Received July 3, 2020, accepted July 8, 2020, date of publication July 15, 2020, date of current version July 30, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3009456*

# Reusing Preconditioners in Projection Based Model Order Reduction Algorithms

**NAVNEET PRATAP SINGH AND KAPIL AHUJA**
Data and Computational Sciences Laboratory, IIT Indore, Indore 453552, India

Corresponding authors: Navneet Pratap Singh (navneet.diat@gmail.com) and Kapil Ahuja (kapsahuja22@gmail.com)

The work of Kapil Ahuja was supported by the Mathematical Research Impact Centric Support (MATRICS) Scheme of Department of Science and Technology-Science and Engineering Research Board (DST-SERB), India, under Grant MTR/2017/001023.

**ABSTRACT** Dynamical systems are pervasive in almost all engineering and scientific applications. Simulating such systems is computationally very intensive. Hence, Model Order Reduction (MOR) is used to reduce them to a lower dimension. Most of the MOR algorithms require solving large sparse sequences of linear systems. Since using direct methods for solving such systems does not scale well in time with respect to the increase in the input dimension, efficient preconditioned iterative methods are commonly used. In one of our previous works, we have shown substantial improvements by reusing preconditioners for the parametric MOR (Singh *et al.* 2019). Here, we had proposed techniques for both, the non-parametric and the parametric cases, but had applied them only to the latter. We have three main contributions here. First, we demonstrate that preconditioners can be reused more effectively in the non-parametric case as compared to the parametric one. Second, we show that reusing preconditioners is an art via detailed algorithmic implementations in multiple MOR algorithms. Third and final, we demonstrate that reusing preconditioners for reducing a real-life industrial problem (of size 1.2 million), leads to relative savings of up to 64% in the total computation time (in absolute terms a saving of 5 days).

**INDEX TERMS** Model order reduction, moment matching, iterative methods, preconditioners, reusing preconditioners.

## I. INTRODUCTION

Dynamical systems arise in many engineering and scientific applications such as weather prediction, machine design, circuit simulation, biomedical engineering, etc. Generally, dynamical systems corresponding to real-world applications are extremely large in size. A set of equations describing a parametric nonlinear second-order dynamical system is represented as

$$\begin{aligned} g(\ddot{x}(t), \mathfrak{p}) &= f(\dot{x}(t), \mathfrak{p}) + h(x(t), \mathfrak{p}, u(t)), \\ y(t) &= C^T x(t), \end{aligned} \tag{1}$$

where $t$ is the time variable, $x(t) : \mathbb{R} \rightarrow \mathbb{R}^n$ is the state, $\mathfrak{p} = (p_1, p_2, \ldots, p_k)$ is the set of parameters (with $p_\jmath \in \mathbb{R}$; for $\jmath = 1, \ldots, k$), $u(t) : \mathbb{R} \rightarrow \mathbb{R}^m$ is the input, $y(t) : \mathbb{R} \rightarrow \mathbb{R}^q$ is the output, $C^T \in \mathbb{R}^{q \times n}$ is the output matrix, and $g(\cdot) : \mathbb{R}^{n+k} \rightarrow \mathbb{R}^n$, $f(\cdot) : \mathbb{R}^{n+k} \rightarrow \mathbb{R}^n$ and $h(\cdot) : \mathbb{R}^{n+k+m} \rightarrow \mathbb{R}^n$ are some nonlinear functions [1]–[6]. If $m$ and $q$ both are equal to one, then we have a Single-Input Single-Output (SISO) system. Otherwise, it is called a Multi-Input Multi-Output (MIMO) ($m$ and $q > 1$) system. The functions $g(\cdot)$, $f(\cdot)$, and $h(\cdot)$ are usually simplified as [2], [6]

$$\begin{aligned} g(\ddot{x}(t), \mathfrak{p}) &= \sum_{\jmath=1}^{k} \mathsf{g}_\jmath(\mathfrak{p}) \mathpzc{g}(\ddot{x}(t)), \\ f(\dot{x}(t), \mathfrak{p}) &= \sum_{\jmath=1}^{k} \mathsf{f}_\jmath(\mathfrak{p}) \mathpzc{f}(\dot{x}(t)), \\ h(x(t), \mathfrak{p}, u(t)) &= \sum_{\jmath=1}^{k} \mathsf{h}_\jmath(\mathfrak{p}) \mathpzc{h}(x(t), u(t)), \end{aligned} \tag{2}$$

where $\mathsf{g}_\jmath(\cdot)$, $\mathsf{f}_\jmath(\cdot)$, $\mathsf{h}_\jmath(\cdot) : \mathbb{R}^k \rightarrow \mathbb{R}$ are scalar-valued functions while $\mathpzc{g}(\cdot)$, $\mathpzc{f}(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^n$, and $\mathpzc{h}(\cdot) : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n$ are vector-valued. Next, we look at simplifications to (1) based upon the three predicates; the presence of parameters; the degree of non-linearity, and the order of the system.

- If $\mathsf{g}_\jmath(\mathfrak{p})$, $\mathsf{f}_\jmath(\mathfrak{p})$, and $\mathsf{h}_\jmath(\mathfrak{p})$ are independent of the parameters, then (1) becomes a non-parametric dynamical system.

The associate editor coordinating the review of this manuscript and approving it for publication was Yan-Jun Liu.

**TABLE 1.** **MOR Algorithms.**

| S. No. | Category | | Order | |
|---|---|---|---|---|
| | | | First | Second |
| 1. | Non-parametric Linear | | IRKA [10], $(Sy)^2$IRKA [11] | SOR-IRKA [12], SO-IRKA [13] SOSPDR [14], AIRGA [15] |
| 2. | Non-parametric | Bilinear | BIRKA [16], T-BIRKA [17], [18] | – |
| | | Quadratic-bilinear | QB-IHOMM [19] | – |
| 3. | Parametric Linear | | I-PMOR [20], RPMOR [21] | RPMOR [22] |
| 4. | Parametric | Bilinear | I-PMOR-Bilinear [23] | – |
| | | Quadratic-bilinear | QB-IRKA [24] | – |

- Bilinear systems are one of the common types of nonlinear dynamical systems. Here, there is a product between the state variables and the input variables. Another important class of nonlinear dynamical systems is the quadratic systems. Here, there is product among the state variables. If $g(\cdot)$ and $f(\cdot)$ are linear functions of the state variables, and $h(\cdot)$ is a linear function of the state and the input variables, then (1) is called a linear dynamical system.
- Finally, if the second derivative term in (1) is not present, then (1) becomes a first-order dynamical system.

Simulation of large dynamical systems can be unmanageable due to high demands on computational resources. These large systems can be reduced into a smaller dimension by using Model Order Reduction (MOR) techniques [4], [7]–[11]. The reduced system has approximately the same characteristics as the original system but it requires significantly less computational effort in simulation. MOR can be done in many ways such as balanced truncation, Hankel approximations, and Krylov projection [4], [7], [8], [11]. Among these, the projection methods are quite popular, and hence, we focus on them.

Some of the commonly used projection-based MOR algorithms for different types of dynamical systems are summarized in Table 1.

In the above mentioned MOR algorithms, sequences of very large and sparse linear systems arise during the model reduction process. Solving such linear systems is the main computational bottleneck in efficient scaling of these MOR algorithms for reducing extremely large dynamical systems. Preconditioned iterative methods are commonly used for solving such linear systems [25], [26]. In most of the above listed MOR algorithms, the change from one linear system to the next is usually very small, and hence, the applied preconditioner could be reused.

Next, we briefly summarize the past work that has been done in the field of reusing preconditioners. References [27] and [28] first applied this technique in the quantum Monte Carlo context, where it is referred to as recycling preconditioners. For the case when the linear system coefficient matrices are perturbed by a varying constant times the identity matrix, efficient preconditioners have also been developed. These preconditioners are independent of the underlying application and are referred to as preconditioner updates (see [29] for Symmetric Positive Definite (SPD) coefficient matrices and [30] for general coefficient matrices).

This approach has been used in the optimization context in [31], where it is again termed as preconditioner updates. In the MOR context, [12] and [32] have used this technique for MOR of non-parametric linear first-order dynamical systems (part of the first category above).

The main goal of this paper is to demonstrate the reuse of preconditioners in the remainder of the algorithms for the first category above (MOR of non-parametric linear second-order dynamical systems) as well as the algorithms for the second category above (MOR of non-parametric bilinear/ quadratic-bilinear dynamical systems).

In one of our recent works [33], we had proposed a general framework for reuse of preconditioners during MOR of both non-parametric and parametric dynamical systems. However, in [33] we had demonstrated application of this framework for the parametric case only. That is, the third category above (MOR of parametric linear dynamical systems). We are currently (and separately) working on the algorithms for the fourth category above as well (MOR of parametric bilinear/quadratic-bilinear dynamical systems).

To summarize, in this paper we broadly demonstrate the application of our above mentioned framework for MOR of non-parametric dynamical systems. We have three contributions as below, which have not been catered in any of the above cited papers.

(i) We demonstrate that because of the lack of the parameters in the non-parametric case, the reuse of preconditioners here can be done more effectively as compared to the parametric case.
(ii) We show that the reuse of preconditioners needs to be fine-tuned for the underlying MOR algorithm. We also highlight that there are multiple pitfalls in the algorithmic implementation of reusing preconditioners.
(iii) We experiment on a massively large and real-life industrial problem (BMW disc brake model), which is of size 1.2 million. Here, we are able to reduce the computation time from 197 hours to about 72 hours (relative saving of 64 %).

The paper has four more sections. We discuss MOR techniques in Section II. The theory of reusing preconditioners is described in Section III. We support our theory with numerical experiments in Section IV. Finally, conclusions and future work are discussed in Section V. For the rest of this paper, $\|\cdot\|_f$ denotes the Frobenius norm, $\|\cdot\|$ denotes the Euclidean norm for vectors and the induced spectral norm for matrices, $\otimes$ refers to the Kronecker product (i.e. an operation on two matrices of arbitrary size), $vec(\cdot)$ signifies the vectorization of a matrix, and $I$ denotes the Identity matrix.

## II. MOR

As above, our focus is on MOR of the non-parametric dynamical systems. Hence, we summarize some of the previously listed such algorithms here. Adaptive Iterative Rational Global Arnoldi (AIRGA) [15] is a Ritz-Galerkin projection based algorithm for MOR of linear second-order MIMO dynamical systems with proportional damping, which for the MIMO case are represented as

$$\begin{aligned} M\ddot{x}(t) &= -D\dot{x}(t) - Kx(t) + Fu(t), \\ y(t) &= C^T x(t), \end{aligned} \tag{3}$$

where $M,\ D,\ K \in \mathbb{R}^{n\times n},\ F \in \mathbb{R}^{n\times m},\ C \in \mathbb{R}^{n\times q}$, and $D = \alpha M + \beta K$. Here, $\alpha,\ \beta$ are some scalar values. Let $V \in \mathbb{R}^{n\times r}$ and its columns span a $r$-dimension subspace ($r \ll n$). In principle, the Ritz-Galerkin projection method involves the steps below.

- Approximating the reduced state vector $\hat{x}(t)$ using $V$ as $x(t) \approx V\hat{x}(t)$ leads to
$$\begin{aligned} MV\ddot{\hat{x}}(t) + DV\dot{\hat{x}}(t) + KV\hat{x}(t) - Fu(t) &= r(t), \\ \hat{y}(t) &= C^T V\hat{x}(t), \end{aligned}$$
where $r(t)$ is the residual after projection.
- Enforcing the residual $r(t)$ to be orthogonal to $V$ or $V^T r(t) = 0$ leads to the reduced system given as follows:
$$\begin{aligned} \hat{M}\ddot{\hat{x}}(t) + \hat{D}\dot{\hat{x}}(t) + \hat{K}\hat{x}(t) - \hat{F}u(t) &= 0, \\ \hat{y}(t) &= \hat{C}^T \hat{x}(t), \end{aligned}$$

where $\hat{M} = V^T MV,\ \hat{D} = V^T DV,\ \hat{K} = V^T KV,\ \hat{F} = V^T F$, and $\hat{C}^T = C^T V$. To compute this projection matrix $V$, AIRGA matches the moments of the original system transfer function and the reduced system transfer function. We briefly summarize AIRGA in Algorithm 1, where parts relevant to solving linear systems are only listed.

Bilinear Iterative Rational Krylov Algorithm (BIRKA) [16] is a Petrov-Galerkin projection based algorithm for MOR of the bilinear first-order dynamical systems, which for the MIMO case are represented as

$$\begin{aligned} \dot{x}(t) &= Kx(t) + \sum_{\mathrm{j}=1}^{m} N_{\mathrm{j}} x(t) \mathrm{u}_{\mathrm{j}}(t) + Fu(t), \\ y(t) &= C^T x(t), \end{aligned} \tag{4}$$

where $K,\ N_{\mathrm{j}} \in \mathbb{R}^{n\times n},\ F \in \mathbb{R}^{n\times m},\ C \in \mathbb{R}^{n\times q}$, and $u = [\mathrm{u}_1,\ \mathrm{u}_2,\ \ldots, \mathrm{u}_m] \in \mathbb{R}^m$. Let columns of $V, W \in \mathbb{R}^{n\times r}$ span two $r$-dimension subspaces (where, as earlier, $r \ll n$ ).

**Algorithm 1** AIRGA [15]

```
    Input: $M,\ D,\ K,\ F,\ C$; $S$ is the set of initial expansion
    points $s_i,\ i = 1,\ \ldots,\ \ell$.
    Output: $\hat{M},\ \hat{D},\ \hat{K},\ \hat{F},\ \hat{C}$.
 1: $z = 1$
 2: while (no convergence) do
 3:     for $i = 1,\ \ldots,\ \ell$ do
 4:         $X^{(0)}(s_i) = (s_i^2 M + s_i D + K)^{-1} F$
 5:         $V_1 = \frac{X^{(0)}(s_i)}{\|X^{(0)}(s_i)\|_f}$
 6:     end for
 7:     j = 1
 8:     while (no convergence) do
 9:         for $i = 1,\ \ldots,\ \ell$ do
10:             $X^{(j)}(s_i) = -(s_i^2 M + s_i D + K)^{-1} M V_j$
11:             $V_{j+1} = \frac{X^{(j)}(s_i)}{\|X^{(j)}(s_i)\|_f}$
12:         end for
13:         $j = j + 1$
14:     end while
15:     "All the given set of expansion points
         (i.e. $s_1,\ s_2,\ \ldots,\ s_\ell$) are updated"
16:     $z = z + 1$
17: end while
18: $\hat{M} = V^T MV,\ \hat{D} = V^T DV,\ \hat{K} = V^T KV,\ \hat{F} =$
    $V^T F$, and $\hat{C}^T = C^T V$
```

In principle, the Petrov-Galerkin projection method involves the steps below.

- Approximating the reduced state vector $\hat{x}(t)$ using $V$ as $x(t) \approx V\hat{x}(t)$ leads to
$$\begin{aligned} V\dot{\hat{x}}(t) - KV\hat{x}(t) - \sum_{\mathrm{j}=1}^{m} N_{\mathrm{j}} V\hat{x}(t)\mathrm{u}_{\mathrm{j}}(t) - Fu(t) &= r(t), \\ \hat{y}(t) &= C^T V\hat{x}(t), \end{aligned}$$
where $r(t)$ is the residual after projection.
- Enforcing the residual $r(t)$ to be orthogonal to $W$ or $W^T r(t) = 0$ leads to the reduced system given by
$$\begin{aligned} \dot{\hat{x}}(t) - \hat{K}\hat{x}(t) - \sum_{\mathrm{j}=1}^{m} \hat{N}_{\mathrm{j}} \hat{x}(t)\mathrm{u}_{\mathrm{j}}(t) - \hat{F}u(t) &= 0, \\ \hat{y}(t) &= \hat{C}^T \hat{x}(t), \end{aligned}$$

where $\hat{K} = (W^T V)^{-1} W^T KV,\ \hat{N}_{\mathrm{j}} = (W^T V)^{-1} W^T N_{\mathrm{j}} V,\ \hat{F} = (W^T V)^{-1} W^T F,\ \hat{C}^T = C^T V$, and $(W^T V)^{-1}$ is assumed to be invertible. Here, $V$ and $W$ are computed by using interpolation, where the original system transfer function and its derivative are respectively matched with the reduced system transfer function and its derivative at a set of points. We briefly summarize BIRKA in Algorithm 2, where again, only parts related to solving linear systems are listed.

Quadratic Bilinear-Implicit Higher Order Moment Matching (QB-IHOMM) [19] is a Petrov-Galerkin projection based

**Algorithm 2** BIRKA [16]

```
    Input $K, N_1, \ldots, N_m, F, C$, and initial guess of the reduced system $\check{K}, \check{N}_1, \ldots, \check{N}_m, \check{F}, \check{C}$
    Output $\hat{K}, \hat{N}_1, \ldots, \hat{N}_m, \hat{F}$, and $\hat{C}$
 1: $z = 1$
 2: while (no convergence) do
 3:     $R\Lambda R^{-1} = \check{K}, \check{\check{F}} = \check{F}^T R^{-T}, \check{\check{C}} = \check{C}R, \check{\check{N}}_j = R^T \check{N}_j R^{-T}$ for j = 1, ..., $m$
 4:     $vec(V) = \left(-\Lambda \otimes I_n - I_r \otimes K - \sum_{j=1}^{m} \check{\check{N}}_j^T \otimes N_j\right)^{-1} \left(\check{\check{F}}^T \otimes F\right) vec(I_m)$
 5:     $vec(W) = \left(-\Lambda \otimes I_n - I_r \otimes K^T - \sum_{j=1}^{m} \check{\check{N}}_j \otimes N_j^T\right)^{-1} \left(\check{\check{C}}^T \otimes C^T\right) vec(I_q)$
 6:     $V = orth(V), \; W = orth(W)$
 7:     $\check{K} = (W^T V)^{-1} W^T K V, \; \check{N}_j = (W^T V)^{-1} W^T N_j V, \; \check{F} = (W^T V)^{-1} W^T F, \; \check{C} = CV$
 8:     $z = z + 1$
 9: end while
10: $\hat{K} = \check{K}, \; \hat{N}_j = \check{N}_j, \; \hat{F} = \check{F}$, and $\hat{C} = \check{C}$
```

algorithm for MOR of the quadratic-bilinear first-order dynamical systems, which for the SISO case are represented as[1]

$$D\dot{x}(t) = Kx(t) + Nx(t)u(t) + H\,(x(t) \otimes x(t)) + Fu(t),$$
$$y(t) = C^T x(t), \tag{5}$$

where $D, K, N \in \mathbb{R}^{n \times n}$, $H \in \mathbb{R}^{n \times n^2}$, $F \in \mathbb{R}^{n \times 1}$, $C \in \mathbb{R}^{n \times 1}$. Let columns of $V, W \in \mathbb{R}^{n \times r}$ span two $r$-dimension subspaces (where as earlier, $r \ll n$). In principle, the Petrov-Galerkin projection method involves the steps below.

- As before, approximating the reduced state vector $\hat{x}(t)$ using $V$ as $x(t) \approx V\hat{x}(t)$ leads to

$$DV\dot{\hat{x}}(t) - KV\hat{x}(t) - NV\hat{x}(t)u(t) - H\left(V\hat{x}(t) \otimes V\hat{x}(t)\right) - Fu(t) = r(t),$$
$$y(t) = C^T V\hat{x}(t),$$

  where $r(t)$ is the residual after projection.
- Enforcing the residual $r(t)$ to be orthogonal to $W$ or $W^T r(t) = 0$ leads to the reduced system given by

$$\hat{D}\dot{\hat{x}}(t) - \hat{K}\hat{x}(t) - \hat{N}\hat{x}(t)u(t) - \hat{H}\left(\hat{x}(t) \otimes \hat{x}(t)\right) - \hat{F}u(t) = 0,$$
$$y(t) = \hat{C}^T \hat{x}(t),$$

where $\hat{D} = W^T DV$, $\hat{K} = W^T KV$, $\hat{N} = W^T NV$, $\hat{H} = W^T H(V \otimes V)$, $\hat{F} = W^T F$, $\hat{C}^T = C^T V$. Here, $V$ and $W$ are computed by matching the moments of the original system transfer function and the reduced system transfer function. We briefly summarize QB-IHOMM in Algorithm 3, where as earlier, only parts related to solving linear systems are listed. Here, as in [19], the computation is done with the first two regular transfer function terms.

[1]A variant of BIRKA for MOR of the quadratic-bilinear first-order dynamical systems also exists. Preconditioned iterative solves and reusing preconditioners can be applied here as done for BIRKA. Hence, we focus on QB-IHOMM that has been developed for the SISO case only.

**Algorithm 3** QB-IHOMM [19]

```
    Input: $D, K, N, H, F, C$; interpolation points $\sigma_i \in \mathbb{C}$ for $i = 1, \ldots, \ell$; higher orders moments numbers $P, Q \in \mathbb{N}$
    Output: $\hat{D}, \hat{K}, \hat{N}, \hat{H}, \hat{F}, \hat{C}$
 1: $V = [\,], \; W = [\,]$
 2: for $j = 0, \ldots, P + Q$ do
 3:     for $i = 1, \ldots, \ell$ do
 4:         $X_j(\sigma_i) = [(\sigma_i D - K)^{-1} D]^j (\sigma_i D - K)^{-1} F$
 5:         $V = [V \;\; X_j(\sigma_i)]$
 6:     end for
 7: end for
 8: for $j = 0, \ldots, Q$ do
 9:     for $i = 1, \ldots, \ell$ do
10:         $X_j(2\sigma_i)^T = [(2\sigma_i D - K)^{-T} D^T]^j (2\sigma_i D - K)^{-T} C^T$
11:         $W = [W \;\; X_j(2\sigma_i)^T]$
12:     end for
13: end for
14: $U = orth([V \; W])$
15: Construct the reduced system as
    $\hat{D} = U^T DU, \; \hat{K} = U^T KU, \; \hat{N} = U^T NU,$
    $\hat{H} = U^T H(U \otimes U), \; \hat{F} = U^T F, \; \hat{C}^T = C^T U.$
```

## III. PROPOSED WORK

Here, we discuss preconditioned iterative methods in Section III-A. In Section III-B, we revisit the theory of reusing preconditioners from [33]. Finally, we discuss application of reusing preconditioners to the earlier discussed algorithms in Section III-C.

### A. PRECONDITIONED ITERATIVE METHODS

Krylov subspace based methods are very popular class of iterative methods [34], [35]. Let $Ax = b$ be a linear system, with $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $x_0$ the initial solution and $r_0$ (where $r_0 = b - Ax_0$) the initial residual. We find the solution of a linear system in $\mathbb{K}_{\hbar}(A, r_0) = span\{r_0, Ar_0, A^2 r_0, \ldots, A^{\hbar - 1} r_0\}$, where $\mathbb{K}_{\hbar}(\cdot, \cdot)$ represents the Krylov subspace.

Often iterative methods are slow or fail to converge, and hence, preconditioning is used to accelerate them. If $P$ is a non-singular matrix that approximates the inverse of $A$, then the preconditioned system becomes $AP\tilde{x} = b$ with $x = P\tilde{x}$. This is termed as right preconditioning. Similarly, left preconditioning can also be performed, where the preconditioner is present on the left side of the matrix [36].[2] If the linear system coefficient matrices are SPD, then both the types of preconditioning give the same results [36]. For our MOR algorithms under-consideration, the linear system coefficient matrices do not have any special structure. Hence, both these types of preconditioning work differently.

In our experiments, we use right preconditioning because it is fairly common [37], [38]. However, to demonstrate that our techniques are independent of the type of preconditioning, for one model, we experiment with left preconditioning in the side as well.

We expect that the preconditioned iterative solves would find a solution in less amount of time as compared to the unpreconditioned ones. For most of the input dynamical systems (as mentioned here), the Krylov subspace methods fail to converge (see Numerical Experiments section). Hence, we use a preconditioner. The goal is to find a preconditioner that is cheap to compute as well as apply. Preconditioning is of two kinds (implicit and explicit), and we focus on the latter [39].

In case of implicit preconditioners, application of preconditioning requires solving linear systems. For example, in factorization based preconditioning $A \approx LU$, where $L$ and $U$ are sparse triangular matrices approximating exact $L$ and $U$ factors. Here, application of the preconditioner requires only forward and backward solves. This is usually referred to as incomplete LU factorization (ILU) based preconditioner. Variations of ILU that exploit certain matrix constructs can also be developed. For example, ILU based upon Schur's complement [40]. Further, ILQ, SSOR and ADI are other kinds of preconditioning that fall under the implicit category [39].

Although implicit preconditioners have been used extensively for a very long time, they have their own drawbacks. For example, ILU based preconditioners do not be scale well when the system size becomes very large (computation time becomes prohibitively expensive). This is because, forward and backward solves in such preconditioners are inherently sequential and cannot be easily parallelized. Besides this, the breakdown in the factorization process because of the zero pivoting carries over from the full factorization case to this incomplete factorization case.

Explicit preconditioning is one where directly the inverse of the coefficient matrix is approximated or $P \approx A^{-1}$. Hence, applying the preconditioner just involves performing matrix-vector products [38]. Sparse approximate inverse (SPAI) are the most commonly used explicit preconditioners, which we use and are discussed in-detail later in this section.

[2] If the preconditioner is present on both the sides of the coefficient matrix, then it is called split/ center preconditioning.

Variations of approximate inverse preconditioners also exist. One example, as we have seen in the case of implicit preconditioning, is the Schur's complement based approximate inverse preconditioner [38]. Another example is where the approximate inverse preconditioner is constructed by using a high-order convergent scheme that relies on matrix-matrix multiplications [41], [42].

Hybrid of implicit and explicit preconditioning is also common. Here, combinations of factorizations and approximate inverses are used to compute a preconditioner. An example of this is given in [43], where for a SPD matrix, Cholesky factorization is first performed. This in-turn is used to obtain a more efficient approximate inverse preconditioner. Another example is where the approximate inverse of the coefficient matrix is used to compute an approximation to matrix's Schur's complement. This is then used to build an ILU preconditioner [40].

Now, we give the details of SPAI. For constructing a preconditioner $P$ corresponding to a coefficient matrix $A$, we focus on methods for finding approximate inverse of $A$ by minimizing the Frobenius norm of the residual matrix $I - AP$. This minimization problem can be rewritten as [37]

$$\min_{P} \|I - AP\|_f^2. \tag{6}$$

Here, the columns of residual matrix $I - AP$ can be computed independently, which is an important property that can be exploited. Hence, the solution of (6) can be separated into $n$ independent least square problems as

$$\begin{aligned} &\min_{P} \sum_{i=1}^{n} \|(I - AP)e^{i}\|_2^2, \text{ or} \\ &\min_{p^{i}} \|e^{i} - Ap^{i}\|_2^2, \quad \text{for } i = 1, 2, \ldots, n, \end{aligned} \tag{7}$$

where $e^{i}$ and $p^{i}$ are the $i$-th column of $I$ and $P$, respectively. The above minimization problem can be implemented in parallel and one can efficiently obtain the explicit approximate inverse $P$ of $A$.

Usually $A$ is sparse. In this case, we can solve a more efficient version of the optimization problem given in (7). Here, first, a good sparsity pattern of $P$ is assumed (usually the Identity matrix). As the solutions of the least squares problems are iteratively computed, this sparsity pattern is updated. One common updating strategy adaptively exploits the number of non-zeros arising in the resulting residuals ($r^{i} = e^{i} - Ap^{i}$), which requires solving 1D minimization problems [38]. A more sophisticated updating strategy uses a multivariate minimization [44]. Second, now since both $A$ and $P$ are sparse, we solve much smaller least squares problems, and all matrix-vector products are done in a *sparse-mode* (operations involving a sparse-matrix and a sparse-vector).

### B. THEORY OF REUSING PRECONDITIONERS

In general, the linear systems of equations generated by lines 4 and 10 of Algorithm 1 (AIRGA); lines 4 and 5 of Algorithm 2 (BIRKA); and lines 4 and 10 of Algorithm 3 (QB-IHOMM) have the following form:

$$\begin{aligned} A_1 X_1 &= F_1, \\ A_2 X_2 &= F_2, \\ &\vdots \\ A_\ell X_\ell &= F_\ell, \end{aligned}$$

where $A_i \in \mathbb{R}^{n\times n}$, $X_i \in \mathbb{R}^n$, and $F_i \in \mathbb{R}^n$; for $i = 1, 2, \ldots, \ell$.[3]

Let $P_1$ be a good preconditioner for $A_1$ (it is a seed preconditioner) that is computed by the theory discussed in the above section ((6)-(7)) or

$$\min_{P_1} \|I - A_1 P_1\|_f^2.$$

Now, we need to find a good preconditioner $P_2$ corresponding to $A_2$. Using the standard SPAI theory, this means solving

$$\min_{P_2} \|I - A_2 P_2\|_f^2. \tag{8}$$

If we are able to enforce $A_1 P_1 = A_2\ P_2$, then $P_2$ will be an equally good preconditioner for $A_2$ as much as $P_1$ is a good preconditioner for $A_1$ (since the Spectrum of $A_2 P_2$ would be same as that of $A_1 P_1$, on which convergence of any Krylov subspace method depends). Since $P_2$ is unknown here, we have a degree of freedom in choosing how to form it. Without loss of generality, we assume that $P_2 = Q_2 P_1$, where $Q_2$ is an unknown matrix. Here, we need to enforce $A_1 P_1 = A_2 Q_2 P_1$. Thus, instead of solving the minimization problem (8), we can solve

$$\min_{Q_2} \|A_1 - A_2\ Q_2\|_f^2.$$

Note that $P_2$ here is never explicitly formed by multiplying two matrices $Q_2$ and $P_1$. Rather, always a matrix-vector product is done to apply the preconditioner.

Next, we apply a similar argument for finding a good preconditioner $P_i$ corresponding to $A_i$. For this we refer to one of our recent works [33], which focused on MOR of parametric linear dynamical systems (category three from the Introduction). We can obtain $P_i$ by enforcing either $A_1 P_1 = A_i P_i$ or $A_{i-1} P_{i-1} = A_i P_i$. For these two cases, $P_i$ would be as effective preconditioner for $A_i$ as $P_1$ is for $A_1$ or $P_{i-1}$ is for $A_{i-1}$, respectively. These two approaches are summarized in Table 2.

In [33], we have conjectured (with evidence) the following two results: (a) In the parametric case, the first approach is more beneficial. This is because, in this case although the two approaches have a similarly hard minimization problem (attributed to slowly varying parameters, and in-turn, slowly changing matrices), the computation of $P_i$ from $P_1$ in the first approach leads to a preconditioner with less approximation errors, and hence, a one which is more accurate. (b) In the non-parametric case, the second approach is more suited. This is because in this case the minimization problem of the second approach is much easier to solve as compared to the first approach (attributed to rapidly changing expansion/ interpolation points, and in-turn, rapidly changing matrices). The computation of $P_i$ from $P_{i-1}$ in this case (rather than $P_1$ as above) does have the drawback of accumulated approximation errors, however, solving the minimization problem efficiently is a bigger bottleneck for scaling to large problems.

[3]In-case of BIRKA, the coefficient matrices are of size $n^2 \times n^2$ and solution vectors as well as right-hand sides of size $n^2$.

**TABLE 2. Cheap preconditioner update approaches [33].**

| First approach | Second approach |
|---|---|
| • $A_1 P_1 = A_i P_i$<br>• If $P_i = Q_i P_1$, then $A_1 P_1 = A_i Q_i P_1$<br>• $\min_{Q_i} \|A_1 - A_i Q_i\|_f^2$ | • $A_{i-1} P_{i-1} = A_i P_i$<br>• If $P_i = Q_i P_{i-1}$, then $A_{i-1} P_{i-1} = A_i Q_i P_{i-1}$<br>• $\min_{Q_i} \|A_{i-1} - A_i Q_i\|_f^2$ |

As mentioned in the Introduction, in [33] we have extensively experimented for the parametric case (again, category three earlier) using the first approach. The focus here is to do a similar experimentation for the non-parametric case (first two categories earlier) using the second approach.

### C. APPLICATION OF REUSING PRECONDITIONER

Here, we first discuss the application of the above presented theory of reusing preconditioners to AIRGA. If we closely observe Algorithm 1, as mentioned earlier, linear systems are solved at lines 4 and 10. To solve these system, we can chose any solver from a large pool of available Krylov subspace methods. For example, GMRES [45], BI-CGSTAB [46], IDR(s) [47], etc. Since GMRES is the most popular one among these, we use it inside AIRGA in our result section.

If we relook at linear systems at lines 4 and 10 in Algorithm 1, we realize that they have more characteristics. These linear systems can be very easily transformed into general shifted linear systems of the form $\varsigma\mathscr{D} + \mathscr{K}$ (see Section 3 of [48]). Therefore, this property can be exploited in solving these sets of linear systems simultaneously [49], [50], which is part of our future work.[4]

Delving further into the complexity of such linear systems, we observe that the matrices change with the index of outer `while` loop (line 2) as well as with the index of the `for` loop corresponding to the expansion points (line 3). Hence, we denote such matrices not only with a subscript as in previous subsection but also with a superscript. That is, $A_i^{(z)} = \left(s_i^{(z)}\right)^2 M + s_i^{(z)} D + K$, where $z = 1, \ldots, \mathfrak{z}$ (until covergence) and $i = 1, \ldots, \ell$. As the matrix

[4]If the linear system coefficient matrices have special properties, then more efficiency can be incorporated. For example, if the coefficient matrices $(\varsigma\mathscr{D} + \mathscr{K})$ have $\mathscr{D}$, $\mathscr{K}$ as real and $\varsigma$ as complex, then we can reduce the number of linear systems that are required to be solved. For more details, please see Section 1 of [54].

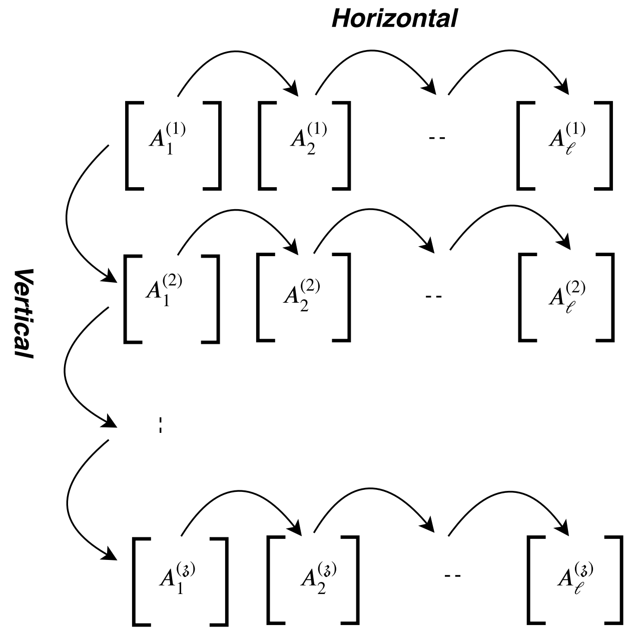

**FIGURE 1.** **Reusing preconditioners in AIRGA.**

$A_i^{(z)}$ changes with respect to two different indices, we can reuse preconditioners in many ways. However, here we use the second approach as discussed in the previous subsection. This approach is diagrammatically represented in Figure 1.

Computation of preconditioners is done only at line 4 because at line 10, matrices do not change, only the right-hand sides do. Hence, we only focus on reusing preconditioners for line 4.

Next, we show how the new preconditioners are computed for both, the horizontal direction and the vertical direction. While looking at the horizontal route, let,

$$A_{i-1}^{(z)} = \left(s_{i-1}^{(z)}\right)^2 M + s_{i-1}^{(z)} D + K$$

and

$A_i^{(z)} = \left(s_i^{(z)}\right)^2 M + s_i^{(z)} D + K$ be the two coefficient matrices for different expansion points $s_{i-1}^{(z)}$ and $s_i^{(z)}$, respectively, with $i = 2, \ldots, \ell$. Using the above theory, we enforce $A_{i-1}^{(z)} P_{i-1}^{(z)} = A_i^{(z)} P_i^{(z)}$ in Figure 2. Thus, we eventually enforce $A_{i-1}^{(z)} P_{i-1}^{(z)} = A_i^{(z)} Q_i^{(z)} P_{i-1}^{(z)}$ and solve the minimization problem

$$\min_{Q_i^{(z)}} \|A_{i-1}^{(z)} - A_i^{(z)} Q_i^{(z)}\|_f^2.$$

This gives us the new preconditioner $P_i^{(z)} = Q_i^{(z)} P_{i-1}^{(z)}$. This minimization is again performed for $n$ independent least square problems as in (7). Similar steps are followed for reusing preconditioners along the rest of the horizontal directions, i.e. for all $z = 1, \ldots, \mathfrak{z}$.

Now, applying this technique for the vertical direction, we have for $z = 2, \ldots, \mathfrak{z}$

$$A_1^{(z-1)} P_1^{(z-1)} = A_1^{(z)} P_1^{(z)}.$$

Following the steps as for the horizontal direction, here, we solve the minimization problem

$$\min_{Q_1^{(z)}} \|A_1^{(z-1)} - A_1^{(z)} Q_1^{(z)}\|_f^2.$$

This gives us the new preconditioner $P_1^{(z)} = Q_1^{(z)} P_1^{(z-1)}$. Again, this is solved as $n$ independent least square problems as in (7).

AIRGA with an efficient implementation of the above discussed theory of reusing preconditioners is given in Algorithm 4. If we closely look at line 4 of Algorithm 1, the solution vector is denoted by $X^{(0)}(s_i)$, where the superscript "0" refers to the index of the inner `while` loop (line 8). We do not bother about this index because, as earlier, matrix does not change inside this inner loop. Rather, we need to capture the change because of the outer `while` loop indexed with $z$. Hence, we denote the solution vector as $\mathcal{X}^{(z)}(s_i)$ in Algorithm 4 (lines 8, 11, 19 & 22). It is important to emphasize again that preconditioners are never computed explicitly. Rather, they are obtained using matrix-vector products (please see line numbers 11, 19 & 22 of Algorithm 4).

Since shift-invariant preconditioners have been proposed for the general shifted linear systems [49], [51], our this reuse

$$A_{i-1}^{(z)} P_{i-1}^{(z)} = \overbrace{A_{i-1}^{(z)} \left(I + \left(\left(s_i^{(z)}\right)^2 - \left(s_{i-1}^{(z)}\right)^2\right)\left(A_{i-1}^{(z)}\right)^{-1} M + \left(s_i^{(z)} - s_{i-1}^{(z)}\right)\left(A_{i-1}^{(z)}\right)^{-1} D\right)}^{A_i^{(z)}} P_i^{(z)},$$

$$= A_{i-1}^{(z)} \left(I + \left(\left(s_i^{(z)}\right)^2 - \left(s_{i-1}^{(z)}\right)^2\right)\left(A_{i-1}^{(z)}\right)^{-1} M + \left(s_i^{(z)} - s_{i-1}^{(z)}\right)\left(A_{i-1}^{(z)}\right)^{-1} D\right)$$

$$\underbrace{\underbrace{\cdot \left(I + \left(\left(s_i^{(z)}\right)^2 - \left(s_{i-1}^{(z)}\right)^2\right)\left(A_{i-1}^{(z)}\right)^{-1} M + \left(s_i^{(z)} - s_{i-1}^{(z)}\right)\left(A_{i-1}^{(z)}\right)^{-1} D\right)^{-1}}_{Q_i^{(z)}} P_{i-1}^{(z)}}_{P_i^{(z)}}.$$

**FIGURE 2.** **Expressing one linear system matrix in terms of the other.**

**TABLE 3. SPAI and reusable SPAI analysis for the academic disk brake model.**

| AIRGA Itr.[†] | Exp. Pts.[‡] | SPAI Case | Reusable SPAI Case | |
|---|---|---|---|---|
| | | Standard | Horizontal | Vertical |
| | | $\frac{\|I-A_i^{(z)}\|_f}{\|I\|_f}$ | $\frac{\|A_{i-1}^{(z)}-A_i^{(z)}\|_f}{\|A_{i-1}^{(z)}\|_f}$ | $\frac{\|A_1^{(z-1)}-A_1^{(z)}\|_f}{\|A_1^{(z-1)}\|_f}$ |
| 1 | 1 | $3.77\times10^{06}$ | *NA* | *NA* |
| | 2 | $4.36\times10^{06}$ | 0.1569 | |
| | 3 | $4.95\times10^{06}$ | 0.3139 | *NA* |
| | 4 | $5.54\times10^{06}$ | 0.4708 | |
| 2 | 1 | $7.63\times10^{06}$ | *NA* | 0.9996 |
| | 2 | $4.06\times10^{06}$ | 0.0180 | |
| | 3 | $1.62\times10^{06}$ | 20.3431 | *NA* |
| | 4 | $3.82\times10^{06}$ | 0.4985 | |

[†] AIRGA Iterations.
[‡] Expansion Points.

**TABLE 4. SPAI and reusable SPAI computation time for the academic disk brake model.**

| AIRGA Iterations ($z$) | Expansion Points ($s_i$) | SPAI (Seconds) | Reusable SPAI (Seconds) |
|---|---|---|---|
| 1 | 1 | 174 | 174 |
| | 2 | 164 | 10 |
| | 3 | 165 | 16 |
| | 4 | 165 | 20 |
| 2 | 1 | 165 | 64 |
| | 2 | 165 | 10 |
| | 3 | 165 | 108 |
| | 4 | 158 | 20 |
| **Total** | **8** | **1321** | **422** |

SPAI technique can be coupled with these preconditioners for further efficiency. We plan to look at this aspect as part of our future work.

The next MOR algorithm under-consideration is BIRKA (Algorithm 2). Here, the linear solver would be chosen in a manner similar to AIRGA above. However, the coefficient matrices here have a block form, and hence, instead of standard Krylov subspace methods, their block variants should be used [52].

For the sake of brevity, the reuse of SPAI preconditioners in BIRKA (Algorithm 2) is discussed as part of Appendix A. Here also, the block structure can be exploited in developing a more efficient preconditioner. An example of this is given in Chapter 11 of [53], where a preconditioner similar to SPAI has been improved upon by utilizing such a structure. This aspect is part of our future work.

The third and the final MOR algorithm under-consideration is QB-IHOMM (Algorithm 3). As for the earlier two algorithms, any general Krylov subspace solver can be used here.

**Algorithm 4** AIRGA With Reuse of SPAI Preconditioner

```
1:  $z = 1$
2:  while no convergence do
3:      if $z == 1$ then
4:          for $i = 1, \ldots, \ell$ do
5:              $A_i^{(1)} = \left( \left( s_i^{(1)} \right)^2 M + \left( s_i^{(1)} \right) D + K \right)$
6:              if $i == 1$ then
7:                  Compute initial $P_1^{(1)}$ by solving
                        $\min_{P_1^{(1)}} \|I - A_1^{(1)} P_1^{(1)}\|_f^2$
                    (First-time; no earlier preconditioner)
8:                  $A_1^{(1)} P_1^{(1)} \mathfrak{X}^{(1)}(s_1) = F$
9:              else
10:                 Compute $Q_i^{(1)}$ by solving
                        $\min_{Q_i^{(1)}} \|A_{i-1}^{(1)} - A_i^{(1)} Q_i^{(1)}\|_f^2$
                    (Reuse along horizontal direction)
11:                 $A_i^{(1)} [Q_i^{(1)} \cdots Q_2^{(1)} P_1^{(1)}] \mathfrak{X}^{(1)}(s_i) = F$
12:             end if
13:         end for
14:     else
15:         for $i = 1, \ldots, \ell$ do
16:             $A_i^{(z)} = \left( \left( s_i^{(z)} \right)^2 M + \left( s_i^{(z)} \right) D + K \right)$
17:             if $i == 1$ then
18:                 Compute $Q_1^{(z)}$ by solving
                        $\min_{Q_1^{(z)}} \|A_1^{(z-1)} - A_1^{(z)} Q_1^{(z)}\|_f^2$
                    (Reuse along vertical direction)
19:                 $A_1^{(z)} \left[ Q_1^{(z)} \ldots Q_1^{(2)} P_1^{(1)} \right] \mathfrak{X}^{(z)}(s_1) = F$
20:             else
21:                 Compute $Q_i^{(z)}$ by solving
                        $\min_{Q_i^{(z)}} \|A_{i-1}^{(z)} - A_i^{(z)} Q_i^{(z)}\|_f^2$
                    (Reuse along horizontal direction)
22:                 $A_i^{(z)} \Big[ \underbrace{Q_i^{(z)} \cdots Q_2^{(z)}}$
23:                     $\underbrace{Q_1^{(z)} \ldots Q_1^{(2)}} P_1^{(1)} \Big] \mathfrak{X}^{(z)}(s_i) = F$
24:             end if
25:         end for
26:     end if
27:     "All the given set of expansion points
         (i.e. $s_1, s_2, \ldots, s_\ell$) are updated"
28:     $z = z + 1$
29: end while
    Note: The minimization problems at lines 7, 10, 18 and
    21 are solved as n independent least square problems
    (see (7)).
```

**TABLE 5.** Condition numbers of the coefficient matrices before and after application of SPAI[¶] for the academic disk brake model.

| AIRGA Iterations ($z$) | Expansion Points ($s_i$) | Before Precond. | After Precond. |
|---|---|---|---|
| 1 | 1 | $6.4 \times 10^{06}$ | $3.3 \times 10^{03}$ |
| | 2 | $4.6 \times 10^{06}$ | $2.9 \times 10^{03}$ |
| | 3 | $2.7 \times 10^{06}$ | $2.2 \times 10^{03}$ |
| | 4 | $1.8 \times 10^{06}$ | $1.6 \times 10^{03}$ |
| 2 | 1 | $2.0 \times 10^{06}$ | $1.3 \times 10^{03}$ |
| | 2 | $3.6 \times 10^{06}$ | $2.6 \times 10^{03}$ |
| | 3 | $5.8 \times 10^{06}$ | $5.1 \times 10^{03}$ |
| | 4 | $7.0 \times 10^{06}$ | $3.2 \times 10^{03}$ |

[¶] SPAI and Reusable SPAI improve the condition number almost equally.

Further, as in the case of AIRGA, the linear systems at lines 4 and 10 belong to the class of general shifted linear systems [48]–[50] and can be solved simultaneously, which is part of our future work[4].

For the sake of brevity, the above theory of reusing SPAI preconditioners applied to QB-IHOMM is discussed briefly in Appendix B. Again, our SPAI reuse theory can be coupled with the specific preconditioners for these kind of systems (e.g., shift-invariant preconditioners [49], [51]) to develop a more efficient preconditioning strategy. We plan to look at this aspect as part of our future work.

## IV. NUMERICAL EXPERIMENTS

For supporting our proposed preconditioned iterative solver theory using AIRGA [15], we perform experiments on two models. The first is a macroscopic equations of motion model (i.e. academic disk brake $M_0$) [55], and is discussed in Section IV-A. The second is also a similar model, however, this is a real-life industrial problem (i.e. industrial disk brake $M_1$) [55]. The experiments on this model are discussed in Section IV-B. These models are described by the following set of equations [55]:

$$\begin{aligned} M_\Omega \ddot{x}(t) &= -D_\Omega \dot{x}(t) - K_\Omega x(t) + Fu(t), \\ y(t) &= C^T x(t), \end{aligned} \tag{9}$$

where $M_\Omega = M$, $K_\Omega = K_E + K_R + \Omega^2 K_G$, $D_\Omega = \alpha M_\Omega + \beta K_\Omega$ (case of proportionally damped system; as needed for AIRGA) with commonly used parameter values as $\Omega = 2\pi$, $\alpha = 5 \times 10^{-02}$, and $\beta = 5 \times 10^{-06}$. Further, $F \in \mathbb{R}^n$ and $C^T \in \mathbb{R}^n$ are taken as $[1\ 0\ \cdots\ 0]^T$, which is the most frequently used choice. We take four expansion points linearly spaced between 1 and 500 based upon experience.

Although our purpose is to just reuse SPAI in AIRGA (Algorithm 4), we also execute original SPAI in AIRGA (Algorithm 1) for comparison. In Algorithms 1 and 4, at line 2 the overall iteration (`while-loop`) terminates when the change in the reduced model (computed as $H_2$-error between the reduced models at two consecutive AIRGA iterations) is less than a certain tolerance. We take this tolerance as $10^{-04}$ based upon the values in [15]. There is one more stopping criteria in Algorithms 1 at line 8 (also in Algorithm 4 but not listed here). This checks the $H_2$-error between two temporary reduced models. We take this tolerance as $10^{-06}$, again based upon the values in [15]. Since this is an adaptive algorithm, the optimal size of the reduced model is determined by the algorithm itself, and is denoted by $r$.

The linear systems that arise here have non-symmetric matrices. There are many iterative methods available for solving such linear systems. We use the Generalized Minimal Residual (GMRES) method [45] because it is very popular [56]. The stopping tolerance in GMRES is taken as $10^{-06}$, which is a common standard. As mentioned in Introduction, for both the given models, we observe that unpreconditioned GMRES fails to converge. Hence, we use the SPAI preconditioner as described above (without and with reuse).

As mentioned earlier, without loss of generality, we perform right preconditioning. To demonstrate the effectiveness of our theory for all types of preconditioning, for the academic disk model, we give data corresponding to left preconditioning as well.

We use Modified Sparse Approximate Inverse (MSPAI 1.0) proposed in [38] as our preconditioner. This is because MSPAI uses a linear algebra library for solving sparse least square problems that arise here. We use standard initial settings of MSPAI $\big($i.e. tolerance (ep) of $10^{-04}\big)$.

We perform our numerical experiments on a machine with the following configuration: Intel Xeon (R) CPU E5-1620 V3 @ 3.50 GHz., frequency 1200 MHz., 8 CPU and 64 GB

**TABLE 6.** GMRES computation time for the academic disk brake model.

| AIRGA Iterations ($z$) | No. of Linear Solves | GMRES Iterations per Linear Solve | GMRES Time when Using SPAI (Seconds) | GMRES Time when Using Reusable SPAI (Seconds) |
|---|---|---|---|---|
| 1 | 10 | 271 | 7.98 | 8.62 |
| 2 | 13 | 270 | 8.12 | 8.93 |
| **Total** | **23** | $10 \times 271 + 13 \times 270$ $= \mathbf{6220}$ | $10 \times 7.98 + 13 \times 8.12$ $= \mathbf{185}$ | $10 \times 8.62 + 13 \times 8.93$ $= \mathbf{202}$ |

**TABLE 7.** GMRES with SPAI and reusable SPAI computation time for the academic disk brake model.

| AIRGA Iterations $(z)$ | GMRES Plus SPAI Time (Seconds) | GMRES Plus Reusable SPAI Time (Seconds) |
|---|---|---|
| 1 | 748 | 306 |
| 2 | 759 | 318 |
| **Total** | **1507** | **624** |

RAM. All the codes are written in MATLAB (2016b) (including AIRGA, GMRES) except SPAI and reusable SPAI. MATLAB is used because of ease of rapid prototyping. Computing SPAI and reusable SPAI in MATLAB is expensive, therefore, we use C++ version of these (SPAI is from MSPAI and reusable SPAI is written by us). MSPAI further uses BLAS, LAPACK and ATLAS libraries.

It is important to emphasize that we do not integrate our MATLAB code base with the C++ based preconditioner. This is because integrating the two is complicated and is not needed here as well.

We compute SPAI and reusable SPAI in-parallel, separately, and save them on the hard-disk in the standard .mtx files [38]. When we run our MATLAB code base, then these files are read from the hard-disk into the main memory and converted into .mat files for further processing.

## A. ACADEMIC DISK BRAKE MODEL

This model is of size $4,669$. Based upon experience, the maximum reduced system size ($r_{max}$) is taken as 20. As mentioned earlier, however, due to the adaptive nature of AIRGA, we obtain a reduced system of size $r = 13$. For this model, AIRGA takes two outer iterations (line 2 of Algorithms 1 and 4) to converge (i.e. $\mathfrak{z} = 2$).

Reusing the SPAI preconditioner is beneficial when the values of $\|I - A_i^{(z)}\|_f / \|I\|_f$ is large, and the values of $\|A_{i-1}^{(z)} - A_i^{(z)}\|_f / \|A_{i-1}^{(z)}\|_f$ and $\|A_1^{(z-1)} - A_1^{(z)}\|_f / \|A_1^{(z-1)}\|_f$ are small, which is true in this case (see Table 3). In this table,

**TABLE 8.** SPAI and reusable SPAI computation time for the industrial disk brake model.

| AIRGA Iterations $(z)$ | Expansion Points $(s_i)$ | SPAI Case | Reusable SPAI Case | |
|---|---|---|---|---|
| | | Standard | Horizontal | Vertical |
| | | $\frac{\|I - A_i^{(z)}\|_f}{\|I\|_f}$ | $\frac{\|A_{i-1}^{(z)} - A_i^{(z)}\|_f}{\|A_{i-1}^{(z)}\|_f}$ | $\frac{\|A_1^{(z-1)} - A_1^{(z)}\|_f}{\|A_1^{(z-1)}\|_f}$ |
| 1 | 1 | $6.54 \times 10^{08}$ | *NA* | *NA* |
| | 2 | $6.54 \times 10^{08}$ | $3.74 \times 10^{-05}$ | *NA* |
| | 3 | $6.54 \times 10^{08}$ | $7.49 \times 10^{-05}$ | |
| | 4 | $6.54 \times 10^{08}$ | $1.12 \times 10^{-04}$ | |
| 2 | 1 | $1.31 \times 10^{09}$ | *NA* | 1.006 |
| | 2 | $6.65 \times 10^{08}$ | 0.49 | *NA* |
| | 3 | $6.53 \times 10^{08}$ | 0.50 | |
| | 4 | $6.56 \times 10^{08}$ | 0.49 | |
| 3 | 1 | $1.30 \times 10^{09}$ | *NA* | 1.009 |
| | 2 | $7.01 \times 10^{08}$ | 0.4658 | *NA* |
| | 3 | $6.53 \times 10^{08}$ | 0.5499 | |
| | 4 | $6.63 \times 10^{08}$ | 0.4940 | |
| 4 | 1 | $1.31 \times 10^{09}$ | *NA* | 1.0015 |
| | 2 | $6.86 \times 10^{08}$ | 0.4641 | *NA* |
| | 3 | $6.53 \times 10^{08}$ | 0.5002 | |
| | 4 | $6.56 \times 10^{08}$ | 0.4933 | |

**TABLE 9. SPAI and reusable SPAI computation time for the industrial disk brake model.**

| AIRGA Iterations ($z$) | Expansion Points ($s_i$) | SPAI[§] | Reusing SPAI[§] |
|---|---|---|---|
| 1 | 1 | 10 hrs | 10 hrs |
| | 2 | 10 hrs | 1 hr |
| | 3 | 10 hrs | 1 hr |
| | 4 | 10 hrs | 1 hr |
| 2 | 1 | 10 hrs | 1 hr 30 mins |
| | 2 | 10 hrs | 1 hr |
| | 3 | 10 hrs | 1 hr |
| | 4 | 10 hrs | 1 hr |
| 3 | 1 | 10 hrs | 1 hr 30 mins |
| | 2 | 10 hrs | 1 hour |
| | 3 | 10 hrs | 1 hour |
| | 4 | 10 hrs | 1 hour |
| 4 | 1 | 10 hrs | 1 hr 30 mins |
| | 2 | 10 hrs | 1 hr |
| | 3 | 10 hrs | 1 hr |
| | 4 | 10 hrs | 1 hr |
| **Total** | **16** | **160 hrs** | **26 hrs 30 mins** |

[§] All times given here differ in seconds (not evident because of the rounding to the nearest minute).

**TABLE 10. Condition numbers of the coefficient matrices before and after application of SPAI for the industrial disk brake model.**

| AIRGA Iterations ($z$) | Expansion Points ($s_i$) | Before Precond. | After Precond. |
|---|---|---|---|
| 1 | 1 | $1.4 \times 10^{16}$ | $1.4 \times 10^{09}$ |
| | 2 | $1.5 \times 10^{16}$ | $1.6 \times 10^{09}$ |
| | 3 | $1.6 \times 10^{16}$ | $2.5 \times 10^{09}$ |
| | 4 | $1.8 \times 10^{16}$ | $2.8 \times 10^{09}$ |
| 2 | 1 | $9.6 \times 10^{16}$ | $8.1 \times 10^{09}$ |
| | 2 | $3.4 \times 10^{16}$ | $2.4 \times 10^{09}$ |
| | 3 | $5.8 \times 10^{16}$ | $9.3 \times 10^{09}$ |
| | 4 | $2.2 \times 10^{16}$ | $3.4 \times 10^{09}$ |
| 3 | 1 | $7.5 \times 10^{16}$ | $1.9 \times 10^{09}$ |
| | 2 | $2.5 \times 10^{16}$ | $2.5 \times 10^{09}$ |
| | 3 | $5.0 \times 10^{16}$ | $6.1 \times 10^{09}$ |
| | 4 | $6.9 \times 10^{16}$ | $4.7 \times 10^{09}$ |
| 4 | 1 | $8.9 \times 10^{16}$ | $3.5 \times 10^{09}$ |
| | 2 | $9.5 \times 10^{16}$ | $8.3 \times 10^{09}$ |
| | 3 | $5.4 \times 10^{16}$ | $5.8 \times 10^{09}$ |
| | 4 | $1.3 \times 10^{16}$ | $5.4 \times 10^{09}$ |

columns 1 and 2 list the AIRGA iterations and the four expansion points, respectively. The above three quantities are listed in columns 3, 4 and 5, respectively. For the first AIRGA iteration and the first expansion point, SPAI preconditioner cannot be reused because there is no earlier preconditioner (mentioned as *NA* in table). From the second expansion point (and the first AIRGA iteration), we perform horizontal reuse of preconditioner (see Figure 1). This is the same for the second AIRGA iteration as well. Vertical reuse of preconditioner is done only for the first expansion point (and the second AIRGA iteration; again see Figure 1).

In Table 4, we compare the SPAI and the reusable SPAI timings. As for Table 3, here columns 1 and 2 list the AIRGA iterations and the four expansion points, respectively. SPAI and reusable SPAI computation times are given in columns 3 and 4, respectively. At the first AIRGA iteration and the first expansion point, both SPAI and reusable SPAI take the same computation time. This is because, as above, reusing of SPAI preconditioner is not applicable here. From the second expansion point of the first AIRGA iteration, we see substantial savings because of the reuse of the SPAI preconditioner (approximately 68%).

Before presenting GMRES data, we would like to discuss improvements in the condition numbers of the coefficient matrices because of the preconditioning. This data is given in Table 5. As evident, preconditioning does substantially improve the quality of the coefficient matrices.

Table 6 provides the iteration count and the computation time of GMRES. Here, we only provide GMRES execution details since the computation time of preconditioner has been discussed above. In this table, column 1 lists the AIRGA iterations. The number of linear solves and average GMRES iterations per linear solve are given in columns 2 and 3, respectively. Finally, columns 4 and 5 list the computation times of GMRES when using SPAI and reusable SPAI, respectively. We notice from this table that solving linear systems by GMRES with SPAI takes less computation time as compared to solving them by GMRES with reusable SPAI. This is because when we reuse the SPAI preconditioner in GMRES, additional matrix-vector products are performed, however, this extra cost is almost negligible when compared to the savings in the preconditioner computation time for the latter case (as evident in Table 3 above; also see total GMRES and preconditioner time below).

As earlier, the data in Table 6 is corresponding to right preconditioning. In the case of left preconditioning we see only a modest change in the metrics under-consideration. That is, the total GMRES iterations, the total GMRES plus SPAI time, and the total GMRES plus reusable SPAI time are 6364, 190, and 204, respectively.

Table 7 gives the computation time of GMRES plus SPAI (column 2) and GMRES plus reusable SPAI (column 3) at each AIRGA iteration (column 1). As evident from this table, reusing the SPAI preconditioner leads to about 60% savings in total time required for solving all the linear systems.

**TABLE 11.** GMRES computation time for the industrial disk brake model.

| AIRGA Iterations ($z$) | No. of Linear Solves | GMRES Iterations per Linear Solve | GMRES Time when Using SPAI (Minutes) | GMRES Time when Using Reusable SPAI (Minutes) |
|---|---|---|---|---|
| 1 | 64 | 421 | 08 | 10 |
| 2 | 64 | 426 | 09 | 11 |
| 3 | 64 | 429 | 10 | 12 |
| 4 | 52 | 432 | 10 | 12 |
| **Total** | **244** | $64 \times (421 + 426 + 429)$ $+ 52 \times 432$ $= \mathbf{104,128}$ | $64 \times (08 + 09 + 10)$ $+ 52 \times 10$ $= \mathbf{2248}$ | $64 \times (10 + 11 + 12)$ $+ 52 \times 12$ $= \mathbf{2736}$ |

## B. INDUSTRIAL DISK BRAKE MODEL

This model is of size 1.2 million. Based upon experience, the maximum reduced system size ($r_{max}$) is taken as 100. As mentioned earlier, however, due to the adaptive nature of AIRGA, we obtain a reduced system of size $r = 52$. For this model, AIRGA takes four outer iterations (line 2 of Algorithms 1 and 4) to converge (i.e. $\mathfrak{z} = 4$).

Again, reusing the SPAI preconditioner is beneficial when the value of $\|I - A_i^{(z)}\|_f / \|I\|_f$ is large, and the value of $\|A_{i-1}^{(z)} - A_i^{(z)}\|_f / \|A_{i-1}^{(z)}\|_f$ and $\|A_1^{(z-1)} - A_1^{(z)}\|_f / \|A_1^{(z-1)}\|_f$ are small, which is true in this case (see Table 8). The structure of this table is same as Table 3. As earlier, for the first AIRGA iteration and the first expansion point, SPAI preconditioner cannot be reused because there is no earlier preconditioner (mentioned as *NA* in table). From the second expansion point (and the first AIRGA iteration), we perform horizontal reuse of preconditioner (see Figure 1). This is the same for the second, the third and the fourth AIRGA iterations as well. Vertical reuse of preconditioner is done only for the first expansion point (and the second, the third, and the fourth AIRGA iterations; again see Figure 1).

In Table 9, we compare the SPAI and the reusable SPAI timings. The structure of this table is same as that of Table 4. As before, at the first AIRGA iteration and the first expansion point, both SPAI and reusable SPAI take the same computation time. This is because, as above, reusing of SPAI preconditioner is not applicable here. From the second expansion point of the first AIRGA iteration, we see substantial savings because of the reuse of the SPAI preconditioner (from 160 hours to 26 hrs 30 minutes; approximately 83%).

As in the case of the academic disk model, here too before presenting GMRES data, we would like to discuss improvements in the condition numbers of the coefficient matrix because of the preconditioning. This data is given in Table 10. As evident, preconditioning does substantially improve the quality of the coefficient matrices.

Table 11 provides the iteration count and the computation time of GMRES. Here, again we have only provided GMRES execution details since the computation time of the

**TABLE 12.** GMRES with SPAI and reusable SPAI computation time for the industrial disk brake model.

| AIRGA Iterations ($z$) | GMRES plus SPAI Time | GMRES plus Reusable SPAI Time |
|---|---|---|
| 1 | 48 hrs 32 mins | 23 hrs 40 mins |
| 2 | 49 hrs 36 mins | 16 hrs 14 mins |
| 3 | 50 hrs 40 mins | 17 hrs 18 mins |
| 4 | 48 hrs 40 mins | 14 hrs 54 mins |
| **Total** | **197 hrs 28 mins** | **72 hrs 06 mins** |

preconditioner has already been discussed above. The structure of this table is same as that of Table 6. As earlier, we notice from this table that solving linear systems by GMRES with SPAI takes less computation time as compared to solving them by GMRES with reusable SPAI. This is again because of additional matrix-vector products in the reusable SPAI case. Here also, this extra cost is almost negligible when compared to the savings in the preconditioner computation time (as evident in Table 9; also see the total GMRES and preconditioner time below).

Table 12 gives the computation time of GMRES plus SPAI (column 2) and GMRES plus reusable SPAI (column 3) at each AIRGA iteration (column 1). As before, it is evident from this table, reusing the SPAI preconditioner leads to about 64% savings in total time (from 197 hours 28 minutes to 72 hours 06 minutes).

To demonstrate the quality of the reduced system, we plot the relative $H_2$ error between the transfer function of the original system and the reduced system with respect to the different expansion points (in Figure 3). The reduced system considered here is obtained by using GMRES with reusable SPAI. These expansion points, denoted by $S$, are computed as $2\pi f$, where the frequency variable $f$ is linearly spaced between 1 and 500. As evident from this figure, the obtained reduced system is good (the error is very small). Further, we also observe from this figure that the reduced model is most accurate in 7–10 range of the expansion points. This is

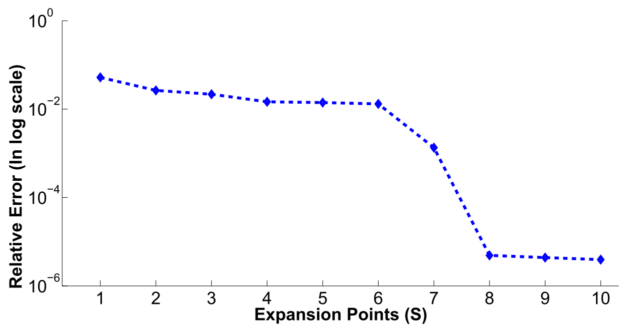

**FIGURE 3.** **Relative error between the original and reduced system for the industrial disk brake model.**

because the final expansion points, upon the convergence of AIRGA, lie in this range.

## V. CONCLUSIONS AND FUTURE WORK

In this work, we have focused on MOR of non-parametric dynamical systems, specifically on the following three algorithms: AIRGA, BIRKA, and QB-IHOMM. Since solving large and sparse linear systems is a bottleneck in scaling these MOR algorithms for reduction of large sized dynamical systems, we have proposed reusing of the SPAI preconditioner.

Specifically, we have demonstrated the following: exploitation of the simplicity because of the lack of parameters in reusing preconditioners, multiple ways of reusing preconditioners within the algorithm, efficient implementation to ensure that the savings because of reusing preconditioners are not negated by bad coding, and experimentation on a massively large industrial problem. Numerical experiments show the effectiveness of our approach, where for a problem of size 1.2 million, we save up to 64% in the computation time. In absolute terms, this gives a saving of 5 days.

Our future work consists of three main directions, focusing on other efficient MOR algorithms, better linear solvers and enhanced preconditioning techniques.

First, we will investigate other important variants of MOR algorithms discussed in our paper (AIRGA, BIRKA, QB-IHOMM). For example, T-BIRKA is a more efficient version of BIRKA [17], and applying our techniques here would be useful.

Second, in our paper, we have used a basic and common linear solver (GMRES). However, as discussed earlier, this aspect can also be optimized. Specifically, for the class of MOR algorithms to which AIRGA and QB-IHOMM belong, we will investigate the use of linear solvers specific to general shifted linear systems [48]–[50].

Finally and third, we will investigate more sophisticated preconditioning strategies that will further exploit the properties of the underlying MOR algorithms as well as the resulting linear systems. Specifically, we will explore five directions as below.

(a) Besides the currently used basic SPAI preconditioner, we will investigate the use of high-order convergent approximate inverse preconditioners [41], [42] as well as hybrid versions, which use a combination of factorization and approximate inverse techniques [40], [43].

(b) For AIRGA, QB-IHOMM, and related MOR algorithms where general shifted linear systems arise, we will investigate the use of our reusable SPAI preconditioner along with shift-invariant preconditioners that have been developed specifically for such shifted linear systems [49], [51].

(c) We will investigate exploiting the block structure of the linear system coefficient matrices in BIRKA such that the SPAI and its reuse can be done more efficiently [53].

(d) Since randomized preconditioners have shown promising results in recent years, we will explore their use in the context of linear systems in MOR algorithms.

(e) Finally, we would also investigate combining machine learning techniques (e.g., spiking neural networks) to optimize the parameters inside the preconditioners.

## APPENDIX A

In the Algorithm 2, we solve linear systems of equations at lines 4 and 5. We first apply our proposed theory of reusing preconditioners to line 4, which is given as

$$
\begin{aligned}
&vec\,(V)\\
&= \left(-\Lambda \otimes I_n - I_r \otimes K - \sum_{\mathrm{j}=1}^{m} \check{\check{N}}_{\mathrm{j}}^{T} \otimes N_{\mathrm{j}}\right)^{-1} \left(\check{\check{F}}^{T} \otimes F\right) vec(I_m).
\end{aligned}
$$

Here, $\Lambda$ is a diagonal matrix comprising of interpolation points, which is updated at the start of the `while` loop at line 2. For ease of explanation, we take $j = 1$ here. Similar steps can be executed for $j = 2, \ldots, m$. Let $A_{z-1} = -\Lambda_{z-1} \otimes I_n - I_r \otimes K - \left(\check{\check{N}}_1^T\right)_{z-1} \otimes N_1$ and $A_z = -\Lambda_z \otimes I_n - I_r \otimes K - \left(\check{\check{N}}_1^T\right)_z \otimes N_1$ be the coefficient matrices corresponding to $\Lambda_{z-1}$ and $\Lambda_z$, respectively (for $z = 1, \ldots, \mathfrak{z}$ (until covergence)). Expressing $A_z$ in terms of $A_{z-1}$, we get

$$
\begin{aligned}
A_z = A_{z-1}\Bigg(& I_{nr} + A_{z-1}^{-1}(-\Lambda_z \otimes I_n) + A_{z-1}^{-1}(\Lambda_{z-1} \otimes I_n)\\
&+ A_{z-1}^{-1}\left(-\left(\check{\check{N}}_1^T\right)_z \otimes N_1\right) + A_{z-1}^{-1}\left(\left(\check{\check{N}}_1^T\right)_{z-1} \otimes N_1\right)\Bigg),
\end{aligned}
$$

where $I_{nr} \in \mathbb{R}^{n \cdot r \times n \cdot r}$ is the Identity matrix. If we define

$$
\begin{aligned}
Q_z = \Bigg(& I_{nr} + A_{z-1}^{-1}(-\Lambda_z \otimes I_n) + A_{z-1}^{-1}(\Lambda_{z-1} \otimes I_n)\\
&+ A_{z-1}^{-1}\left(-\left(\check{\check{N}}_1^T\right)_z \otimes N_1\right) + A_{z-1}^{-1}\left(\left(\check{\check{N}}_1^T\right)_{z-1} \otimes N_1\right)\Bigg)^{-1},
\end{aligned}
$$

then above is equivalent to

$$A_z = A_{z-1} Q_z^{-1}. \tag{10}$$

Now, we enforce

$$A_{z-1} P_{z-1} = A_z P_z. \tag{11}$$

Using (10), instead we enforce

$$A_{z-1} P_{z-1} = A_{z-1} Q_z^{-1} P_z.$$

If we take $P_z = Q_z P_{z-1}$, then we eventually enforce

$$A_{z-1}P_{z-1} = A_{z-1}Q_z^{-1}Q_zP_{z-1},$$

which is true.

Thus, instead of solving for $P_z$ by enforcing (11), which is harder to solve, we obtain the preconditioner at the $z^{th}$ iteration ($P_z = Q_zP_{z-1}$) by enforcing

$$A_{z-1}P_{z-1} = A_zQ_zP_{z-1},$$

which is more easily solvable. The remaining derivation here is same as earlier (see Section III-C). We reuse preconditioners at line 5 similarly.

## APPENDIX B

In the Algorithm 3, we solve linear systems of equations at line 4 and 10. Again, we first apply our proposed theory of reusing preconditioners to line 4, which is given as

$$X_j(\sigma_i) = [(\sigma_iD - K)^{-1}D]^j(\sigma_iD - K)^{-1}F,$$
$$\text{for } j = 1, \ldots, P + Q \text{ and } i = 1, \ldots, \ell.$$

Let $A_{i-1} = \sigma_{i-1}D - K$ and $A_i = \sigma_iD - K$ be the two coefficient matrices for different interpolation points $\sigma_{i-1}$ and $\sigma_i$, respectively (for $i = 1, \ldots, \ell$). Expressing $A_i$ in terms of $A_{i-1}$, we get

$$A_i = A_{i-1}(I + (\sigma_i - \sigma_{i-1})A_{i-1}^{-1}D).$$

If we define $Q_i = (I + (\sigma_i - \sigma_{i-1})A_{i-1}^{-1}D)^{-1}$, then above is equivalent to

$$A_i = A_{i-1}Q_i^{-1}.$$

As for AIRGA and BIRKA, instead of obtaining $P_i$ by enforcing

$$A_{i-1}P_{i-1} = A_iP_i,$$

which is harder to solve, we obtain the preconditioner at the $i^{th}$ iteration ($P_i = Q_iP_{i-1}$) by enforcing

$$A_{i-1}P_{i-1} = A_iQ_iP_{i-1},$$

which is more easily solvable. Again, here also, the remaining derivation is same as earlier (see Section III-C). We reuse preconditioners at line 10 similarly.

## ACKNOWLEDGMENT

The authors would like to deeply thank Prof. Dr. Heike Faßbender (at Institut Computational Mathematics, AG Numerik, Technische Universität Braunschweig, Germany) for discussions and help regarding different aspects of this project. They would also like to thank Ms. Apoorva Joshi (IIT Indore, India) for help in numerical experiments, which she did as part of her undergraduate thesis.

Thanks to the anonymous reviewers that helped them greatly improve the quality of this manuscript. Finally, they would like to thank Dr. Yan-Jun Liu (Editor handling our manuscript) for his tremendous support during the whole reviewing process.

## REFERENCES

[1] O. Katsuhiko, *Modern Control Engineering*. Upper Saddle River, NJ, USA: Prentice-Hall, 2001.

[2] M. Rewienski and J. White, "A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 2, pp. 155–170, Feb. 2003.

[3] A. C. Antoulas, "Approximation of large-scale dynamical systems: An overview," *IFAC Proc. Volumes*, vol. 37, no. 11, pp. 19–28, Jul. 2004.

[4] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*. Philadelphia, PA, USA: SIAM, 2005.

[5] J. E. S. Socolar, "Nonlinear dynamical systems," in *Complex Systems Science in Biomedicine*, T. S. Deisboeck and J. Y. Kresh Eds. Boston, MA, USA: Springer, 2006, pp. 115–140.

[6] B. N. Bond, "Parameterized model order reduction for nonlinear dynamical systems," M.S. thesis, Dept. Elect. Eng. Comput. Sci., MIT, Cambridge, MA, USA, 2006.

[7] E. J. Grimme, "Krylov projection methods for model reduction," Ph.D. dissertation, Dept. Elect. Eng., Univ. Illinois Urbana-Champaign, Urbana, IL, USA, 1997.

[8] S. Gugercin, "Projection methods for model reduction of large-scale dynamical systems," Ph.D. dissertation, Dept. Elect. Comp. Eng., Rice Univ., Houston, TX, USA, 2003.

[9] W. H. Schilders, H. A. Van der Vorst, and J. Rommes, *Model Order Reduction: Theory, Research Aspects and Applications*, vol. 13. Berlin, Germany: Springer, 2008.

[10] S. Gugercin, A. C. Antoulas, and C. Beattie, "$H_2$ model reduction for large-scale linear dynamical systems," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 609–638, 2008.

[11] T. Breiten, "Interpolation methods for model reduction of large-scale dynamical systems," Ph.D. dissertation, Dept. Math., Otto-von-Guericke-Universität Magdeburg, Magdeburg, Germany, 2013.

[12] S. A. Wyatt, "Issues in interpolatory model reduction: Inexact solves, second-order systems and DAEs," Ph.D. dissertation, Dept. Math., Virginia Tech, Blacksburg, VA, USA, 2012.

[13] Z.-Y. Qiu, Y.-L. Jiang, and J.-W. Yuan, "Interpolatory model order reduction method for second order systems," *Asian J. Control*, vol. 20, no. 1, pp. 312–322, Jan. 2018.

[14] Z. Bai and Y. Su, "Dimension reduction of large-scale second-order dynamical systems via a second-order Arnoldi method," *SIAM J. Sci. Comput.*, vol. 26, no. 5, pp. 1692–1709, Jan. 2005.

[15] T. Bonin, H. Faßbender, A. Soppa, and M. Zaeh, "A fully adaptive rational global Arnoldi method for the model-order reduction of second-order MIMO systems with proportional damping," *Math. Comput. Simul.*, vol. 122, pp. 1–19, Apr. 2016.

[16] P. Benner and T. Breiten, "Interpolation-based $H_2$-model reduction of bilinear control systems," *SIAM J. Matrix Anal. Appl.*, vol. 33, no. 3, pp. 859–885, Aug. 2012.

[17] G. M. Flagg, "Interpolation methods for the model reduction of bilinear systems," Ph.D. dissertation, Dept. Math., Virginia Tech, Blacksburg, VA, USA, 2012.

[18] R. Choudhary and K. Ahuja, "Inexact linear solves in model reduction of bilinear dynamical systems," *IEEE Access*, vol. 7, pp. 72297–72307, May 2019.

[19] M. M. A. Asif, M. I. Ahmad, P. Benner, L. Feng, and T. Stykel, "Implicit higher-order moment matching technique for model reduction of quadratic-bilinear systems," 2019, *arXiv:1911.05400*. [Online]. Available: http://arxiv.org/abs/1911.05400

[20] U. Baur, C. Beattie, P. Benner, and S. Gugercin, "Interpolatory projection methods for parameterized model reduction," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2489–2518, Jan. 2011.

[21] P. Benner and L. Feng, "A robust algorithm for parametric model order reduction based on implicit moment matching," in *Reduced Order Methods for Modeling and Computational Reduction*, A. Quarteroni and G. Rozza, Eds. Cham, Switzerland: Springer, 2014, pp. 159–185.

[22] L. Feng, P. Benner, and J. G. Korvink, "Subspace recycling accelerates the parametric macro-modeling of MEMS," *Int. J. Numer. Methods Eng.*, vol. 94, no. 1, pp. 84–110, Apr. 2013.

[23] A. C. Rodriguez, S. Gugercin, and J. Borggaard, "Interpolatory model reduction of parameterized bilinear dynamical systems," *Adv. Comput. Math.*, vol. 44, no. 6, pp. 1887–1916, Dec. 2018.

[24] X. Cao, "Optimal model order reduction for parametric nonlinear systems," Ph.D. dissertation, Dept. Math. Comp. Sci., TU Eindhoven, Eindhoven, The Netherlands, 2019.

[25] K. Ahuja, E. de Sturler, S. Gugercin, and E. R. Chang, "Recycling BiCG with an application to model reduction," *SIAM J. Sci. Comput.*, vol. 34, no. 4, pp. A1925–A1949, Jan. 2012.

[26] K. Ahuja, P. Benner, E. de Sturler, and L. Feng, "Recycling BiCGSTAB with an application to parametric model order reduction," *SIAM J. Sci. Comput.*, vol. 37, no. 5, pp. S429–S446, Jan. 2015.

[27] K. Ahuja, "Recycling Krylov subspaces and preconditioners," Ph.D. dissertation, Dept. Math., Virginia Tech, Blacksburg, VA, USA, 2011.

[28] K. Ahuja, B. K. Clark, E. de Sturler, D. M. Ceperley, and J. Kim, "Improved scaling for quantum Monte Carlo on insulators," *SIAM J. Sci. Comput.*, vol. 33, no. 4, pp. 1837–1859, Jan. 2011.

[29] S. Bellavia, V. De Simone, D. di Serafino, and B. Morini, "Efficient preconditioner updates for shifted linear systems," *SIAM J. Sci. Comput.*, vol. 33, no. 4, pp. 1785–1809, Jan. 2011.

[30] W.-H. Luo, T.-Z. Huang, L. Li, Y. Zhang, and X.-M. Gu, "Efficient preconditioner updates for unsymmetric shifted linear systems," *Comput. Math. with Appl.*, vol. 67, no. 9, pp. 1643–1655, May 2014.

[31] A. K. Grim-McNally, E. de Sturler, and S. Gugercin, "Preconditioning parametrized linear systems," 2016, *arXiv:1601.05883*. [Online]. Available: http://arxiv.org/abs/1601.05883

[32] A. K. Grim-McNally, "Reusing and updating preconditioners for sequences of matrices," M.S. thesis, Dept. Math., Virginia Tech, Blacksburg, VA, USA, 2015.

[33] N. P. Singh and K. Ahuja, "Preconditioned linear solves for parametric model order reduction," *Int. J. Comput. Math.*, vol. 97, no. 7, pp. 1484–1502, Jul. 2020.

[34] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Philadelphia, PA, USA: SIAM, 2003.

[35] H.-L. Shen, S.-Y. Li, and X.-H. Shao, "The NMHSS iterative method for the standard Lyapunov equation," *IEEE Access*, vol. 7, pp. 13200–13205, Jan. 2019.

[36] M. Benzi, "Preconditioning techniques for large linear systems: A survey," *J. Comput. Phys.*, vol. 182, no. 2, pp. 418–477, Nov. 2002.

[37] E. Chow and Y. Saad, "Approximate inverse preconditioners via sparse-sparse iterations," *SIAM J. Sci. Comput.*, vol. 19, no. 3, pp. 995–1023, May 1998.

[38] K. Alexander, "Modified sparse approximate inverses (MSPAI) for parallel preconditioning," Ph.D. dissertation, Dept. Math., TU Munich, Munich, Germany, 2008.

[39] M. Benzi and M. Tuma, "A sparse approximate inverse preconditioner for nonsymmetric linear systems," *SIAM J. Sci. Comput.*, vol. 19, no. 3, pp. 968–994, May 1998.

[40] S. C. Buranay and O. C. Iyikal, "Approximate Schur-block ILU preconditioners for regularized solution of discrete ill-posed problems," *Math. Problems Eng.*, pp. 1–18, Apr. 2019, Art. no. 1912535, doi: 10.1155/2019/1912535.

[41] S. C. Buranay, D. Subasi, and O. C. Iyikal, "On the two classes of high-order convergent methods of approximate inverse preconditioners for solving linear systems," *Numer. Linear Algebra Appl.*, vol. 24, no. 6, p. e2111, Dec. 2017.

[42] F. Soleymani, "A fast convergent iterative solver for approximate inverse of matrices," *Numer. Linear Algebra Appl.*, vol. 21, no. 3, pp. 439–452, May 2014.

[43] L. Y. Kolotilina and A. Y. Yeremin, "Factorized sparse approximate inverse preconditionings I. Theory," *SIAM J. Matrix Anal. Appl.*, vol. 14, no. 1, pp. 45–58, Jan. 1993.

[44] N. I. M. Gould and J. A. Scott, "Sparse approximate-inverse preconditioners using norm-minimization techniques," *SIAM J. Sci. Comput.*, vol. 19, no. 2, pp. 605–625, Mar. 1998.

[45] Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 7, no. 3, pp. 856–869, Jul. 1986.

[46] H. A. van der Vorst, "Bi-CGSTAB: A fast and smoothly converging variant of bi-CG for the solution of nonsymmetric linear systems," *SIAM J. Sci. Stat. Comput.*, vol. 13, no. 2, pp. 631–644, Mar. 1992.

[47] P. Sonneveld and M. B. Van Gijzen, "IDR(s): A family of simple and fast algorithms for solving large non-symmetric systems of linear equations," *SIAM J. Sci. Comput.*, vol. 31, no. 2, pp. 1035–1062, Jan. 2009.

[48] V. Simoncini, "Restarted full orthogonalization method for shifted linear systems," *BIT Numer. Math.*, vol. 43, no. 2, pp. 459–466, Jun. 2003.

[49] T. Bakhos, P. K. Kitanidis, S. Ladenheim, A. K. Saibaba, and D. B. Szyld, "Multipreconditioned GMRES for shifted systems," *SIAM J. Sci. Comput.*, vol. 39, no. 5, pp. S222–S247, Jan. 2017.

[50] X.-M. Gu, T.-Z. Huang, B. Carpentieri, A. Imakura, K. Zhang, and L. Du, "Efficient variants of the CMRH method for solving a sequence of multi-shifted non-Hermitian linear systems simultaneously," *J. Comput. Appl. Math.*, vol. 375, Sep. 2020, Art. no. 112788.

[51] X.-M. Gu, T.-Z. Huang, G. Yin, B. Carpentieri, C. Wen, and L. Du, "Restarted Hessenberg method for solving shifted nonsymmetric linear systems," *J. Comput. Appl. Math.*, vol. 331, pp. 166–177, Mar. 2018.

[52] V. Simoncini, "Computational methods for linear matrix equations," *SIAM Rev.*, vol. 58, no. 3, pp. 377–441, Jan. 2016.

[53] C. C. K. Mikkelsen, "Numerical methods for Lyapunov equations," Ph.D. dissertation, Dept. Math., Purdue Univ., Lafayette, IN, USA, 2009.

[54] O. Axelsson and A. Kucherov, "Real valued iterative methods for solving complex symmetric linear systems," *Numer. Linear Algebra Appl.*, vol. 7, no. 4, pp. 197–218, Jun. 2000.

[55] N. Gräbner, V. Mehrmann, S. Quraishi, C. Schröder, and U. von Wagner, "Numerical methods for parametric model reduction in the simulation of disk brake squeal," *J. Appl. Math. Mech.*, vol. 96, no. 12, pp. 1388–1405, Dec. 2016.

[56] T. Han and Y. Han, "Numerical solution for super large scale systems," *IEEE Access*, vol. 1, pp. 537–544, Aug. 2013.

**NAVNEET PRATAP SINGH** received the bachelor's degree in computer science and engineering from UPTU, Lucknow, India, and the master's degree in modeling and simulation from the Defence Institute of Advanced Technology, Pune, India. He is currently pursuing the Ph.D. degree with IIT Indore.

His thesis focuses on Stable Linear Solves with Preconditioner Updates for Model Reduction. His research interests include intersection of computer science and mathematics, especially numerical linear algebra, optimization, dynamical systems, and machine learning.

**KAPIL AHUJA** received the bachelor's degree from IIT (BHU), India, the double master's and Ph.D. degrees from Virginia Tech, Blacksburg, VA, USA, and the Postdoctoral training from the Max Planck Institute, Germany.

He is currently an Associate Professor in computer science and engineering with IIT Indore, India. In the past, he was a Visiting Professor at TU Braunschweig, Germany, TU Dresden, Germany, and Sandia National Labs, USA. He is also working on mathematics of data science as well as computational science. His research interests in artificial intelligence, machine learning, numerical methods, and optimization.