use of large-scale dynamic simulation models for research into energy management and control strategies or for real-time building energy performance prediction.

Some of the techniques designed for large problems that have been proposed in the literature are discussed in the following paragraphs. For the convenience of illustration, the solution process for a dynamic system simulation is categorized into two steps: (1) translation process; (2) solving nonlinear algebraic equations (obtained from the translation process). The translation process determines many details about the mathematical model formulation, such as how the equations are grouped/decomposed, how the simulation problem is coded in a computer (e.g. dense/sparse), and how the differential equations are integrated/approximated in time. The focus of this study is on solving nonlinear algebraic equations.

Tools that are able to simulate the dynamic behaviour of building systems have been developed, including SPARK (SPARK 2003), TRNSYS (Klein et al. 2017), HVACSIM+ (Park, Clark, and Kelly 1986), and Modelica-based tools such as Dymola (Dynasim 2004), JModelica (Magnusson and Åkesson 2015) and OpenModelica (Open Source Modelica Consortium 2018). Detailed reviews of these simulation tools can be found in (Pourarian et al. 2016; Chen 2019). The major difference among these tools is the translation process by which the mathematical model is formulated. An AHU-VAV system developed in ASHRAE Research Project 825 (Haves, Norford, and DeSimone 1998) is used for a comparison between SPARK and HVACSIM+ in (Sowell and Haves 2001). It shows that the execution time of the same model in SPARK is 15 to 20 times faster than in HVACSIM+. The primary contributor to this acceleration is the graph-theoretic algorithm, which automatically decomposes a large problem into several small subproblems based on the strength of the coupling. Another implementation of an automatic decomposition strategy can be found in Dymola, in which an algorithm called 'tearing' is employed to reduce the size of coupling problems (Wetter et al. 2008). It is shown that Dymola and SPARK have similar performance for the solution of the VAV system mentioned above. HVACSIM+ also employs a decomposition strategy, which reduces the size of the equation system by allowing the state variables to be 'frozen' during the simulation (Park, Clark, and Kelly 1986). In addition, HVACSIM+ employs a superblock-block-unit hierarchical structure (further discussed in Section 3.1) that enables the decomposition of a large coupled problem into several coupled sub-problems. In addition to various grouping strategies in these tools, SPARK, Dymola, JModelica, and OpenModelica are also equipped with sparse solvers, which can improve the efficiency of a simulation significantly.

Solution methods for nonlinear algebraic equations in the above-mentioned simulation tools are summarized in Table 1. The successive substitution method is efficient and reliable for nearly-linear equation system, however, in some cases it requires a starting that is close to the solution, and it can take a larger number of iterations when far from the solution. Variants of Newton's method are frequently used to solve nonlinear algebraic equations. These methods often use an approximation to the Jacobian, something that is not always available and may be prohibitively expensive when problem sizes are large. Secant methods employed by SPARK and Powell's Hybrid (PH) method employed by HVACSIM+

Table 1. Existing tools for dynamic building system simulation and their solution methods.

Tool	Solution Methods	Reference

SPARK	Newton's method;
	Perturbed Newton's method;
	Successive substitution method;
	Secant methods
		(SPARK 2003)

TRNSYS	Successive substitution method Powell's Hybrid method*	(Klein et al. 2017)
HVACSIM+	Powell's Hybrid method	(Park, Clark, and Kelly 1986)
Dymola	Newton's method	(Petzold 1982; Dynasim AB 2004)

JModelica (Modelica)	Newton's method;
	Newton-Krylov methods;
		(Magnusson and Åkesson 2015; Hindmarsh, Serban, and Reynolds 2017b)

OpenModelica (Modelica)	Newton's method;
	Newton-Krylov methods;
		(Hindmarsh, Serban, and Collier 2017a; Open Source Modelica Consortium 2018)

*In the most recent TRNSYS 18, Powell's Hybrid method is deactivated.

are Quasi-Newton (QN) methods that build inexpensive approximations to the Jacobian matrix, which can be difficult if problems are very nonlinear or poorly scaled. A previous study shows that the PH method using a quasi-Newton step is less robust but more efficient than the Levenberg-Marquardt (LM) method using a direct Newton step (Pourarian et al. 2016). Further improvement in computational efficiency can be seen in Newton-Krylov (NK) methods where the explicit form of the Jacobian is not required and algorithms that avoid dense matrix storage are readily available. Often, a good preconditioner is needed to achieve improved performance. Modelica-based tools such as JModelica and OpenModelica are equipped with NK methods provided by SUNDIAL (Hindmarsh, Serban, and Collier 2017a; Hindmarsh, Serban, and Reynolds 2017b), but the performance of these methods in terms of building simulation is not thoroughly discussed, especially in the context of preconditioning.

In general, recently developed numerical algorithms for solving dynamic building system simulation have drawn little attention from the building simulation community. Recent advances in building simulation focus on model formulation, such as building system components libraries, equation-based modelling, co-simulation, model exchange (Wetter 2011; Nouidui, Wetter, and Zuo 2013; Wetter and van Treeck 2017), and advanced integration methods (Bergero et al. 2018). For large-scale simulations, not limited to building simulations, efforts have been made to improve the translation process, such as parallel computing, multi-scale algorithms and sparse solvers (Casella 2015).

In this paper, we present the application of a preconditioned NK method (Pernice and Walker 1998) that solves dynamic building system simulation problems efficiently and robustly. Although the NK method is well known in the mathematical community, it is not well understood or evaluated in the building and HVAC simulation area. To illustrate this point, we performed a thorough literature search from 1986 to 2021, using keyword combinations that were constructed by one phrase in sub-keyword group1 [‘"newton krylov"‘, ‘"inexact newton"‘, ‘"jacobian free"‘, ‘"GMRES"‘, ‘"matrix free"‘] and one phrase in sub-keyword group 2 [‘"hvac"‘, ‘"in building"‘, ‘"building system"‘, ‘"building dynamic system"‘, ‘"air handling unit"‘, ‘"AHU"‘, ‘"vav"‘, ‘"building control"‘, ‘"building simulation"‘, ‘"building performance"‘, ‘"building energy"‘, ‘"modelica"‘, ‘"TRNSYS"‘, ‘"air conditioning"‘, ‘"SPARK"‘] using the SSSS (https://github.com/lz356/SSSS) (Zhang et al. 2021) searching method. In the literatures, NK method is often introduced as a general-purpose solver (Fritzson et al. 2020), while its performance is rarely compared with other methods. The few literatures that describe the application of NK methods or simply Krylov methods are mostly related to solving finite element analysis problems for building envelop simulation (Kośny 2015; Romaní Picas, Gracia Cuesta, and Cabeza 2016), building structure simulation (Pakiding 2016) and aeroacoustics simulation (Kaltenbacher et al. 2016; Borelli and Schenone 2009), and computational fluid dynamics problems for heat exchanger simulation (Hermanns 2018 Pourhoseinian, Asasian-Kolur, and Sharifian 2021;) and natural ventilation simulation (Xu et al. 2021), which often have different numerical challenges from system level HVAC simulation that we focus on in this paper. Only one article mentioned the NK method in comparison with other methods in building system simulation, but its performance is only evaluated on extremely small-scale examples (Cedeño et al. 2018).

HVACSIM+ is chosen as the tool for this study, though similarities among the different tools at the level of the numerical solver allow the conclusions to be applied beyond HVACSIM+. Some common numerical challenges (e.g. non-differentiable or even discontinuous equations), which negatively impact the robustness and stability of a numerical solver, can also be seen in HVACSIM+. In this study, an application of the NK solver package (NITSOL (Pernice and Walker 1998)) within the HVACSIM+ environment and a cost-effective preconditioning technique that appears effective in HVAC system simulation are presented, together with a comparison between the NK method, preconditioned-NK method and PH method on typical HVAC system simulation. The methodology of the NK method is written in detail to be accessible to readers of all levels.

The remainder of the paper is organized as follows. Section 2 introduces a preconditioned NK method. Section 3 compares the preconditioned NK method and PH method using four testbeds that vary in scale and complexity. Section 4 contains conclusions.

2. Methodology

The primary mathematical problem at the heart of all HVAC system simulations is the approximation to the solution of a collection of nonlinear equations (Dennis and Schnabel 1996) that is obtained through the integration of a nonlinear differential algebraic equations (DAE) system (Hindmarsh, Serban, and Collier 2017a). Thus, a method that approximates the solution to Equation (1) is sought,

$$\boldsymbol{F}(\boldsymbol{x}) = 0, \quad \boldsymbol{F}: \mathbb{R}^n \to \mathbb{R}^n, \tag{1}$$

where $\boldsymbol{x}$ is the state vector containing n variables/ unknowns; $\boldsymbol{F}$ is a set of n functions that depend exclusively on $\boldsymbol{x}$ and is assumed to be continuously differentiable everywhere in $\mathbb{R}^n$. Notice that this assumption is not

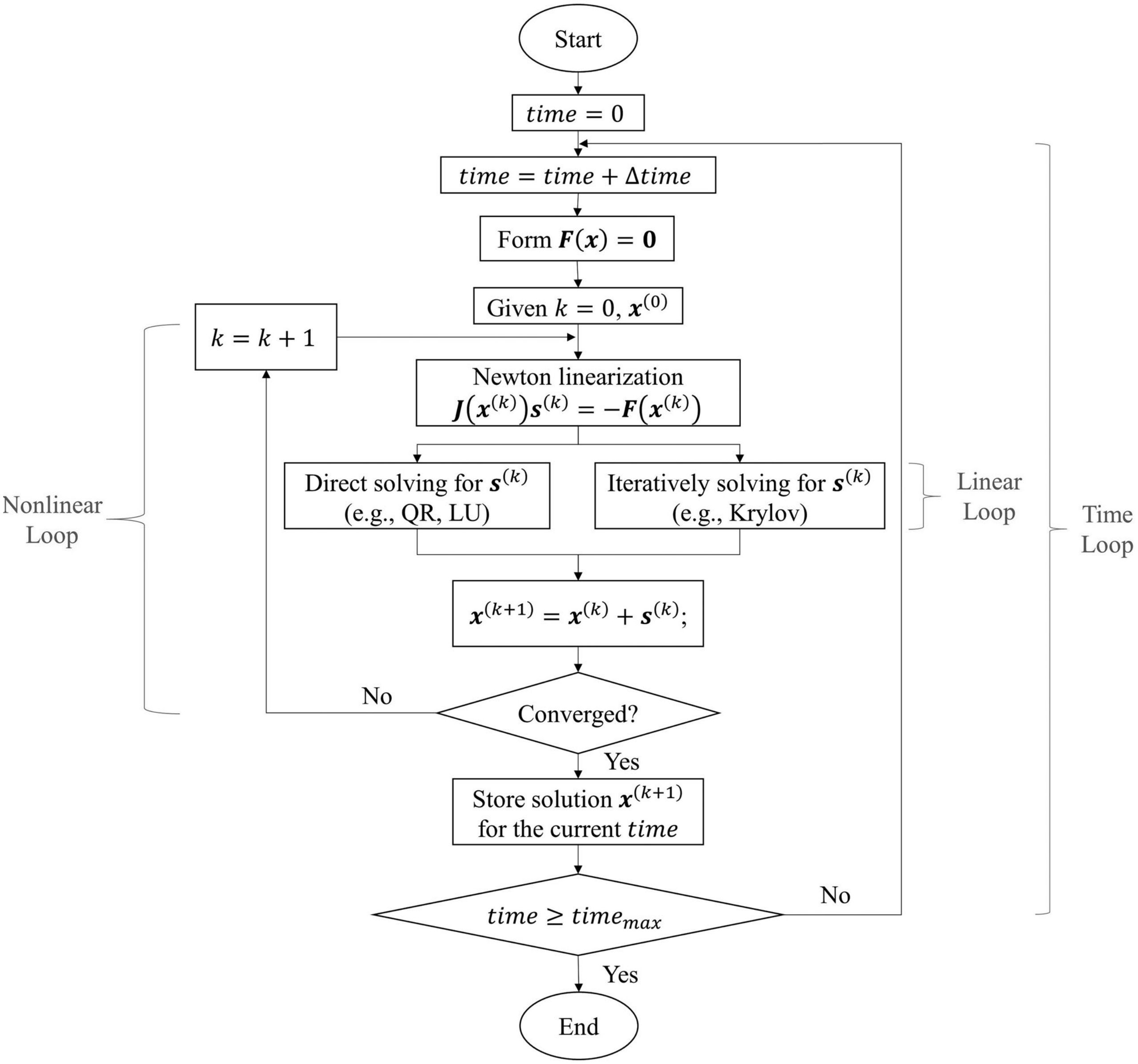
Start
$time = 0$
$time = time + \Delta time$
Form $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$
Given $k = 0$, $\boldsymbol{x}^{(0)}$
$k = k + 1$
Newton linearization
$\boldsymbol{J}(\boldsymbol{x}^{(k)})\boldsymbol{s}^{(k)} = -\boldsymbol{F}(\boldsymbol{x}^{(k)})$
Direct solving for $\boldsymbol{s}^{(k)}$ (e.g., QR, LU)
Iteratively solving for $\boldsymbol{s}^{(k)}$ (e.g., Krylov)
Linear Loop
Nonlinear Loop
Time Loop
$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)}$;
Converged?
No
Yes
Store solution $\boldsymbol{x}^{(k+1)}$ for the current $time$
$time \geq time_{max}$
No
Yes
End

Figure 1. Newton's method for solving nonlinear equation system.

always satisfied. Equations of a building dynamic system model can be non-differentiable or even discontinuous, which results in convergence and stability issues (Wetter 2005). For discontinuity issues, readers can refer to (Wetter 2005; Rädler 2010; Kitchin 2011; Chen 2019).

Newton's method is often used to solve Equation (1). At the *k*-th Newton (nonlinear) iteration, Newton's method solves the following linear system,

$$\boldsymbol{J}(\boldsymbol{x}^{(k)})\boldsymbol{s}^{(k)} = -\boldsymbol{F}(\boldsymbol{x}^{(k)}) \quad (2)$$

where $\boldsymbol{x}^{(k)}$ are the state variables at the *k*-th iteration, $\boldsymbol{J}(\boldsymbol{x}^{(k)}) \equiv \boldsymbol{F}'(\boldsymbol{x}^{(k)})$ is the Jacobian matrix at that iteration, and $\boldsymbol{s}^{(k)}$ is the Newton step that generates the next iteration, $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)}$. For large and sparse nonlinear systems, iteratively solving Equation (2) is preferable because it usually requires much less storage as compared to direct solving (Chan and Jackson 1984) (e.g. LU and QR factorizations (Dennis and Schnabel 1996)). The iteration loops for solving Equation (1) and Equation (2) are often referred to as the nonlinear loop and the linear loop, respectively. The flow chart for solving a nonlinear equation system is shown in Figure 1.

We investigate the application of a NK method from the solver package NITSOL (Pernice and Walker 1998) that solves Equation (1) by solving Equation (2) iteratively using a Krylov subspace method. This method is an **I**nexact **N**ewton method with **B**acktracking (INB) algorithm, where linear systems are solved using **G**eneralized **M**inimal **RES**idual (GMRES) (Saad and Schultz

Table 2. Inexact Newton Method with Backtracking (Pernice and Walker 1998).

```
INB Algorithm
1  Let $x_0, \eta_{max} \in [0,1), t \in (0,1)$, and $0 < \theta_{min} < \theta_{max} < 1$ be given
2  For $k = 0, 1, \ldots$ until converged Do:
3      Choose initial $\eta^{(k)} \in [0, \eta_{max}]$ and $\boldsymbol{s}^{(k)}$ such that
4      $||\boldsymbol{F}(\boldsymbol{x}^{(k)}) + \boldsymbol{J}(\boldsymbol{x}^{(k)})\boldsymbol{s}^{(k)}|| \le \eta^{(k)}||\boldsymbol{F}(\boldsymbol{x}^{(k)})||$
5      While $||\boldsymbol{F}(\boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)})|| > [1 - t(1 - \eta^{(k)})]||F(\boldsymbol{x}^{(k)})||$ Do:
6          Choose $\theta \in [\theta_{min}, \theta_{max}]$
7          Update $\boldsymbol{s}^{(k)} \leftarrow \theta \boldsymbol{s}^{(k)}$ and $\eta^{(k)} \leftarrow 1 - \theta(1 - \eta^{(k)})$
8      Set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)}$
```

1986), a Krylov subspace method. The implementation allows for the application of a preconditioner to accelerate and improve convergence of the iterative method. Details about the INB framework, the GMRES algorithm, and the preconditioning technique are summarized in the following subsections.

2.1. *Inexact Newton method with backtracking*

An approximation to Equation (2) can be assembled where the right-hand side of the equation decreases as iterates draw closer to the solution. In this inexact Newton condition,

$$||\boldsymbol{J}(\boldsymbol{x}^{(k)})\boldsymbol{s}^{(k)} + \boldsymbol{F}(\boldsymbol{x}^{(k)})|| \le \eta^{(k)}||\boldsymbol{F}(\boldsymbol{x}^{(k)})|| \tag{3}$$

the superscript k represents the iteration number, $\eta^{(k)}$ is a 'forcing term' that satisfies $\eta^{(k)} \in [0, 1)$, and $|| \cdot ||$ denotes the L2 norm. If $\eta^{(k)} = 0$, this equation is solved to machine precision. In this study, the default choice of $\eta^{(k)}$ provided by the NITSOL package (Pernice and Walker 1998) as presented in Equation (4) is employed.

$$\eta^{(k)} = \frac{\mathrm{abs}(||\boldsymbol{F}(\boldsymbol{x}^{(k)})|| - ||\boldsymbol{F}(\boldsymbol{x}^{(k-1)}) + \boldsymbol{J}(\boldsymbol{x}^{(k-1)})\boldsymbol{s}^{(k-1)}||)}{||\boldsymbol{F}(\boldsymbol{x}^{(k-1)})||} \tag{4}$$

After solving Equation (3) to obtain an initial $\boldsymbol{s}^{(k)}$, a backtracking line-search method is used to search for a minimum. The INB algorithm given by (Pernice and Walker 1998) is summarized in Table 2, where lines 3 and 4 are associated with the inexact Newton method, and lines 5, 6 and 7 are associated with the backtracking method.

2.2. *Generalized minimal residual (GMRES)*

In this subsection, a summary of the GMRES method used to solve Equation (2) is presented. A linear system $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A}$ is an $n \times n$ non-singular real-valued square matrix, $\boldsymbol{b}$ is an $n \times 1$ real-valued vector and $\boldsymbol{x}$ is the solution, can be approximated (Krylov 1931) by,

$$\boldsymbol{x}_* = \boldsymbol{A}^{-1}\boldsymbol{b} \cong \beta_1\boldsymbol{b} + \beta_2\boldsymbol{Ab} + \beta_3\boldsymbol{A}^2\boldsymbol{b} + \ldots + \beta_j\boldsymbol{A}^{j-1}\boldsymbol{b} \tag{5}$$

where $\beta_1, \beta_2, \beta_3, \ldots \beta_j$ are scalar unknowns and j is a natural number less than or equal to the system dimension n. Equation (5) suggests that approximated solutions belong to the subspace spanned by the vectors $(\boldsymbol{b}, \boldsymbol{Ab}, \boldsymbol{A}^2\boldsymbol{b}, \ldots \boldsymbol{A}^{j-1}\boldsymbol{b})$, called the Krylov subspace. A Krylov subspace with dimension j is denoted as $\mathcal{K}_j(\boldsymbol{A}, \boldsymbol{b}) \equiv \mathrm{span}(\boldsymbol{b}, \boldsymbol{Ab}, \boldsymbol{A}^2\boldsymbol{b}, \ldots \boldsymbol{A}^{j-1}\boldsymbol{b})$. Instead of finding a solution $\boldsymbol{x}$ in the entire real space $\mathbb{R}^n$, a Krylov method finds an approximate solution restricted to $\mathcal{K}_j(\boldsymbol{A}, \boldsymbol{b})$ by finding $\boldsymbol{\beta} = [\beta_1, \beta_2, \beta_3, \ldots \beta_j]^T$. Alternatively, given an initial value $\boldsymbol{x}_{(0)}$ to the linear system $\boldsymbol{Ax} = \boldsymbol{b}$, and an initial residual $\boldsymbol{r}_{(0)} = \boldsymbol{b} - \boldsymbol{Ax}_{(0)}$, Equation (5) can be rewritten as,

$$\begin{aligned} \boldsymbol{x}_* = \boldsymbol{A}^{-1}\boldsymbol{b} = \boldsymbol{x}_{(0)} + \boldsymbol{A}^{-1}\boldsymbol{r}_{(0)} \cong \boldsymbol{x}_{(0)} + z_1\boldsymbol{r}_{(0)} + z_2\boldsymbol{Ar}_{(0)} \\ + z_3\boldsymbol{A}^2\boldsymbol{r}_{(0)} + \ldots + z_j\boldsymbol{A}^{j-1}\boldsymbol{r}_{(0)} \end{aligned} \tag{6}$$

where the subscript in parentheses represents the linear iteration number, and, just as $\beta_1, \beta_2, \beta_3, \ldots \beta_j$ determines a vector in $\mathcal{K}_j(\boldsymbol{A}, \boldsymbol{b})$, $z_1, z_2, z_3, \ldots z_j$ determines a vector in $\mathcal{K}_j(\boldsymbol{A}, \boldsymbol{r}_{(0)})$. Equation (6) suggests that the approximated solution of $\boldsymbol{Ax} = \boldsymbol{b}$ can be found in a Krylov subspace, i.e. $\boldsymbol{x}_{(0)} + \mathcal{K}_j(\boldsymbol{A}, \boldsymbol{r}_{(0)})$, where$\mathcal{K}_j(\boldsymbol{A}, \boldsymbol{r}_{(0)}) \equiv \mathrm{span}(\boldsymbol{r}_{(0)}, \boldsymbol{Ar}_{(0)}, \boldsymbol{A}^2\boldsymbol{r}_{(0)}, \ldots \boldsymbol{A}^{j-1}\boldsymbol{r}_{(0)})$. The following illustration of the GMRES method is based on Equation (6).

At each iteration, the GMRES method finds the best solution in the j dimensional Krylov subspace, $\boldsymbol{x}_{(0)} + \mathcal{K}_j(\boldsymbol{A}, \boldsymbol{r}_{(0)})$, by minimizing the residual of the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ in a least squares sense, i.e.

$$\min_{\boldsymbol{x} \in \boldsymbol{x}_{(0)} + \mathcal{K}_j(\boldsymbol{A}, \boldsymbol{r}_{(0)})} ||\boldsymbol{b} - \boldsymbol{Ax}|| \tag{7}$$

where $\boldsymbol{x}$ is to be found in the Krylov subspace $\boldsymbol{x}_{(0)} + \mathcal{K}_j(\boldsymbol{A}, \boldsymbol{r}_{(0)})$.

Specifically, the least squares problem in Equation (7) is solved by constructing an orthonormal basis $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_j\}$ for $\mathcal{K}_j(\boldsymbol{A}, \boldsymbol{r}_{(0)})$ using an Arnoldi iteration[1] (Arnoldi 1951; Sorensen 1992). Starting with a normalized vector $\boldsymbol{q}_1 = \boldsymbol{r}_{(0)}/||\boldsymbol{r}_{(0)}||$ as a basis for $\mathcal{K}_j(\boldsymbol{A}, \boldsymbol{r}_{(0)})$, Arnoldi iteration constructs an orthonormal basis for $\mathcal{K}_{j+1}(\boldsymbol{A}, \boldsymbol{r}_{(0)})$ from an orthonormal basis for the previous Krylov subspace $\mathcal{K}_j(\boldsymbol{A}, \boldsymbol{r}_{(0)})$. That is,

$$\boldsymbol{q}_{j+1} = \frac{\boldsymbol{Aq}_j - (h_{1j}\boldsymbol{q}_1 + h_{2j}\boldsymbol{q}_2 + \ldots + h_{jj}\boldsymbol{q}_j)}{||\boldsymbol{Aq}_j - (h_{1j}\boldsymbol{q}_1 + h_{2j}\boldsymbol{q}_2 + \ldots + h_{jj}\boldsymbol{q}_j)||} \tag{8}$$

where $h_{ij} = \boldsymbol{q}_i^T\boldsymbol{Aq}_j$.

Let $\boldsymbol{Q}_j$ be defined as a Krylov matrix as presented in Equation (9).

$$\boldsymbol{Q}_j = [\ \boldsymbol{q}_1 \mid \boldsymbol{q}_2 \mid \boldsymbol{q}_3 \mid \ldots \mid \boldsymbol{q}_j\ ]. \tag{9}$$

Through the Arnoldi iteration, there is a relationship between $\boldsymbol{Q}_j$ and $\boldsymbol{Q}_{j+1}$,

$$\boldsymbol{AQ}_j = \boldsymbol{Q}_{j+1}\tilde{\boldsymbol{H}}_j \tag{10}$$

where $\tilde{\boldsymbol{H}}_j$ is a $(j+1) \times j$ matrix with a $j \times j$ upper Hessenberg matrix[2] and an additional $(j+1)$ row with a nonzero entry at the end.

With the Krylov matrix in Equation (9), the least squares problem in Equation (7) can be rewritten as,

$$\min_{\boldsymbol{z} \in \mathbb{R}^j} ||\boldsymbol{b} - \boldsymbol{A}(\boldsymbol{x}_{(0)} + \boldsymbol{Q}_j\boldsymbol{z})|| = \min_{\boldsymbol{z} \in \mathbb{R}^j} ||\boldsymbol{r}_{(0)} - \boldsymbol{A}\boldsymbol{Q}_j\boldsymbol{z}|| \qquad (11)$$

where $\boldsymbol{z} = [z_1, z_2, z_3, \ldots z_j]^T$ is found in the real space $\mathbb{R}^j$. Then, with Equation (10), the least squares problem is further transformed to,

$$\min_{\boldsymbol{z} \in \mathbb{R}^j} ||\boldsymbol{r}_{(0)} - \boldsymbol{Q}_{j+1}\tilde{\boldsymbol{H}}_j\boldsymbol{z}|| = \min_{\boldsymbol{z} \in \mathbb{R}^j} ||\boldsymbol{Q}_{j+1}^T\boldsymbol{r}_{(0)} - \tilde{\boldsymbol{H}}_j\boldsymbol{z}||$$

$$= \min_{\boldsymbol{z} \in \mathbb{R}^j} ||||\boldsymbol{r}_{(0)}||\boldsymbol{e}_1 - \tilde{\boldsymbol{H}}_j\boldsymbol{z}|| \qquad (12)$$

where $\boldsymbol{e}_1$ is the first column of a $j \times j$ identity matrix. After solving for $\boldsymbol{z}$ in Equation (12), the approximated solution of the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ at the j-th iteration is obtained as $\boldsymbol{x}_{(j)} = \boldsymbol{x}_{(0)} + \boldsymbol{Q}_j\boldsymbol{z}$.

The memory requirement for storing the orthonormal basis $\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_j\}$ for the Krylov subspace grows larger as the number of iterations increases, and it can be controlled using a restarting parameter 'm'; this is often referred to as GMRES(m) (Saad and Schultz 1986). In simple terms, the GMRES algorithm will be restarted every 'm' linear iterations, and the solution of the current m-th iteration will serve as an initial guess to the next 'm' iterations. Picking an appropriate restarting parameter 'm' is important. An 'm' that is very small may cause stagnation in the linear iteration, while a large 'm' requires more memory. For a well-conditioned linear system, the solution of Equation (3) typically converges within a few linear iterations. In this study, we use the default choice provided by the NITSOL package, m = 20.

NITSOL employs the SGMRES(m) algorithm, a simplified variant of GMRES(m) proposed in (Walker and Zhou 1994). SGMRES(m) begins with $\boldsymbol{q}_1 = \boldsymbol{A}\boldsymbol{r}_{(0)}/||\boldsymbol{A}\boldsymbol{r}_{(0)}||$ instead of $\boldsymbol{q}_1 = \boldsymbol{r}_{(0)}/||\boldsymbol{r}_{(0)}||$, so the upper Hessenberg system in Equation (12) is reduced to an upper triangular system. The computational cost of SGMRES(m) is less than that of GMRES(m).

SGMRES(m) (Walker and Zhou 1994) for solving Equation (2), $\boldsymbol{Js} = -\boldsymbol{F}(x^{(k)})$, is presented in Table 3. The Jacobian matrix $\boldsymbol{J}$ is never formed explicitly, instead, only Jacobian-vector products $\boldsymbol{Jv}$ are approximated**.** They can be approximated using a finite difference direction derivative method, with a cost of one function evaluation (i.e. evaluating $\boldsymbol{F}$ once). This is one reason why Krylov subspaces are commonly used to solve large sparse linear systems.

Table 3. SGMRES(m) algorithm (Walker and Zhou 1994).

```
SGMRES(m) Algorithm
INITIALIZE: Given $\boldsymbol{s}$, set $\boldsymbol{r} \equiv -\boldsymbol{F} - \boldsymbol{Js}$ and $\rho_0 \equiv ||\boldsymbol{r}||$.
    TERMINATION: If $\rho_0 \leq tol$, accept $\boldsymbol{s}$ and exist; otherwise,
    update $\boldsymbol{r} \leftarrow \boldsymbol{r}/\rho_0$ and set $\rho = 1$
ITERATE: For $j = 1, 2, \ldots, m$, Do:
    Evaluate $\boldsymbol{v}_j \equiv \boldsymbol{J}\boldsymbol{v}_{j-1} (\boldsymbol{v}_1 \equiv \boldsymbol{Jr})$
    If $j > 1$, then For $i = 1, 2, \ldots, j-1$, Do:
        Set $\rho_{i,j} \equiv \boldsymbol{v}_i^T\boldsymbol{v}_j$
        Update $\boldsymbol{v}_j \leftarrow \boldsymbol{v}_j - \rho_{i,j}\boldsymbol{v}_i$
    Set $\rho_{j,j} \equiv ||\boldsymbol{v}_j||$; update $\boldsymbol{v}_j \leftarrow \boldsymbol{v}_j/\rho_{j,j}$
    Set $\boldsymbol{R}_j \equiv \begin{pmatrix} & \rho_{1,j} \\ \boldsymbol{R}_{j-1} & \vdots \\ & \vdots \\ 0 \ldots 0 & \rho_{j,j} \end{pmatrix} (\boldsymbol{R}_1 \equiv (\rho_{1,1}))$
    Set $\xi_j \equiv \boldsymbol{r}^T\boldsymbol{v}_j$; Update $\rho \leftarrow \rho \sin(\cos^{-1}(\xi_j/\rho))$
    TERMINATION: If $\rho \cdot \rho_0 \leq tol$, Go to SOLVE
    Update $\boldsymbol{r} \leftarrow \boldsymbol{r} - \xi_j\boldsymbol{v}_j$
SOLVE: Let $j$ be the final iteration number from ITERATE
    Solve $\boldsymbol{R}_j\boldsymbol{y} = (\xi_1, \xi_2, \ldots, \xi_j)^T$ For $\boldsymbol{y} = (y_1, y_2, \ldots, y_j)^T$
    Form $\boldsymbol{z} = \begin{cases} y_1 r, & \text{if } j = 1 \\ y_1 r + \sum_{i=1}^{j-1} (y_{i+1} + y_1\xi_i)\boldsymbol{v}_i, & \text{if } j > 1 \end{cases}$
    Update $\boldsymbol{s} \leftarrow \boldsymbol{s} + \rho_0\boldsymbol{z}$
    TERMINATION: If $\rho \cdot \rho_0 \leq tol$, accept $\boldsymbol{s}$ and exit; otherwise,
    update $\boldsymbol{r} \leftarrow (\boldsymbol{r} - \xi_j\boldsymbol{v}_j)/\rho$, $\rho_0 \leftarrow \rho \cdot \rho_0$, $\rho \leftarrow 1$ and return to
    ITERATE.
```

2.3. Preconditioning

In this study, only the right-preconditioning technique is employed, such that Equation (2) can be rewritten,

$$\boldsymbol{J}\boldsymbol{M}^{-1}\boldsymbol{S} = -\boldsymbol{F}, \; \boldsymbol{S} \equiv \boldsymbol{Ms} \qquad (13)$$

where $\boldsymbol{M}$ is an n by n preconditioner. Finding an effective and less expensive preconditioner (i.e. $\boldsymbol{M} \cong \boldsymbol{J}$) whose inverse is easily determined requires significant effort and expert knowledge. To balance the cost and effectiveness of preconditioning, an ideal preconditioner (i.e. $\boldsymbol{M}^{-1} = \boldsymbol{J}^{-1}$) is computed through matrix factorization and applied to the linear system of the current iteration and all the following nonlinear iterations until it is no longer effective. In other words, a new preconditioner does not have to be calculated at each iteration and time step. An update of the preconditioner is triggered when convergence difficulties arise.

The major computational costs introduced by the use of the ideal preconditioner in this study include the forward difference approximation of the explicit form of the Jacobian at the preconditioner update, calculation of the inverse of the Jacobian (by LU factorization) at the

Table 4. INB-PSGMRES(m) algorithm in a simulation over time.

```
INB-PSGMRES(m) Algorithm
Let $\boldsymbol{X}_0$, $tstep = 1$, and $tstep_{end}$ be given,
Initialize $\boldsymbol{M}^{-1} = \boldsymbol{J}^{-1}(\boldsymbol{X}_0)$
While $tstep \leq tstep_{end}$ Do:
    Set $\boldsymbol{x}^{(0)} = \boldsymbol{X}_{tstep-1}$
    Update $\boldsymbol{M}^{-1} = \boldsymbol{J}^{-1}(\boldsymbol{x}^{(0)})$ if $iterm = = 1, 5, 6,$ or $7$
    Let $\eta_{max} \in [0, 1), t \in (0, 1), 0 < \theta_{min} < \theta_{max} < 1,$
    For $k = 0, 1, \ldots$ until converged Do:
      Update $\boldsymbol{M}^{-1} = \boldsymbol{J}^{-1}(\boldsymbol{x}^{(k)})$ if $itrmks = = 3, 4,$ or $5$
      Given $\eta^{(k)}$; Solve $||\boldsymbol{F}(\boldsymbol{x}^{(k)}) + \boldsymbol{J}(\boldsymbol{x}^{(k)})\boldsymbol{M}^{-1}\boldsymbol{S}|| \leq$
      $\eta^{(k)}||\boldsymbol{F}(\boldsymbol{x}^{(k)})||$ using Algorithm SGMRES(m) from Table 3;
      output $itrmks$
      Set $\boldsymbol{s}^{(k)} = \boldsymbol{M}^{-1}\boldsymbol{S}$
      While $||\boldsymbol{F}(\boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)})|| > [1 - t(1 - \eta^{(k)})]||F(\boldsymbol{x}^{(\boldsymbol{k})})||$ Do:
        Choose $\theta \in [\theta_{min}, \theta_{max}]$
        Update $\boldsymbol{s}^{(k)} \leftarrow \theta\boldsymbol{s}^{(k)}$ and $\eta^{(k)} \leftarrow 1 - \theta(1 - \eta^{(k)})$
      Set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)}$; output $iterm$
    Set $\boldsymbol{X}_{tstep} = \boldsymbol{x}^{(k+1)}$
```

preconditioner update, an additional matrix-vector multiplication at every linear iteration[3], and an additional matrix-vector multiplication at every nonlinear iteration[4]. Similar schemes that amortize the cost of preconditioner construction over a few Newton iterations or a period of time have been documented in the literature (Knoll and McHugh 1998 Pernice and Walker 1998; Benzi 2002; Hindmarsh, Serban, and Collier 2017a;), but the effectiveness varies.

The preconditioning technique presented here is very effective when the preconditioner does not need to be updated frequently. The algorithm of the NK method with preconditioning (i.e. **I**nexact **N**ewton with **B**acktracking and **P**reconditioned **SGMRES(m)** : INB-PSGMRES(m)) is presented in Table 4. In this table, *iterm* and *itrmks* are NITSOL's termination flags for the nonlinear loop and linear loop, respectively, which indicate convergence difficulties.

From preliminary testing of various scaling methods (Chen et al. 2017), it is found that the simulation performance of the problem we test in this study is insensitive to the scale of variable space. Therefore, the impact of scaling has not been considered in the test cases. Yet, for general applications, a scaled PSGMRES(m) method that solves $\boldsymbol{J}(\boldsymbol{D}^{-1}\boldsymbol{M}^{-1})(\boldsymbol{MDs}) = -\boldsymbol{F}$ instead of Equation (13) may be considered, where $\boldsymbol{D}$ is the scaling matrix. And $||\boldsymbol{Ds}||$, instead of $||\boldsymbol{s}||$, shall be evaluated when judging for convergence. In HVACSIM+, $\boldsymbol{D}$ is constructed as a diagonal matrix (stored as a vector in the computer) at the beginning of each time step, using the reciprocal of the state variables' initial values (which is also the solution of the previous time step).

3. Numerical experiments

3.1. *Introduction of the testbeds*

The testbeds used in this study were developed in the HVACSIM+ environment. Developed by the U.S. National Institute of Standards and Technology (NIST), HVACSIM+ is a dynamic component-based simulation tool and computational environment (Park, Clark, and Kelly 1986) that includes a collection of subroutines in three categories: pre-processing, simulation, and post-processing. During the pre-processing stage, a simulation work file is created by use of the interactive front-end program of HVACSIM+. The simulation work file is then converted to a model definition file to make the model information readable by the main simulation program, MODSIM.

HVACSIM+ employs a hierarchical structure, illustrated in Figure 2. In this structure, a UNIT is an individual instance of a generic component model representing a specific component in the system. Each UNIT is a TYPE that contains equations for a component (e.g. a component that calculates the inlet pressure for a fluid resistance given the outlet pressure and the inlet mass flow rate). Users link UNITs together through their inputs and outputs to represent their relationships in the real physical system. Closely coupled UNITs are grouped by the user into blocks for simultaneous solution. Blocks are then grouped into a superblock for simultaneous solution. State variables in different superblocks are weakly coupled, and each superblock is treated as an independent subsystem of the overall simulation (Park, Clark, and Kelly 1986).

The simultaneous equation system resulting from input/output connections and discretization of differential equations in each subsystem, as presented in Equation (1), will be solved using HVSCSIM+'s default solver using the PH method (Powell 1970) or the NITSOL solver with the INB-PSGMRES(m) method presented in Section 2.

Four testbeds, 4Z5B (4 zones, 5 blocks), 4Z1B (4 zones, 1 block), 12Z5B (12 zones, 5 blocks), and 40Z5B (40 zones, 5 blocks) are used to compare the performance of the two solution methods. 4Z5B and 4Z1B are based on a small commercial building simulation model developed as part of ASHRAE Research Project 1312 (Wen and Li 2011). This model simulates a single duct, dual fan VAV system serving four building zones: west-facing, south-facing, east-facing, and internal spaces. Each superblock in the two testbeds is constituted by one block in which equations are solved simultaneously. The 12Z5B and 40Z5B testbeds are expansions of the 4Z5B testbed with two more zones and eight more zones on each orientation, respectively. Their zones are identical to the zones of the 4Z5B testbed

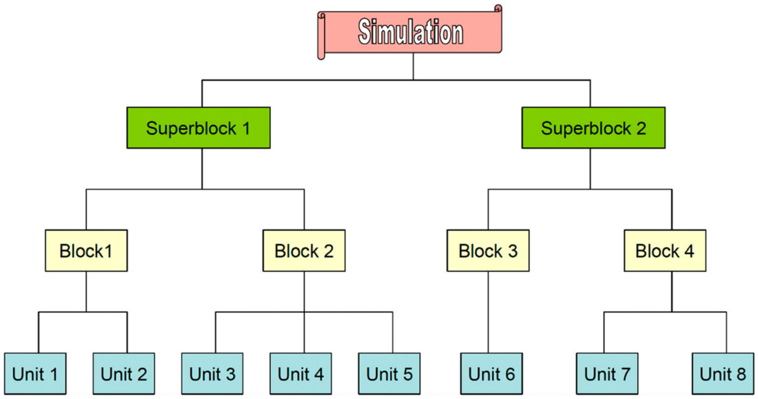

Figure 2. Hierarchical simulation setup.

on the same orientation. Their systems' parameters have been redesigned according to manufacturer catalogues to handle larger airflow rates.

In the 4Z5B, 12Z5B, and 40Z5B testbeds, components are grouped into five superblocks as follows: control, actuator, airflow (i.e. equations related to mass flow and pressure), thermal (i.e. equations related to heat and moisture transfer), and sensor. This formulation imposes a weak coupling among equations from different subsystems. At each time step, equations within each superblock (or block) are solved simultaneously. The solution of each superblock is provided to the superblocks that numerically follow the solved superblock (e.g. the actuator superblock is solved after the control superblock). The sensor superblock is computed last in a time step *t*, and captures the values of time step *t*, such as the temperatures and the humidity ratio, which are computed earlier in the same time step by the thermal superblock. These values will be passed to the control superblock at the next time step, $t + 1$, to determine whether adjustments of damper and valve positions are needed. Then, at time step $t + 1$, based on the determination of the control superblock, command signals are passed to the actuator superblock within the same time step to move dampers and valves. In the same time step $t + 1$, mass-pressure equations in the airflow superblock and then energy balance equations in the thermal superblock are solved simultaneously for a new solution based on the new damper and valve positions. Among these superblocks, the three superblocks that contain variables that require simultaneous solution are the control superblock, airflow superblock and thermal superblock.

The 4Z1B testbed groups all components into one superblock, instead of having components grouped into five different superblocks. The purpose of this testbed is to present the solvers with a different challenge as equations with different kinds of variables are strongly coupled.

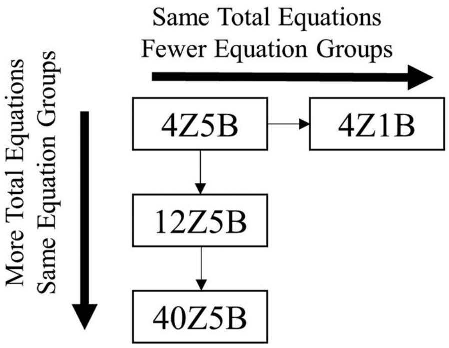

Figure 3. Relationship between the testbeds.

Table 5. Number of Simultaneous Variables in Each Superblock of the Four Testbeds.

Testbed	Control	Airflow	Thermal
4Z5B	7	37	48
12Z5B	7	109	124
40Z5B	7	361	383
4Z1B	140 (one superblock)		

The relationship among these testbeds is presented in Figure 3. The total number of simultaneous variables to be solved in each superblock is summarized in Table 5. For more details about these testbeds, please refer to (Chen 2019). Although the four testbeds to be used in the numerical experiment are not yet in the range of large-scale systems, the superior performance of INB-PSGMRES(m) as the problem size and complexity inevitably grow larger will still be apparent. Superior performance as a function of problem size is easily demonstrated in the four presented testbeds.

Table 6. Weather scenarios.

Weather Scenario	Location	Min. Temp. (Coincident RH)	Max. Temp. (Coincident RH)	System Operation	Comment
1	Ankeny, IA	23.6° C (92%)	40.3 °C (43%)	Cooling (MOA)	Significant cooling coil operation; partially wet or completely wet coil condition
2	Ankeny, IA	19.2 °C (93%)	24.1 °C (73%)	Cooling (MOA)	Moderate cooling coil operation; partially wet or completely wet coil condition
3	Ankeny, IA	-17.5 °C (13%)	-14.5 °C (21%)	Heating (MOA)	Significant heating coil operation; dry coil condition
4	Ankeny, IA	4.9 °C (92%)	14.5 °C (34%)	Cooling (ECO)	Frequent outdoor air damper movement; limited cooling coil operation;

Note: 1) Temp.: Temperature; 2) RH: Relative Humidity; 3) MOA: Minimum outdoor air damper; 4) ECO: Economizer

3.2. Test settings

In this study, the NITSOL default restarting parameter m = 20 is used, INB-PSGMRES(20). The performance of INB-PSGMRES(20) is investigated under four different weather scenarios. Descriptions of the four weather scenarios, as well as why these scenarios are chosen, are summarized in Table 6. In general, these weather scenarios are chosen so that the air and thermal systems would experience different challenges and different operating conditions.

The duration of each simulation is 24 hours, and the simulation time step is fixed at 2.5 seconds. During the 24-hour simulation period, the AHU-VAV system operates from 6:00 to 18:00. Thus, there are a total of 34,560 time steps during a whole day simulation, and 17,280 time steps during the operating period.

For both methods, the solution at each time step is accepted when one of the criteria summarized in Table 7 is satisfied. In this table, $\boldsymbol{x}^{(k+1)}$ and $\boldsymbol{x}^{(k)}$ are the solution of the $(k+1)$-th and k-th iterations, respectively; ϵ_{mach} is the machine precision, equal to 1.19×10^{-7} in our case, i.e. single precision; NFE is the number of function evaluations; n is the number of simultaneous variables to be solved in a superblock (see Table 5), and NNI is the number of nonlinear iterations. The solution of a time step is considered to have converged if the residual test, i.e. $||\boldsymbol{F}|| \leq ftol$, or the successive iterate test, i.e. $||\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}|| \leq xtol$, is satisfied. However, it is difficult to determine a step tolerance, $xtol$, in this study, because two very different step size control strategies, i.e. trust-region and backtracking, are employed by PH and INB-PSGMRES(20), respectively. Specifically, if $xtol$ is set relatively large, around 10^{-3} or 10^{-4}, INB-PSGMRE(20) will accept the solution early because the backtracking algorithm iteratively decreases the step size until an acceptable step is found in a single nonlinear iteration, while PH maintains a constant step size in a single nonlinear iteration. Therefore, only the residual test is used to judge convergence. The step tolerance is set to the machine precision (i.e. an extreme case). If the solution of a time step satisfies the step tolerance first, it is considered not converged, and the preconditioner update will be triggered.

Table 7. Termination criteria for each time step.

Condition	Criteria	User Specified Value
PH:		
Converged	$\lVert\boldsymbol{F}\rVert \leq ftol$	$ftol = 5 \times 10^{-5}$
Not Converged	$\lVert\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\rVert \leq xtol$	$xtol = \epsilon_{mach}$
	$\text{NFE} \geq \text{NFE}_{\max}$	$\text{NFE}_{\max} = 200 \times (n+1)$
INB-PSGMRES(20):		
Converged	$\lVert\boldsymbol{F}\rVert \leq ftol$	$ftol = 5 \times 10^{-5}$
Not Converged	$\lVert\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\rVert \leq xtol$ ($iterm = 7$)	$xtol = \epsilon_{mach}$
	$\text{NNI} \geq \text{NNI}_{\max}$ ($iterm = 1$)	$\text{NNI}_{\max} = 200$
	insufficient initial model norm reduction for adequate progress ($iterm = 5$)	Not Available
	failure to reach an acceptable step through backtracking ($iterm = 6$)	Not Available

3.3. Results and discussions

In this section, the performance indices used for evaluating the performance are introduced. Next, the overall performance of the NK method without preconditioning is presented. Finally, a detailed performance comparison between the preconditioned version described in Section 3.2, i.e. INB-PSGMRES(20), and the default PH method provided by HVACSIM+ is presented.

3.3.1. Performance indices

The following performance indices are used when evaluating the performance of a method.

Successful/Unsuccessful termination:

A simulation that terminates at the user-specified end time is considered to be successfully terminated. Otherwise, it is considered to be unsuccessfully terminated. This is an index for evaluating the robustness of the simulation.

CPU time:
The CPU time is the amount of time consumed by processing a segment of the computer program. It is used for evaluating the efficiency of the simulation. Notice that the CPU time is only a representative of the algorithm efficiency performance, and it is not as thorough as the computational complexity of the algorithm.
Number of Function Evaluations (NFE) per time step:
The NFE per time step is the total NFE divided by the total number of time steps. A one-day simulation with a 2.5 second step size contains 34,560 time steps. One function evaluation means evaluating $\boldsymbol{F}(\boldsymbol{x})$ once with a given $\boldsymbol{x}$. It is used for evaluating the efficiency of the simulation.
Number of Non-Converged-solutions (NNC):
The accepted solution of a time step can be either converged or not converged upon termination of the time step according to Table 7. NNC is used to evaluate the accuracy and robustness of the simulation. In terms of accuracy, a converged-solution is considered more accurate than a non-converged-solution. NNC is also a measure of the robustness of the solution. A larger value for NNC indicates a lack of robustness because continuously accepting non-converged-solutions may result in early termination of the simulation.
Number of Preconditioner Updates (NPU):
NPU reflects the total cost of preconditioner computations in a simulation. In this study, the preconditioner is only updated when convergence difficulties appear in either the nonlinear or linear iteration loop. Therefore, it is also a good indicator for the robustness of the simulation.

3.3.2. Overall performance without preconditioning

As discussed earlier, the performance of the NK method depends heavily on the conditioning of the Jacobian matrix. Without preconditioning, the nonlinear system is typically ill-conditioned, especially for large problems. The overall performance of the NK method without preconditioning i.e. INB-SGMRES(m) without the letter ‘P’, on the four testbeds are summarized in Table 8. In addition to testing with the default value of the restarting parameter, m = 20, larger values are tested because ill-conditioned systems require more linear iterations to reduce the residual. In general, the INB-SGMRES(m) method ends in early termination of the simulation. In some cases, this simulation failure is removed by varying the restarting parameter, but there is insufficient evidence to show that m is a critical factor. An effective preconditioning technique can be used to overcome this difficulty.

Table 8. Overall Performance of INB-SGMRES(m).

		Successful Termination?			
		Weather Scenario			
Testbed	Restarting Parameter: m	1	2	3	4
4Z5B	20	✓	✓	✓	✓
4Z1B	20				
	46		✓		
	1000		✓		
12Z5B	20				
	40			✓	
	1000				✓
40Z5B	20				
	128	✓		✓	✓
	1000				

3.3.3. Comparison between INB-PSGMRES(20) and PH

Simulation of the four testbeds using either INB-PSGMRES(20) or PH terminates successfully. The efficiency of both methods is evaluated by CPU time, which is calculated as the average run time of repeated runs on a personal laptop. The CPU time required to solve each superblock (i.e. control, airflow, and thermal) using both methods is compared and presented in Figure 4. In this figure, the CPU time of the control superblock is presented only for the 4Z5B testbed, because the size (i.e. number of simultaneous variables) of the control superblock is the same for all testbeds with the five block structure (i.e. 4Z5B, 12Z5B, and 40Z5B). According to the figure, INB-PSGMRES(20) is more efficient than PH even for a small size problem, and its superiority becomes more pronounced as the problem size increases. Based on the computation time trend, using INB-PSGMRES(20) to solve large scale problems shows promise. Overall, INB-PSGMRES(20)’s total CPU consumption for running a one-day simulation is at least 50%, 92%, 89%, and 97% less than the PH method for the 4Z5B, 4Z1B, 12Z5B, and 40Z5B testbeds, respectively.

The primary contributor to the reduction of CPU time is the reduction of the number of function evaluations (NFE) in INB-PSGMRES(20), as presented in Figure 5. It is shown that NFE per time step increases significantly with the problem size using PH, while using INB-PSGMRES(20), NFE per time step is 10.1 at most.

The number of non-converged-solutions (NNC), which indicates the accuracy and the robustness of a method, is presented in Figure 6. The lower the NNC, the more accurate and robust the solution. In most cases, the NNC of INB-PSGMRES(20) is greater than or equal to the NNC of PH. Greater NNC while using INB-PSGMRES(20) is due to a ‘delay’ in the preconditioner update scheme. The preconditioner is updated if the previous time step or the previous nonlinear iteration shows convergence difficulties. Updating the preconditioner at the current time step will not prevent the convergence difficulties from happening at the previous time step. Although the NNC has increased with the use of INB-PSGMRES(20), in most cases the increase is insignificant relative to the total of

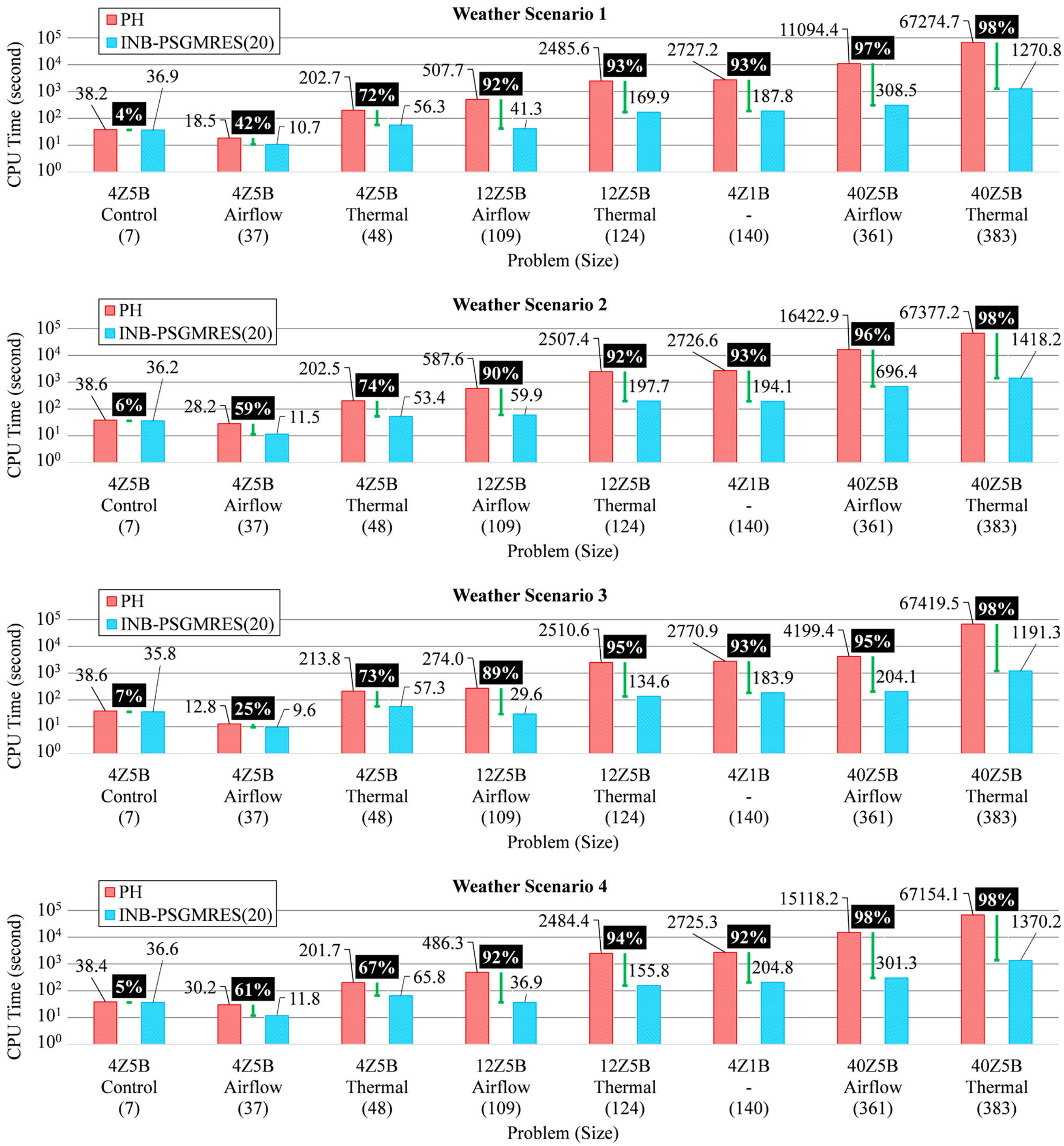

Figure 4. CPU time comparison between PH and INB-PSGMRES(20).

34,560 solutions (1 solution for 1 time step) for a 24-hour simulation. The number of preconditioner updates (NPU) for each case using INB-PSGMRES(20) is presented side by side with the NNC in Figure 7. Whenever there is a non-converged-solution within either nonlinear or linear iteration, the preconditioner update scheme sees it as an ill-conditioning warning that triggers the preconditioner update. Therefore, the NPU is equal to or larger than the NNC for all cases using INB-PSGMRES(20). In the airflow superblock, the preconditioner update usually appears at the beginning (within the first few time steps) of the simulation, the start-up period (around 6:00), and the shut-down period (around 18:00) of the AHU-VAV system. According to the operating scheme described in Section 3.2, the simulation starts with the nonoperating mode, in which the values of airflow rates and pressures in the AHU-VAV system are typically close to zero. The corresponding nonlinear equation system is potentially

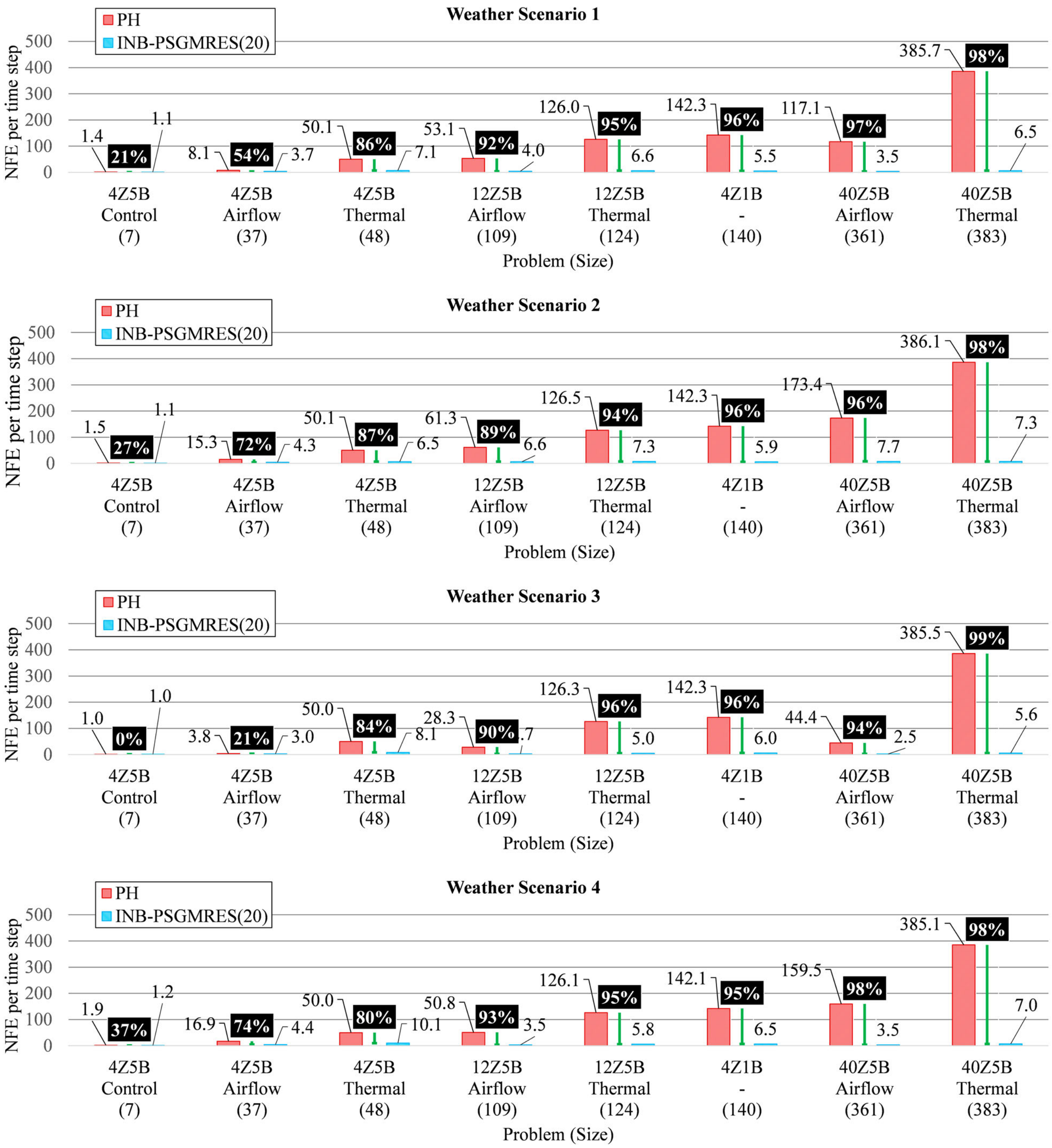

Figure 5. Comparison of NFE per Time Step between PH and INB-PSGMRES(20).

ill-conditioned due to finite precision computation with small numbers. In this experiment, an ideal preconditioner is employed. Once a preconditioner is computed at the beginning of the simulation, it can be reused in the nonoperating hours until the start-up of the AHU-VAV system. For the nonoperating period, a diagonal scaling matrix is as effective as the ideal preconditioner, as it improves the conditioning of the nonlinear system by moving values of the Jacobian entries away from zero. However, a diagonal scaling matrix is not effective for preconditioning during the operating hours. In this study, we use the same type of preconditioner (i.e. the ideal preconditioner) in all periods for easy implementation of the preconditioning technique. During the start-up or shut-down periods, the preconditioner needs to be updated more often because it loses effectiveness quickly due to the rapid dynamic changes in the AHU-VAV system.

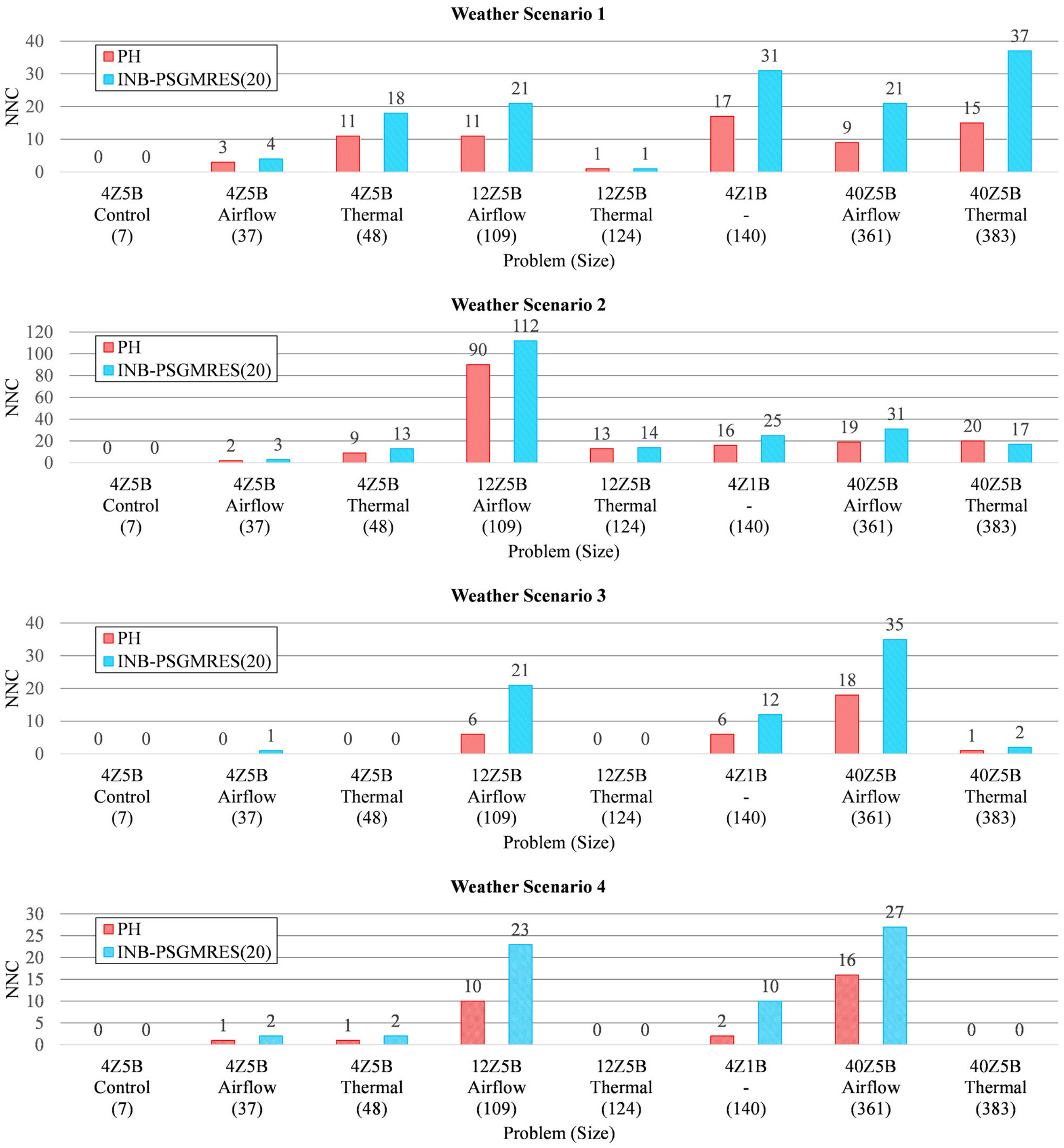

Figure 6. NNC comparison between PH and INB-PSGMRES(20).

3.3.4. More about the performance of INB-PSGMRES(20)

Given the success of the simulations under four representative weather scenarios during a year, we have good reason to believe that INB-PSGMRES(20) will achieve the same performance in other cases. In addition to the tests presented in the above sections, INB-PSGMRES(20) has also been applied to annual simulations of a four-zone dual duct system, a four-zone parallel fan powered system, and a four-zone series fan powered system, respectively. All simulations were accelerated significantly (at least 70% faster compared to the PH method) and terminated successfully.

In summary, INB-PSGMRES(20) is as accurate and robust as PH while maintaining high efficiency. In this experiment, an ideal preconditioner is employed rather than a less expensive preconditioner that approximates the Jacobian with less accuracy (such as incomplete LU and diagonal scaling methods). Although constructing an ideal preconditioner is expensive, the cost is amortized

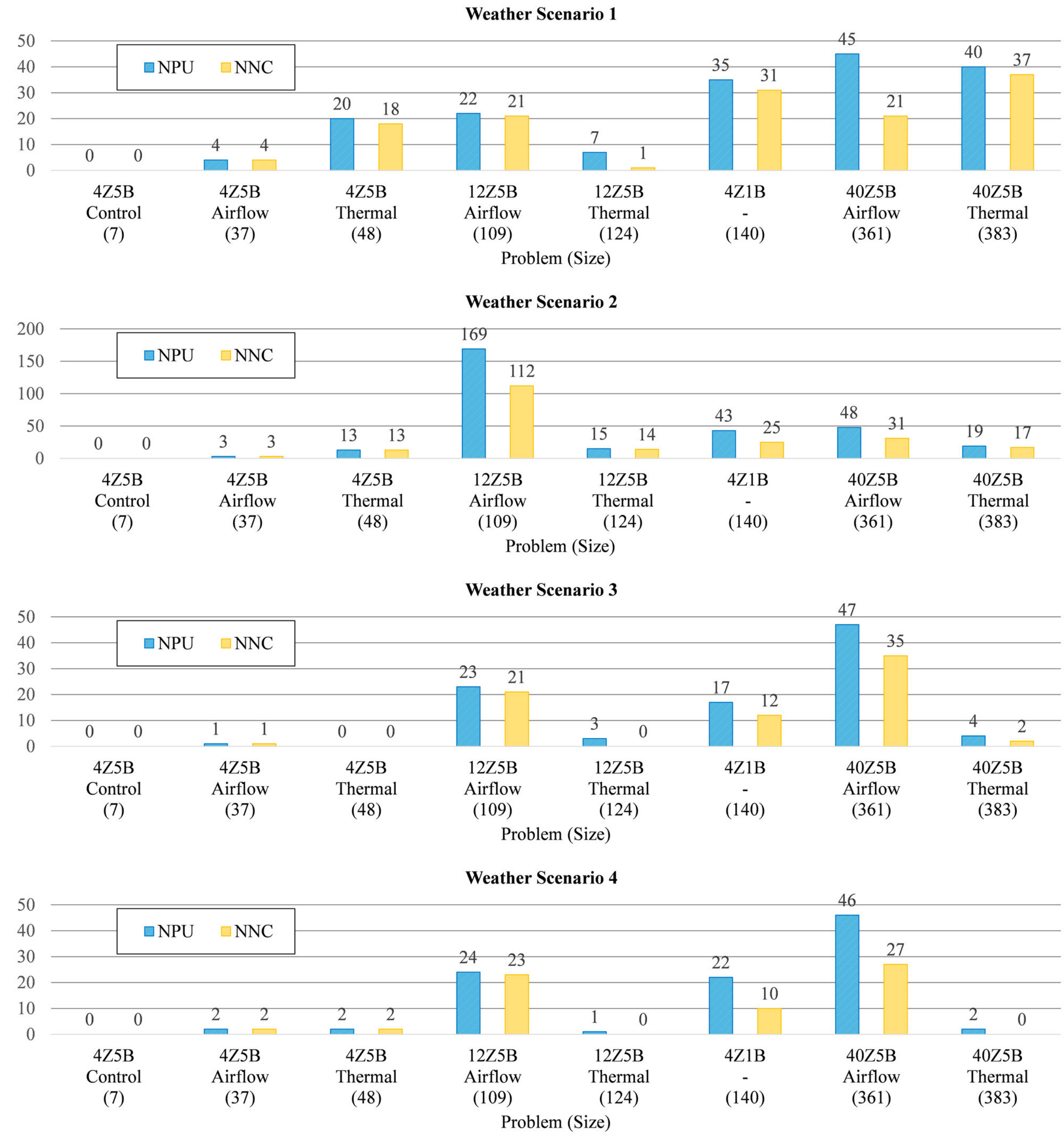

Figure 7. NPU and NNC of INB-PSGMRES(20).

because at most 169 updates are needed for a total of 34,560 time steps during a 24-hour period. Choosing a less expensive preconditioner saves time for a single update, but it may increase the total number of updates due to its lack of accuracy. Therefore, it is not necessary to consider less expensive preconditioners unless a simulation has convergence difficulties for a significant number of time steps due to frequent on-off operations, rapid dynamic changes, or poor model quality (such as discontinuities), which require a significant number of preconditioner updates. Moreover, in the case that a very small $\eta^{(k)}$ (e.g. $\eta^{(k)} = 0.0001$) is chosen, we recommend disabling the preconditioner update within the nonlinear loop (i.e. line 8 in Table 4) to save unnecessary computational cost, because the linear iteration is more likely to return an ill-conditioning warning in such a case. Based on the above discussion, it is believed that INB-PSGMRES(20) is a reliable method to replace PH for large-scale dynamic building system simulation.

4. Conclusions

In this study, we present a preconditioned NK method, INB-PSGMRES(m), for solving large-scale dynamic building system simulations. This method is found to be as robust as a traditional PH method while saving a significant amount of computational time in an HVACSIM+ environment. The success of the method largely depends on the ideal preconditioner and its update scheme. This method is expected to work well in a system where rapid dynamic changes are infrequent. For systems with frequent rapid dynamic changes, such as grid-integrated systems with frequent on-off operations, fan modulation, or setpoint adjustments, the performance of INB-PSGMRES(m) is yet to be seen. In the case that the preconditioner needs to be updated frequently, a less expensive preconditioner is desired. A potential improvement may be found by employing physics-based preconditioners.

Notes

1. When using Gram-Schmidt iteration to find the orthonormal basis for a Krylov subspace, it is known as Arnoldi iteration.
2. An upper Hessenberg matrix is a square matrix that has zero entries below the first subdiagonal.
3. In the case without preconditioning, $\boldsymbol{Jv}$ is approximated with a given $\boldsymbol{v}$ at every linear iteration. In the case with preconditioning, $\boldsymbol{J}(\boldsymbol{M}^{-1}\boldsymbol{v})$ is approximated with a given $\boldsymbol{M}^{-1}$ and $\boldsymbol{v}$ at every linear iteration. In comparison, an additional matrix-vector multiplication for $\boldsymbol{M}^{-1}\boldsymbol{v}$ is performed in the preconditioning case.
4. In the case with preconditioning, an additional matrix-vector multiplication is performed to convert $\boldsymbol{S}$ to $\boldsymbol{s}$ at every nonlinear iteration, i.e., $\boldsymbol{s} = \boldsymbol{M}^{-1}\boldsymbol{S}$.

Data availability statement

The source code and test examples supporting the results of this study are available from the corresponding author on request.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the National Institute of Standards and Technology (NIST) [grant number 60NANB15D251].

ORCID

Zhelun Chen http://orcid.org/0000-0002-5570-1264

References

Arnoldi, W. E. 1951. "The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem." *Quarterly of Applied Mathematics* 9 (1): 17–29.

Benzi, M. 2002. "Preconditioning Techniques for Large Linear Systems: a Survey." *Journal of Computational Physics* 182 (2): 418–477.

Bergero, F. M., F. Casella, E. Kofman, and J. Fernández. 2018. "On the Efficiency of Quantization-Based Integration Methods for Building Simulation." *Building Simulation* 11 (2): 405–418.

Boggs, P. T., A. J. Kearsley, and J. W. Tolle. 1999. "A Practical Algorithm for General Large Scale Nonlinear Optimization Problems." *SIAM Journal on Optimization* 9 (3): 755–778.

Borelli, D., and C. Schenone. 2009. "Analysis of Sound Propagation in Lined Ducts by means of a Finite Element Model." COMSOL Conference 2019 Milan.

Casella, F. 2015. "Simulation of Large-Scale Models in Modelica: State of the art and Future Perspectives." Linköping electronic conference proceedings, pp. 459–468.

Cedeño, C. R. A., M. Vásquez-Bermúdez, J. L. F. Blázquez, I. R. Maestre, J. Hidalgo, M. del Pilar Avilés-Vera, and N. Vera-Lucio. 2018. "Evaluation of the Computation Times for Direct and Iterative Resolution Methods of MTJ Library Matrices Applied in a Thermal Simulation System." International conference on Technologies and innovation: Springer, pp. 110-123.

Chan, T. F., and K. R. Jackson. 1984. "Nonlinearly Preconditioned Krylov Subspace Methods for Discrete Newton Algorithms." *SIAM Journal on Scientific and Statistical Computing* 5 (3): 533–542.

Chen, Z. 2019. "Advanced Solver Development for Large-scale Dynamic Building System Simulation," Ph.D. thesis, Drexel University.

Chen, Z., J. Wen, A. J. Kearsley, and A. J. Pertzborn. 2017. "Scaling Methods for Dynamic Building System Simulation in an HVACSIM+ Environment." 15th IBPSA conference, San Francisco, CA, US, Aug. 7-9, 2017, pp. 2059-2065.

Dennis, J. E., and R. B. Schnabel. 1996. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Philadelphia: SIAM.

Dynasim, A. B. 2004. "Dymola - User's manual".

Fritzson, P., A. Pop, K. Abdelhak, A. Asghar, B. Bachmann, W. Braun, D. Bouskela, et al. 2020. "The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development." *Modeling, Identification and Control* 41 (4): 241–295.

Goellner, J., M. Prica, J. Miller, S. Pullins, J. Westerman, J. Harmon, T. Grabowski, et al. 2011. "Demand dispatch intelligent demand for a more efficient grid," *US National Energy Technology Lab*.

Haves, P., L. K. Norford, and M. DeSimone. 1998. "A Standard Simulation Test Bed for the Evaluation of Control Algorithms and Strategies." *ASHRAE Transactions* 104: 460.

Hermanns, M. 2018. "An Order 102 Speedup in the Computation of the Steady-State Thermal Response of Geothermal Heat Exchangers." *AIP Conference Proceedings* 2040 (1), AIP Publishing LLC, 150002.

Hindmarsh, A. C., R. Serban, and A. Collier. 2017a. "User Documentation for IDA v3.0.0 (SUNDIALS v3.0.0)".

Hindmarsh, A. C., R. Serban, and D. R. Reynolds. 2017b. "User Documentation for CVODE v3.0.0 (SUNDIALS v3.0.0)".

Kaltenbacher, M., A. Hüppe, A. Reppenhagen, M. Tautz, S. Becker, and W. Kuehnel. 2016. "Computational Aeroacoustics for HVAC Systems Utilizing a Hybrid Approach." *SAE International Journal of Passenger Cars-Mechanical Systems* 9 (2016-01-1808): 1047–1052.

Kearsley, A. J. 1996. "The Use of Optimization Techniques in the Solution of Partial Differential Equations from Science and Engineering," Ph.D. thesis, Rice University.

Kitchin, J. 2011. "Smooth Transitions between Discontinuous Functions," [Online]. Available: http://matlab.cheme.cmu.edu/2011/10/30/smooth-transitions-between-discontinuous-functions/.

Klein, S., W. Beckman, J. Mitchell, J. Duffie, N. Duffie, T. Freeman, J. Mitchell, J. Braun, and B. Evans. 2017. "TRNSYS 18 a TRaNsient SYstem Simulation Program, Volume 6: TRNEdit," *Solar Energy Laboratory, Universitiy of Wisconsin-Madison, USA*.

Knoll, D. A., and P. R. McHugh. 1998. "Enhanced Nonlinear Iterative Techniques Applied to a Nonequilibrium Plasma Flow." *SIAM Journal on Scientific Computing* 19 (1): 291–301.

Kośny, J. 2015. "Thermal and Energy Modeling of PCM-Enhanced Building Envelopes." In *PCM-Enhanced Building Components: An Application of Phase Change Materials in Building Envelopes and Internal Structures*, edited by Jan Kośny, 167–234. Cham: Springer.

Krylov, A. 1931. "On the Numerical Solution of the Equation by Which in Technical Questions Frequencies of Small Oscillations of Material Systems are Determined, *Izvestija AN SSSR (News of Academy of Sciences of the USSR)*." *Otdel. mat. i estest. nauk* 7 (4): 491–539.

Magnusson, F., and J. Åkesson. 2015. "Dynamic Optimization in Jmodelica. org." *Processes* 3 (2): 471–496.

Nouidui, T., M. Wetter, and W. Zuo. 2013. "Functional Mock-up Unit for co-Simulation Import in EnergyPlus." *Journal of Building Performance Simulation* 7 (3): 192–202.

Open Source Modelica Consortium. 2018. "OpenModelica User's Guide," vol. 1, no. 1.

Pakiding, L. 2016. "Experimental and Numerical Studies of Seismic Resistant Unbonded Post-Tensioned Cast-in-Place Concrete Walls." Ph.D. thesis, Lehigh University.

Park, C., D. Clark, and G. Kelly. 1986. "HVACSIM+ building systems and equipment simulation program: building-loads calculation," National Bureau of Standards, Washington, DC (USA). Building Equipment Div.

Pernice, M., and H. F. Walker. 1998. "NITSOL: A Newton Iterative Solver for Nonlinear Systems." *SIAM Journal on Scientific Computing* 19 (1): 302–318.

Petzold, L. R. 1982. "A Description of DASSL: A Differential/Algebraic System Solver." Proc. IMACS world congress, pp. 430-432.

Pourarian, S., A. Kearsley, J. Wen, and A. Pertzborn. 2016. "Efficient and Robust Optimization for Building Energy Simulation." *Energy and Buildings* 122: 53–62.

Pourhoseinian, M., N. Asasian-Kolur, and S. Sharifian. 2021. "CFD Investigation of Heat and Moisture Recovery from air with Membrane Heat Exchanger." *Applied Thermal Engineering* 191: 116911.

Powell, M. J. 1970. "A Hybrid Method for Nonlinear Equations." *Numerical Methods for Nonlinear Algebraic Equations* 7: 87–114.

Rädler, J. 2010. "Smooth Transition between Functions with tanh()." [Online]. https://www.j-raedler.de/2010/10/smooth-transition-between-functions-with-tanh/.

Romaní Picas, J., A. D. Gracia Cuesta, and L. F. Cabeza. 2016. "Simulation and Control of Thermally Activated Building Systems (TABS)." *Energy and Buildings* 127: 22–42.

Saad, Y., and M. H. Schultz. 1986. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems." *SIAM Journal on Scientific and Statistical Computing* 7 (3): 856–869.

Sorensen, D. C. 1992. "Implicit Application of Polynomial Filters in a k-Step Arnoldi Method." *SIAM Journal on Matrix Analysis and Applications* 13 (1): 357–385.

Sowell, E. F., and P. Haves. 2001. "Efficient Solution Strategies for Building Energy System Simulation." *Energy and Buildings* 33 (4): 309–317.

SPARK. 2003. "SPARK 2.0 Reference Manual." *Lawrence Berkeley National Laboratory and Ayres Sowell Associates Inc*.

U.S. Energy Information Administration. 2020. "March 2020 Monthly Energy Review."

Walker, H. F., and L. Zhou. 1994. "A Simpler Gmres." *Numerical Linear Algebra with Applications* 1 (6): 571–581.

Wen, J., and S. Li. 2011. *Tools for Evaluating Fault Detection and Diagnostic Methods for Air-Handling Units, ASHRAE RP-1312 Final Report*. Atlanta: American Society of Heating, Refrigerating and Air Conditioning Engineers Inc.

Wetter, M. 2005. "BuildOpt—a new Building Energy Simulation Program That is Built on Smooth Models." *Building and Environment* 40 (8): 1085–1092.

Wetter, M. 2011. "A View on Future Building System Modeling and Simulation." In *Building Performance Simulation for Design and Operation*, edited by Jan L. M. Hensen and Roberto Lamberts, 481–504. New York: Spon Press.

Wetter, M., P. Haves, M. A. Moshier, and E. F. Sowell. 2008. "Using SPARK as a Solver for Modelica." *Lawrence Berkeley National Laboratory*.

Wetter, M., and C. van Treeck. 2017. *IEA EBC Annex 60: New Generation Computing Tools for Building and Community Energy Systems*. Birmingham: EBC Bookshop.

Xu, F., S. Xu, U. Passe, and B. Ganapathysubramanian. 2021. "Computational Study of Natural Ventilation in a Sustainable Building with Complex Geometry." *Sustainable Energy Technologies and Assessments* 45: 101153.

Zhang, L., J. Wen, Y. Li, J. Chen, Y. Ye, Y. Fu, and W. Livingood. 2021. "A Review of Machine Learning in Building Load Prediction." *Applied Energy* 285: 116452.