An Efficient, Memory-Saving Approach for the Loewner Framework

Davide Palitta[1] [iD] · **Sanda Lefteriu[2]**

Abstract

The Loewner framework is one of the most successful data-driven model order reduction techniques. If N is the cardinality of a given data set, the so-called Loewner and shifted Loewner matrices $\mathbb{L} \in \mathbb{C}^{N \times N}$ and $\mathbb{S} \in \mathbb{C}^{N \times N}$ can be defined by solely relying on information encoded in the considered data set and they play a crucial role in the computation of the sought rational model approximation.In particular, the singular value decomposition of a linear combination of $\mathbb{S}$ and $\mathbb{L}$ provides the tools needed to construct accurate models which fulfill important approximation properties with respect to the original data set. However, for highly-sampled data sets, the dense nature of $\mathbb{L}$ and $\mathbb{S}$ leads to numerical difficulties, namely the failure to allocate these matrices in certain memory-limited environments or excessive computational costs. Even though they do not possess any sparsity pattern, the Loewner and shifted Loewner matrices are extremely structured and, in this paper, we show how to fully exploit their Cauchy-like structure to reduce the cost of computing accurate rational models while avoiding the explicit allocation of $\mathbb{L}$ and $\mathbb{S}$. In particular, the use of the *hierarchically semiseparable* format allows us to remarkably lower both the computational cost and the memory requirements of the Loewner framework obtaining a novel scheme whose costs scale with $N \log N$.

✉ Davide Palitta
   palitta@mpi-magdeburg.mpg.de

   Sanda Lefteriu
   sanda.lefteriu@imt-lille-douai.fr

[1] Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, 39106 Magdeburg, Germany

[2] IMT Lille Douai, Institut Mines-Télécom, Univ. Lille, Centre for Education, Research and Innovation (CERI) Digital Systems, 59000 Lille, France

🍠 Springer

1 Introduction

The Loewner framework, originally proposed in [30] for solving the generalized realization problem coupled with tangential interpolation, was successfully employed for data-driven model order reduction from frequency domain data [26]. Measurements of the frequency response are available in several communities: electrical engineering (impedance, admittance or scattering parameters [26]), mechanical and civil engineering (structural and vibro-acoustic frequency response functions [35] or frequency response measurements of thermal systems [10]), to name a few. The first step in the Loewner framework consists in setting up the data matrices and building the Loewner and shifted Loewner matrices entry-wise based on the chosen partition into right and left data, followed by computing the singular value decomposition (SVD) of a linear combination of these matrices and forming the model by projection, using the dominant singular triplets. The main advantages of the Loewner framework over existing approaches are, on the one hand, its system identification capabilities, in the sense that the order of the system can be deduced from the singular value drop, and, on the other hand, its potential in dealing with systems with a large number of inputs and outputs efficiently thanks to the incorporation of tangential interpolation. The main drawbacks, however, are the large storage requirements paired with the significant CPU cost inherent to the full SVD computation for data sets with a large number of measurements (values in the range 10^5 are common in industrial applications). To bypass these inconveniences, greedy-type approaches were proposed in [26], thus reducing memory requirements, from $\mathcal{O}(N^2)$ for storing the dense Loewner and shifted Loewner matrices to $\mathcal{O}(N + n^2)$, and the computational cost, from $\mathcal{O}(N^3)$ for computing the SVD to $\mathcal{O}(Nn^3)$ and $\mathcal{O}(Nn^4)$, where N is the size of the data set and n is the order of the model.

Taking advantage of numerical linear algebra tools to reduce storage and computational requirements for the Loewner framework is another avenue worth exploring due to the inherent structure embedded in the albeit dense Loewner and shifted Loewner matrices. The factored ADI-Galerkin method for computing these matrices as solutions to certain Sylvester equations with a factored right-hand side was investigated in [18]. Such a scheme computes low-rank approximations to the dense Loewner matrix to speed-up the SVD computation. However, in [18] no results about the accuracy of the computed reduced models are reported. Moreover, the memory constraints coming from the allocation of $\mathbb{L}$ and $\mathbb{S}$ are still present. Alternatively, one can focus on accelerating solely the step of the SVD calculation by employing Krylov methods (see, e.g., [3, 19, 25, 41] to name a few), by using the randomized SVD [31] to compute the dominant singular triplets instead of the full SVD or other types of inexact SVD-type decompositions (adaptive cross approximation [4], particularly suited for hierarchical matrices, or a CUR decomposition [11] as in [22, 38]).

The novel approach proposed in this paper tackles the issue of the memory requirements, at the same time as reducing the CPU cost of the Loewner framework while maintaining the accuracy of the standard approach for large values of the number of measurements. As the Loewner and shifted Loewner matrices satisfy Sylvester equations with diagonal coefficient matrices, they are, in fact, Cauchy-like matrices, obtained as the Hadamard product between a Cauchy matrix $\mathcal{C}$ and low-rank right-hand sides. Extensive research has been devoted to fully exploiting the rich structure of Cauchy matrices. Several algorithms for computing the matrix-vector product $\mathcal{C}\mathbf{x}$ can be found in the literature and many avoid assembling the full matrix $\mathcal{C}$ (see, e.g., [7, 15, 17, 33]). Hierarchically semiseparable matrices (HSS) have been deemed efficient for approximating Cauchy matrices with a low off-diagonal rank [33, 34]. HSS and other rank-structured matrices are

widely used in developing fast algorithms for algebraic operations (matrix-vector multiplications, matrix factorizations, matrix inversion, etc., see, e.g., [8, 33, 34, 44, 47] and references therein) used as building blocks for the solution of certain problems like linear systems of equations [48], eigenvalue problems [45], linear and quadratic matrix equations [23, 27], and many more. For our application, the approximation of the Cauchy matrix in HSS format considerably decreases the computational cost of matrix-vector products involving a linear combination of the Loewner and shifted Loewner matrices needed for the partial SVD computation, while avoiding to form them. All results involving HSS-matrices presented in this paper have been obtained by means of the `hm-toolbox` [28].

The employment of an HSS-representation of $\mathcal{C}$ may introduce some inexactness in our scheme and this has to be taken into account in the iterative SVD computation. The use of inexact matrix-vector products within iterative procedures has been the subject of numerous research papers: Krylov techniques for solving linear systems and matrix equations [6, 24, 32, 40, 43], eigenvalue problems [13, 39], or an inexact variant of the Lanczos bidiagonalization for the computation of leading singular triplets of a generic matrix function [14]. In our case, we do not need an accurate approximation of the singular triplets, but rather have meaningful spaces spanned by the computed left and right singular vectors so that the obtained reduced model inherits the desired approximation properties (see, e.g., [2, 21]).

The remainder of the paper is structured as follows. Section 2 provides a review of the Loewner framework, whereas Sect. 3 presents results showcasing the special structure of the Loewner and shifted Loewner matrices as Cauchy-like matrices and their approximation as hierarchically semiseparable matrices allowing for efficient, inexact matrix-vector products in the partial SVD computation. Section 4 presents the results of our numerical experiments and Sect. 5 concludes the paper.

2 Review of the Loewner Framework

The Loewner framework has been proposed to address the rational interpolation/approximation problem. In the control community, this is referred to as system identification from frequency domain measurements and is stated below.

Problem Statement (Rational approximation in the complex plane) Given points s_j in the complex plane (which can represent angular frequencies $i\omega_j$ if s_j are on the imaginary axis) and the corresponding transfer function measurement $\mathbf{H}_j \in \mathbb{C}^{p \times q}$ for a system with q inputs and p outputs:

$$(s_j; \mathbf{H}_j), \quad j = 1, \ldots, N, \tag{1}$$

with p and q assumed to be much smaller than N, the problem amounts to finding the rational transfer function $\mathbf{H}(s)$ which approximates the data:

$$\mathbf{H}_j \approx \mathbf{H}(s = s_j), \quad \forall j = 1, \ldots, N. \tag{2}$$

Thus, the transfer function evaluated for the Laplace variable $s = s_j$ should be close (in some norm) to the corresponding measurement $\mathbf{H}_j$. Several equivalent representations are possible for the rational transfer function, namely pole-residue, pole-zero, state-space or descriptor-form.

Most systems of interest are real, with their transfer function satisfying the complex conjugate condition $\mathbf{H}(\bar{s}) = \overline{\mathbf{H}(s)}$. Hence, we add complex conjugate measurements $(\bar{s}_j; \overline{\mathbf{H}}_j)$ to the set (1).

We proceed by presenting the Loewner framework as a solution scheme addressing the rational approximation problem. The first step in the Loewner framework [26, 30] is partitioning the data in two disjoint sets. This partition influences the conditioning of the problem [21][Ch. 2.1] and finding the optimal partition for each data set is beyond the scope of this paper. The most natural partitions are summarized in the following (assuming an even number of measurements N and sampling points sorted in ascending order with respect to their absolute value):

– HALF&HALF: the first half of the data in one set and the other half in the second set:

$$\left\{ s_1, \overline{s}_1, \ldots, s_{\frac{N}{2}}, \overline{s}_{\frac{N}{2}} \right\} \cup \left\{ s_{\frac{N}{2}+1}, \overline{s}_{\frac{N}{2}+1}, \ldots, s_N, \overline{s}_N \right\} \tag{3}$$

and, correspondingly,

$$\left\{ \mathbf{H}_1, \overline{\mathbf{H}}_1, \ldots, \mathbf{H}_{\frac{N}{2}}, \overline{\mathbf{H}}_{\frac{N}{2}} \right\} \cup \left\{ \mathbf{H}_{\frac{N}{2}+1}, \overline{\mathbf{H}}_{\frac{N}{2}+1}, \ldots, \mathbf{H}_N, \overline{\mathbf{H}}_N \right\}, \tag{4}$$

– ODD&EVEN: data with odd indices in the first set and data with even indices in the second set:

$$\{ s_1, \overline{s}_1, \ldots, s_{N-1}, \overline{s}_{N-1} \} \cup \{ s_2, \overline{s}_2, \ldots, s_N, \overline{s}_N \} \tag{5}$$

and, correspondingly,

$$\left\{ \mathbf{H}_1, \overline{\mathbf{H}}_1, \ldots, \mathbf{H}_{N-1}, \overline{\mathbf{H}}_{N-1} \right\} \cup \left\{ \mathbf{H}_2, \overline{\mathbf{H}}_2, \ldots, \mathbf{H}_N, \overline{\mathbf{H}}_N \right\}. \tag{6}$$

The first set on the right in (3) and (5) comprises the *right points*, denoted by λ_k, $k = 1, \ldots, N$, while the second set comprises the *left points* μ_h, $h = 1, \ldots, N$. This splitting into right and left points is related to the concept of tangential interpolation, which is explained in the following paragraph.

The following step in the Loewner framework is choosing tangential directions as vectors which transform matrix data $\mathbf{H}_j$ into vector data: *right tangential directions* are column vectors $\mathbf{r}_k \in \mathbb{C}^q$ such that $\mathbf{H}_k \mathbf{r}_k = \mathbf{w}_k$, whereas *left tangential directions* are row vectors $\ell_h \in \mathbb{C}^{1 \times p}$ such that $\ell_h \mathbf{H}_h = \mathbf{v}_h$. The column vectors $\mathbf{w}_k \in \mathbb{C}^p$ are referred to as *right vector data*, while the row vectors $\mathbf{v}_h \in \mathbb{C}^{1 \times q}$ are referred to as *left vector data*. For simplicity, tangential directions can be chosen as alternating columns/rows of the identity matrix [26], resulting in vector data being column and row vectors of the original matrix data $\mathbf{H}_j$ in (1).

Remark 1 For scalar data obtained from single-input single-output (SISO) systems ($p = q = 1$), tangential directions $\mathbf{r}_k$, ℓ_h are simply equal to 1.

Remark 2 If the loss of information due to utilizing a single tangential direction per measurement, instead of the whole matrix $\mathbf{H}_j$, does not allow one to obtain an accurate approximation, one can employ the original matrix $\mathbf{H}_j$. This is equivalent to considering several tangential directions for the same point. To obtain block right matrix data for $\mathbf{H}_j \in \mathbb{C}^{p \times q}$, the corresponding frequency should be repeated q times as a right point and all columns of the identity matrix of size $q \times q$ should be considered as right directions. Similarly, to obtain block left matrix data for $\mathbf{H}_j \in \mathbb{C}^{p \times q}$, the corresponding frequency should be repeated p times as a left point and all rows of the identity matrix of size $p \times p$ should be considered as left directions.

With this notation in place, the Loewner matrix is defined entry-wise as

$$\mathbb{L}_{hk} = \frac{\mathbf{v}_h \mathbf{r}_k - \ell_h \mathbf{w}_k}{\mu_h - \lambda_k}, \quad h, k = 1, \ldots, N, \tag{7}$$

and the shifted Loewner matrix is defined as

$$\mathbb{S}_{hk} = \frac{\mu_h \mathbf{v}_h \mathbf{r}_k - \lambda_k \ell_h \mathbf{w}_k}{\mu_h - \lambda_k}, \quad h, k = 1, \ldots, N. \tag{8}$$

Note that the numerators are scalar quantities as they are obtained by taking inner products.

The quantities defined previously are collected into the following matrices

$$\mathbf{\Lambda} = \mathrm{diag}\left(\left[\lambda_1, \ldots, \lambda_N\right]\right) \in \mathbb{C}^{N \times N}, \qquad \mathbf{R} = \left[\mathbf{r}_1, \ldots, \mathbf{r}_N\right] \in \mathbb{C}^{q \times N},$$

$$\mathbf{W} = \left[\mathbf{w}_1, \ldots, \mathbf{w}_N\right] \in \mathbb{C}^{p \times N},$$

$$M = \mathrm{diag}\left(\left[\mu_1, \ldots, \mu_N\right]\right) \in \mathbb{C}^{N \times N}, \qquad \mathbf{L} = \begin{bmatrix} \ell_1 \\ \vdots \\ \ell_N \end{bmatrix} \in \mathbb{C}^{N \times p},$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \end{bmatrix} \in \mathbb{C}^{N \times q}, \tag{9}$$

$$\mathbb{L} = \begin{bmatrix} \frac{\mathbf{v}_1 \mathbf{r}_1 - \ell_1 \mathbf{w}_1}{\mu_1 - \lambda_1} & \cdots & \frac{\mathbf{v}_1 \mathbf{r}_N - \ell_1 \mathbf{w}_N}{\mu_1 - \lambda_N} \\ \vdots & \ddots & \vdots \\ \frac{\mathbf{v}_N \mathbf{r}_1 - \ell_N \mathbf{w}_1}{\mu_N - \lambda_1} & \cdots & \frac{\mathbf{v}_N \mathbf{r}_N - \ell_N \mathbf{w}_N}{\mu_N - \lambda_N} \end{bmatrix} \in \mathbb{C}^{N \times N},$$

$$\mathbb{S} = \begin{bmatrix} \frac{\mathbf{v}_1 \mathbf{r}_1 - \ell_1 \mathbf{w}_1}{\mu_1 - \lambda_1} & \cdots & \frac{\mathbf{v}_1 \mathbf{r}_N - \ell_1 \mathbf{w}_N}{\mu_1 - \lambda_N} \\ \vdots & \ddots & \vdots \\ \frac{\mathbf{v}_N \mathbf{r}_1 - \ell_N \mathbf{w}_1}{\mu_N - \lambda_1} & \cdots & \frac{\mathbf{v}_N \mathbf{r}_N - \ell_N \mathbf{w}_N}{\mu_N - \lambda_N} \end{bmatrix} \in \mathbb{C}^{N \times N}. \tag{10}$$

By construction, the Loewner and shifted Loewner matrices satisfy the following Sylvester equations:

$$M\mathbb{L} - \mathbb{L}\mathbf{\Lambda} = \mathbf{VR} - \mathbf{LW}, \qquad M\mathbb{S} - \mathbb{S}\mathbf{\Lambda} = M\mathbf{VR} - \mathbf{LW}\mathbf{\Lambda}, \tag{11}$$

as well as the following relations:

$$\mathbb{S} - \mathbb{L}\mathbf{\Lambda} = \mathbf{VR}, \qquad \mathbb{S} - M\mathbb{L} = \mathbf{LW}, \tag{12}$$

which will prove useful in our proposed matrix-free matrix-vector product approach.

After introducing notation, we are ready to state the solution provided by the Loewner framework to the rational approximation problem. A (non minimal) model for the transfer function in descriptor-form $\mathbf{H}(s) = \mathbf{C}(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$ is given by

$$\mathbf{H}(s) = \mathbf{W}(\mathbb{S} - s\mathbb{L})^{-1}\mathbf{V}. \tag{13}$$

Since we have recast the original problem as a tangential interpolation problem, this transfer function satisfies the right and left interpolation conditions [30] $\mathbf{H}(\lambda_k)\mathbf{r}_k = \mathbf{w}_k$ and $\ell_h \mathbf{H}(\mu_h) = \mathbf{v}_h$, $h, k = 1, \ldots, N$ exactly. To obtain a minimal model, we perform a singular value decomposition

$$[\mathbf{Y}, \mathbf{\Sigma}, \mathbf{X}] = \mathrm{svd}(\mathbb{S} - x\mathbb{L}), \quad x \in \{s_j\}, \tag{14}$$

where $\mathbf{\Sigma}$ is diagonal and $\mathbf{Y}, \mathbf{X}$ contain the left and right singular vectors, respectively. Choosing the order n of the truncated SVD (n is application-dependent), we define (in Matlab

notation) $\mathbf{X}_n = \mathbf{X}(:, 1{:}n)$ and $\mathbf{Y}_n = \mathbf{Y}(:, 1{:}n)^*$. Finally, the model of size n in descriptor form is

$$\mathbf{E} = -\mathbf{Y}_n\mathbb{L}\mathbf{X}_n = -\mathbb{L}_n, \ \mathbf{A} = -\mathbf{Y}_n\mathbb{S}\mathbf{X}_n = -\mathbb{S}_n, \ \mathbf{B} = \mathbf{Y}_n\mathbf{V} = \mathbf{V}_n, \ \mathbf{C} = \mathbf{W}\mathbf{X}_n = \mathbf{W}_n,$$
$$\mathbf{D} = \mathbf{0}. \tag{15}$$

In the following section, we exploit the Cauchy-like structure of the Loewner and shifted Loewner matrices to design efficient approaches, both in terms of memory storage and CPU time, to compute the SVD in (14) by making use of hierarchical matrices.

3 Exploiting the Structure of $\mathbb{L}$ and $\mathbb{S}$

For data sets with a sizable number N of measurements $\mathbf{H}_j$, the construction of the large, dense Loewner and shifted Loewner matrices is demanding, both in terms of computational efforts as well as storage requirements. The computation of each entry of $\mathbb{L}$ and $\mathbb{S}$ using (7) and (8) yields a total cost of $\mathcal{O}(N^2 (p + q))$ floating point operations (FLOPs) for assembling the entire $\mathbb{L}$ and $\mathbb{S}$ matrices. The number of nonzero entries in $\mathbb{L}$ and $\mathbb{S}$ is $\mathcal{O}(N^2)$, much larger than the memory requirements for storing the data in $\mathbf{\Lambda}$, M, $\mathbf{R}$, $\mathbf{W}$, $\mathbf{L}$, and $\mathbf{V}$[1]. Besides these excessive storage requirements, there are also considerations to be made regarding the CPU time required for the SVD computation of the matrix $\mathbb{S} - x\mathbb{L}$, $x \in \{f_i\}$ in (14). Especially for large dimensional problems, for which we expect a fast decay, it is preferred to compute only the first n singular triplets, thus avoiding wasting resources in computing the full SVD. To this end, many iterative methods have been developed for computing partial SVDs; see, e.g., [3, 19, 25, 41] to name a few. The bottleneck in these approaches is the matrix-vector product with the coefficient matrix, namely $\mathbb{S} - x\mathbb{L}$ in our case. This operation costs $\mathcal{O}(N^2)$ FLOPs due to the dense pattern of $\mathbb{S} - x\mathbb{L}$.

This section tackles the cost reduction of performing a matrix-vector product with $\mathbb{S} - x\mathbb{L}$ while avoiding the explicit allocation of $\mathbb{L}$ and $\mathbb{S}$. The proposed strategy is supported by a thorough analysis of the computational cost, showing that, for very large data sets for which carrying out the full SVD is intractable, our strategy leads to remarkable reductions in both the computational efforts and the storage demand for building minimal realizations in the Loewner framework.

3.1 Hadamard Product and Cauchy Matrices

We present novel results which exploit the particular structure of the Loewner and shifted Loewner matrices. These developments involve the Sylvester equations (11) with diagonal coefficient matrices $\mathbf{\Lambda}$ and $\mathbf{M}$.

Theorem 1 *The Loewner and shifted Loewner matrices $\mathbb{L}$ and $\mathbb{S}$ satisfying the Sylvester equations in (11) are such that*

$$\mathbb{L} = \sum_{j=1}^{q} diag(\widetilde{\mathbf{v}}_j)\mathcal{C}diag(\widetilde{\mathbf{r}}_j^*) - \sum_{j=1}^{p} diag(\widetilde{\ell}_j)\mathcal{C}diag(\widetilde{\mathbf{w}}_j^*), \tag{16}$$

---

[1] The number of nonzero entries in the data matrices $\mathbf{V}$ and $\mathbf{R}$ amounts to $\mathcal{O}(qN)$ and to $\mathcal{O}(pN)$ for $\mathbf{W}$ and $\mathbf{L}$.

and

$$\mathbb{S} = \sum_{j=1}^{q} diag(M\widetilde{\mathbf{v}}_j)\mathcal{C}diag(\widetilde{\mathbf{r}}_j^*) - \sum_{j=1}^{p} diag(\widetilde{\ell}_j)\mathcal{C}diag(\Lambda^*\widetilde{\mathbf{w}}_j^*), \tag{17}$$

where $\mathcal{C}$ denotes the following Cauchy matrix

$$\mathcal{C} = \begin{bmatrix} \frac{1}{\mu_1-\lambda_1} & \cdots & \frac{1}{\mu_1-\lambda_N} \\ \vdots & \ddots & \vdots \\ \frac{1}{\mu_N-\lambda_1} & \cdots & \frac{1}{\mu_N-\lambda_N} \end{bmatrix},$$

while the vectors $\widetilde{\mathbf{v}}_j \in \mathbb{C}^N$ and $\widetilde{\ell}_j \in \mathbb{C}^N$ denote the j-th columns of $\mathbf{V}$ and $\mathbf{L}$, respectively, so that

$$V = [\widetilde{\mathbf{v}}_1, \ldots, \widetilde{\mathbf{v}}_q], \quad L = [\widetilde{\ell}_1, \ldots, \widetilde{\ell}_q].$$

Similarly, the vectors $\widetilde{\mathbf{r}}_j \in \mathbb{C}^{1\times N}$ and $\widetilde{\mathbf{w}}_j \in \mathbb{C}^{1\times N}$ are the j-th rows of $\mathbf{R}$ and $\mathbf{W}$, respectively, namely

$$R = \begin{bmatrix} \widetilde{\mathbf{r}}_1 \\ \vdots \\ \widetilde{\mathbf{r}}_q \end{bmatrix}, \quad W = \begin{bmatrix} \widetilde{\mathbf{w}}_1 \\ \vdots \\ \widetilde{\mathbf{w}}_q \end{bmatrix}.$$

Proof The Loewner and shifted Loewner matrices $\mathbb{L}$ and $\mathbb{S}$ are Cauchy-like matrices as they are obtained by taking the Hadamard product $\circ$ between the Cauchy matrix $\mathcal{C}$ and the right-hand sides of the Sylvester equations in (11). In particular,

$$\mathbb{L} = \mathcal{C} \circ (\mathbf{VR} - \mathbf{LW}), \qquad \mathbb{S} = \mathcal{C} \circ (M\mathbf{VR} - \mathbf{LW}\Lambda). \tag{18}$$

An important property of the Hadamard product reads as follows. For any vectors $\mathbf{x}, \mathbf{y} \in \mathbb{C}^N$, it holds

$$\mathcal{C} \circ (\mathbf{xy}^*) = diag(\mathbf{x})\mathcal{C}diag(\mathbf{y}^*).$$

This, along with the low-rank structure of $\mathbf{VR} - \mathbf{LW}$ and $M\mathbf{VR} - \mathbf{LW}\Lambda$, yields the results in (16) and (17).   □

Corollary 1 *Given a vector $\mathbf{y} \in \mathbb{C}^N$ and $x \in \{s_j\}$, we have*

$$(\mathbb{S} - x\mathbb{L})\mathbf{y} = \sum_{j=1}^{q} \widetilde{\mathbf{v}}_j \circ (\mathcal{C}(\widetilde{\mathbf{r}}_j^* \circ \widetilde{\mathbf{y}})) - \sum_{j=1}^{p} \widetilde{\ell}_j \circ (\mathcal{C}(\widetilde{\mathbf{w}}_j^* \circ \widetilde{\mathbf{y}})) + \mathbf{VRy},$$

where $\widetilde{\mathbf{y}} = (\Lambda - x\mathbf{I})\mathbf{y}$, with $\mathbf{I}$, the identity matrix.

Proof Thanks to (12), we can write

$$(\mathbb{S} - x\mathbb{L})\mathbf{y} = (\mathbb{L}\Lambda + \mathbf{VR} - x\mathbb{L})\mathbf{y} = \mathbb{L}(\Lambda - x\mathbf{I})\mathbf{y} + \mathbf{VRy}.$$

The result follows by substituting the expression of $\mathbb{L}$ given in Theorem 1 in the equation above.   □

Corollary 1 shows that the majority of the computational cost of performing the matrix-vector multiplication $(\mathbb{S} - x\mathbb{L})\mathbf{y}$ amounts to computing $p + q$ matrix-vector products with the Cauchy matrix $\mathcal{C}$.

Extensive research has been devoted to fully exploiting the rich structure of Cauchy matrices. Several algorithms for computing the matrix-vector product $\mathcal{C}\mathbf{y}$ can be found in the literature and many avoid assembling the full matrix $\mathcal{C}$ (see, e.g., [7, 15, 17, 33]). In the next section we recall the strategy presented by Pan in [33, 34] to represent $\mathcal{C}$ in terms of a hierarchically semiseparable (HSS) matrix. Even though the novel scheme proposed in this paper does not depend on the strategy employed for performing the matrix-vector product $\mathcal{C}\mathbf{y}$—as long as it is efficient—we believe that the HSS framework may be advantageous as, in principle, many matrix-vector products with $\mathcal{C}$ are needed for computing a (partial) SVD of the matrix $\mathbb{S} - x\mathbb{L}$.

We conclude this section with the following remarks.

Remark 3 The number n of singular triplets needed to be computed to achieve the minimal realization $(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$ in (15) is difficult to estimate a-priori[2]. However, the expression of $\mathbb{L}$ and $\mathbb{S}$ in terms of the Hadamard product can be useful to this end. Indeed, another important property of the Hadamard product is that, for any matrices $\mathbf{A}$ and $\mathbf{B}$, $\mathrm{rank}(\mathbf{A} \circ \mathbf{B}) \leq \mathrm{rank}(\mathbf{A})\mathrm{rank}(\mathbf{B})$. Therefore,

$$\mathrm{rank}(\mathbb{L}) \leq \mathrm{rank}(\mathcal{C}) \cdot \mathrm{rank}(\mathbf{VR} - \mathbf{LW}) \leq (p + q) \cdot \mathrm{rank}(\mathcal{C}),$$

and similarly for $\mathbb{S}$. Thus, we have

$$\mathrm{rank}(\mathbb{S} - x\mathbb{L}) \leq 2(p + q) \cdot \mathrm{rank}(\mathcal{C}), \quad \forall\, x \in \{s_j\}. \tag{19}$$

In general, the Cauchy matrix $\mathcal{C}$ is full rank so this inequality is trivially satisfied. However, depending on the partitioning of the points into λ_k and μ_h (as in (3) and (5)), it can be *numerically* low-rank (see, e.g., [33][Theorem 5], [5, 9]). If $\pi_{\mathcal{C}}$ denotes the numerical rank of $\mathcal{C}$, then $2(p + q)\pi_{\mathcal{C}}$ is a rough estimate for the numerical rank of $\mathbb{S} - x\mathbb{L}$. Oftentimes, the underlying dynamical system is of much lower complexity, thus allowing for the computation of a minimal realization of reduced order n. One can also use insight of the system itself or count the number of peaks in the frequency response to estimate n (for systems with poles having dominant imaginary parts).

Remark 4 The expression of $\mathbb{L}$ and $\mathbb{S}$ in terms of the Hadamard product provides us with an upper bound of the spectral norm of the Loewner and shifted Loewner matrix. Indeed, the spectral norm is submultiplicative with respect to the Hadamard product [20][Theorem 5.5.1], hence

$$\|\mathbb{L}\| = \|\mathcal{C} \circ (\mathbf{VR} - \mathbf{LW})\| \leq \|\mathcal{C}\| \cdot \|\mathbf{VR} - \mathbf{LW}\|$$
$$\leq \|\mathcal{C}\|_F \cdot \|\mathbf{VR} - \mathbf{LW}\| = \|\mathbf{VR} - \mathbf{LW}\|\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\frac{1}{\mu_i - \lambda_j}\right|^2},$$

where $\|\mathcal{C}\|_F$ denotes the Frobenius norm of $\mathcal{C}$. Note that $\|\mathbf{VR} - \mathbf{LW}\|$ can be computed cheaply, e.g., by a power method exploiting the low rank of $\mathbf{VR} - \mathbf{LW}$.

Similarly,

$$\|\mathbb{S}\| \leq \|\mathbf{MVR} - \mathbf{LW\Lambda}\|\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\frac{1}{\mu_i - \lambda_j}\right|^2}.$$

---

[2] In [5][Section 4.3], some results on the numerical rank of $\mathbb{L}$ are presented provided $[\min \lambda_k, \max \lambda_k] \cap [\min \mu_h, \max \mu_h] = \emptyset$.

Remark 5 Low-rank approximations to $\mathbb{L}$ and $\mathbb{S}$ may be computed by adaptive cross approximation [4], particularly suited for hierarchical matrices, the CUR decomposition [11] as in [22, 38], or related schemes. These approaches select a certain number of columns and rows of the original matrices in a greedy fashion based on various heuristics, and a *core* matrix is utilised to compute a low-rank approximation. If a given threshold on the desired accuracy of the computed approximation is provided as an input, these algorithms often construct matrices whose rank is much larger than the one of the target matrices $\mathbb{L}$ and $\mathbb{S}$. On the other hand, by fixing the rank k of the approximation, $k \approx \mathtt{rank}(\mathbb{L}), \mathtt{rank}(\mathbb{S})$ - assuming we know an estimate of $\mathtt{rank}(\mathbb{L}), \mathtt{rank}(\mathbb{S})$ - the accuracy we achieve may be very low affecting the reliability of the computed reduced models.

3.2 Hierarchically Semiseparable (HSS) Representation of a Cauchy Matrix

The literature on HSS matrices is rather vast and technical (see, e.g., [8, 33, 34, 44, 47] and references therein). Here we recall only the main properties of this class of matrices and their role in the efficient representation of Cauchy matrices. Such a technique is also closely related to the Fast Multipole Method (FMM). We refer the interested reader to, e.g., [8, 9] for more details on the interconnection between HSS matrices and FMM.

Definition 1 [34, Definition 27] Let $\mathbf{A}$ be an $N \times N$ matrix with α being the maximum rank of all its subdiagonal blocks, namely the blocks of all sizes lying strictly below the block diagonal, and β the maximum rank of all its superdiagonal blocks, namely the blocks of all sizes lying strictly above the block diagonal, respectively. Then, $\mathbf{A}$ is (α, β)-HSS if its diagonal blocks consist of $\mathcal{O}((\alpha + \beta)N)$ entries.

The (α, β)-HSS representation of a matrix $\mathbf{A}$ is very advantageous whenever α and β are small. For instance, it allows us to express $\mathbf{A}$ in terms of $\mathcal{O}((\alpha + \beta)N)$ parameters avoiding storing its N^2 entries. Moreover, a whole, efficient HSS arithmetic has been developed in the last decades (see, e.g., [8, 47]). For instance, the computational cost of the matrix-vector product $\mathbf{A}\mathbf{y}$ amounts to $\mathcal{O}((\alpha + \beta)N)$ FLOPs. If $\mathbf{A}$ is nonsingular, its inverse is also a (α, β)-HSS matrix that can be computed in $\mathcal{O}((\alpha + \beta)^3 N)$ FLOPs (see, e.g., [34, Section 6]).

To fully exploit the HSS framework for our purposes, we wish to represent the Cauchy matrix $\mathcal{C}$ in terms of a HSS matrix with a low off-diagonal rank. In light of Corollary 1, this would considerably decrease the computational cost of the matrix-vector products involving $\mathbb{S} - x\mathbb{L}$ while avoiding forming the dense matrices $\mathbb{S}$ and $\mathbb{L}$.

The construction of an HSS approximation $\widetilde{\mathcal{C}}$ to $\mathcal{C}$ is rather involved and the magnitude of the (α, β)-rank of the computed $\widetilde{\mathcal{C}}$ strictly depends on the partitioning of the frequencies along with the accuracy that has been selected for the actual computation of $\widetilde{\mathcal{C}}^3$. Given two parameters $c \in \mathbb{C}$ and $1 \le r \le N$, the cardinal relation underlying this construction is the following

$$
\begin{aligned}
\frac{1}{\mu_i - \lambda_j} &= \frac{1}{\mu_i - c} \cdot \frac{1}{1 - \frac{\lambda_j - c}{\mu_i - c}} \\
&= \sum_{\ell=0}^{r-1} \frac{1}{(\mu_i - c)^{\ell+1}} (\lambda_j - c)^\ell + \underbrace{\frac{1}{\mu_i - c} \left(\frac{\lambda_j - c}{\mu_i - c} \right)^r \frac{1}{1 - \frac{\lambda_j - c}{\mu_i - c}}}_{=: \mathcal{E}},
\end{aligned}
$$

---

[3] Roughly speaking, such a threshold is related to the computation of the low-rank approximations to the off-diagonal blocks of $\mathcal{C}$ (see, e.g., [46, Corollary 4.3], [23, Theorem 4.7]).

and the approximation obtained by neglecting the error term $\mathcal{E}$ can be respresented in low-rank format thanks to the separability (in μ_i and λ_j) of the first term in the last step of the relation above. See, e.g., [34, Section 8] for further details on the computation of an HSS-representation of a Cauchy matrix. In this paper we employ the readily available `hm-toolbox` [28].

Example 1 We investigate the impact of the most commonly-used frequency partitions (HALF&HALF and ODD&EVEN on the HSS-rank of the computed $\widetilde{\mathcal{C}}$ for a mechanical structure. We emphasize that the most effective partition is problem-dependent and is still an open problem, beyond the scope of this paper. However, in [12], [21, Ch. 2.1] the authors suggest the use of partitions with interleaving frequencies like ODD&EVEN in order to avoid the introduction of an "artifical" ill-conditioning.

We consider the *Flexible Aircraft* data set [36] from the MORWIKI [42]. This dataset contains 421 frequency values ω_j expressed in rad/s and the corresponding measurements of the transfer function $\mathbf{H}_j$. We disregard the last data point and consider the remaining frequencies ranging from $f_1 = 0.1$Hz to $f_{420} = 42$Hz. As this is a mechanical structure, frequencies considered are in the low spectrum, as opposed to electrical systems, for which frequencies typically span the GHz range.

To avoid complex arithmetic, it is preferred and more advantageous to perform a change of basis when dealing with sampling points on the imaginary axis: $s_j = \mathrm{i}\omega_j$. By defining

$$\boldsymbol{\Pi} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -\mathrm{i} \\ 1 & \mathrm{i} \end{bmatrix} \text{ and } \mathbf{P} = \mathrm{blkdiag}\left(\begin{bmatrix} \boldsymbol{\Pi}, & \ldots, & \boldsymbol{\Pi} \end{bmatrix}\right) \in \mathbb{C}^{N \times N}, \tag{20}$$

we obtain matrices with real entries:

$$\boldsymbol{\Lambda}_r := \mathbf{P}^*\boldsymbol{\Lambda}\mathbf{P}, \quad \boldsymbol{M}_r := \mathbf{P}^*\boldsymbol{M}\mathbf{P}, \quad \mathbf{L}_r := \mathbf{P}^*\mathbf{L}, \quad \mathbf{V}_r := \mathbf{P}^*\mathbf{V},$$

$$\mathbf{R}_r := \mathbf{R}\mathbf{P}, \quad \mathbf{W}_r := \mathbf{W}\mathbf{P}, \quad \mathbb{L}_r := \mathbf{P}^*\mathbb{L}\mathbf{P}, \quad \mathbb{S}_r := \mathbf{P}^*\mathbb{S}\mathbf{P},$$

where $\mathbf{P}^*$ stands for the complex conjugate transpose of the matrix $\mathbf{P}$ and $\mathbf{P}^{-1} = \mathbf{P}^*$. These quantities satisfy analogous expressions as in (11) and (12). Unfortunately, $\boldsymbol{\Lambda}_r$ and $\boldsymbol{M}_r$ are no longer diagonal and this represents a major drawback in taking advantage of the Sylvester equations (11) for a fast computation of $\mathbb{L}_r$ and $\mathbb{S}_r$. However, $\boldsymbol{\Lambda}_r^2$ and $\boldsymbol{M}_r^2$ are diagonal and given by

$$\boldsymbol{\Lambda}_r^2 = \mathrm{blkdiag}\begin{bmatrix} -\omega_k^2 & 0 \\ 0 & -\omega_k^2 \end{bmatrix}, \quad k = 1, \ldots, N,$$

$$\boldsymbol{M}_r^2 = \mathrm{blkdiag}\begin{bmatrix} -\omega_h^2 & 0 \\ 0 & -\omega_h^2 \end{bmatrix}, \quad h = 1, \ldots, N. \tag{21}$$

By multiplying the first equation in (11) by $\boldsymbol{M}_r$ on the left and, afterwards, multiplying it by $\boldsymbol{\Lambda}_r$ on the right and adding the results together, a new Sylvester equation with diagonal coefficient matrices is obtained:

$$\boldsymbol{M}_r^2 \mathbb{L}_r - \mathbb{L}_r \boldsymbol{\Lambda}_r^2 = \boldsymbol{M}_r \left(\mathbf{V}_r \mathbf{R}_r - \mathbf{L}_r \mathbf{W}_r\right) + \left(\mathbf{V}_r \mathbf{R}_r - \mathbf{L}_r \mathbf{W}_r\right) \boldsymbol{\Lambda}_r. \tag{22}$$

By performing the same operations on the second equation in (11), a similar Sylvester equation is obtained for the shifted Loewner matrix:

$$\boldsymbol{M}_r^2 \mathbb{S}_r - \mathbb{S}_r \boldsymbol{\Lambda}_r^2 = \boldsymbol{M}_r \left(\boldsymbol{M}_r \mathbf{V}_r \mathbf{R}_r - \mathbf{L}_r \mathbf{W}_r \boldsymbol{\Lambda}_r\right) + \left(\boldsymbol{M}_r \mathbf{V}_r \mathbf{R}_r - \mathbf{L}_r \mathbf{W}_r \boldsymbol{\Lambda}_r\right) \boldsymbol{\Lambda}_r. \tag{23}$$

Table 1 Example 1. HSS-rank and relative error of the HSS representation $\widetilde{\mathcal{C}}$ of $\mathcal{C}$ for different frequency partitions along with the rank of $\mathcal{C}$

	HALF&HALF	ODD&EVEN	ODD&EVEN (REAL)
`hssrank`$(\widetilde{\mathcal{C}})$	32	30	13
`rank`$(\mathcal{C})$	36	420	210
$\|\widetilde{\mathcal{C}} - \mathcal{C}\|/\|\mathcal{C}\|$	2.68e–12	2.61e–11	6.62e–13

In the following, we refer to this as the ODD&EVEN (REAL) partition[4]

We recall the three different partitions of the sampling points $\{s_j = i\omega_j\}_{j=1}^{j=420}$:

- HALF&HALF: $\mathbf{\Lambda} = \text{diag}([i\omega_1, -i\omega_1, \ldots, i\omega_{210}, -i\omega_{210}])$
  $\mathbf{M} = \text{diag}([i\omega_{211}, -i\omega_{211}, \ldots, i\omega_{420}, -i\omega_{420}])$.
- ODD&EVEN: $\mathbf{\Lambda} = \text{diag}([i\omega_1, -i\omega_1, \ldots, i\omega_{419}, -i\omega_{419}])$
  $\mathbf{M} = \text{diag}([i\omega_2, -i\omega_2, \ldots, i\omega_{420}, -i\omega_{420}])$.
- ODD&EVEN (REAL): $\mathbf{\Lambda}_r = \text{diag}([-\omega_1^2, -\omega_1^2, \ldots, -\omega_{419}^2, -\omega_{419}^2])$
  $\mathbf{M}_r = \text{diag}([-\omega_2^2, -\omega_2^2, \ldots, -\omega_{420}^2, -\omega_{420}^2])$.

For each partition, we compute the corresponding Cauchy matrix in HSS format $\widetilde{\mathcal{C}}$ without assembling the full $\mathcal{C}$ beforehand, by means of the function `hss` of the `hm-toolbox`:

$$\widetilde{\mathcal{C}} = \text{hss}('\text{cauchy}', \text{dM}, -\text{dL}, \text{N}, \text{N})$$

where `dM` and `dL` are N dimensional vectors containing the frequencies μ_h and λ_k, respectively. We then calculate its rank by `hssrank`$(\widetilde{\mathcal{C}})$[5]. In Table 1 we report the HSS-rank of the matrix $\widetilde{\mathcal{C}}$ for the partitions mentioned above. Thanks to the small dimension of the dataset, we are able to compute the full Cauchy matrix $\mathcal{C}$ and document its (standard) rank along with the relative error $\|\widetilde{\mathcal{C}} - \mathcal{C}\|/\|\mathcal{C}\|$. As expected, having two disjoint sets of frequencies like in the HALF&HALF partition leads to a Cauchy matrix $\mathcal{C}$ whose (standard) rank is low. This does not happen in the other two scenarios we examine so that taking advantage of the HSS format is necessary to achieve memory-saving representations of $\mathcal{C}$. The results in Table 1 show that a good accuracy in terms of the relative error can be achieved for all three frequency partitions. Nevertheless, the HSS rank of $\widetilde{\mathcal{C}}$ is significantly lower for the ODD&EVEN (REAL) partition, most likely due to the squaring of the frequencies performed in ODD&EVEN (REAL), which leads to a fast decay in the magnitude of the off-diagonal entries of $\mathcal{C}$. Hence, for a fixed threshold, the off-diagonal blocks of the Cauchy matrix associated to the ODD&EVEN (REAL) partition can be approximated by matrices having a smaller rank than those associated to the other two scenarios we examined.

Figure 1 we display the absolute value—on a logarithmic scale—of the entries of the Cauchy matrix $\mathcal{C}$ stemming from the different partitions. The same scale has been used in all the three figures, enforcing the observation that the ODD & EVEN (REAL) partition exhibits the fastest decay in the magnitude of the off-diagonal entries of $\mathcal{C}$.

---

[4] Notice that all the results stated in the previous sections still hold also for the ODD&EVEN (REAL) partition with straightforward modifications due to the different expression of the right-hand side in (22)–(23).

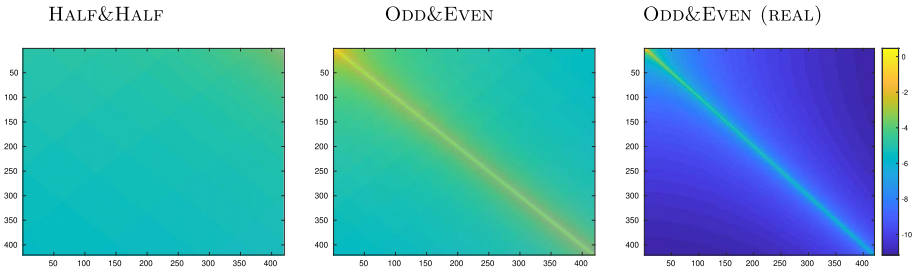[5] Following Definition 1, this function returns $\max\{\alpha, \beta\}$.

HALF&HALF   ODD&EVEN   ODD&EVEN (REAL)

Fig. 1 Example 1. Absolute value—on a logarithmic scale—of the entries of the Cauchy matrix $\mathcal{C}$ stemming from the three different frequency partitions we have examined

3.3 Efficient, Inexact Matrix-Vector Products

Whenever the matrix $\mathcal{C}$ admits an accurate approximation in terms of a low-rank HSS matrix $\widetilde{\mathcal{C}}$, the computational cost of performing the matrix-vector product $(\mathbb{S}-x\mathbb{L})\mathbf{y}$ can be significantly reduced.

Proposition 1 *Let $\widetilde{\mathcal{C}}$ be an (α, β)-HSS matrix that approximates the Cauchy matrix $\mathcal{C}$ accurately. If $\mathbb{L}$ and $\mathbb{S}$ satisfy the Sylvester equations in (11), then*

$$(\mathbb{S} - x\mathbb{L})\mathbf{y} = \sum_{j=1}^{q} \widetilde{\mathbf{v}}_j \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{r}}_j^* \circ \widetilde{\mathbf{y}})) - \sum_{j=1}^{p} \widetilde{\ell}_j \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{w}}_j^* \circ \widetilde{\mathbf{y}})) + \mathbf{VRy} + \mathbf{E}\widetilde{\mathbf{y}}, \qquad (24)$$

where $\|\boldsymbol{E}\| \leq (p+q)\|\mathcal{C} - \widetilde{\mathcal{C}}\| \max^2\{\|\mathbf{V}\|_\infty, \|\mathbf{R}\|_\infty, \|\mathbf{L}\|_\infty, \|\mathbf{W}\|_\infty\}$. Moreover, the computational cost of performing

$$\sum_{j=1}^{q} \widetilde{\mathbf{v}}_j \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{r}}_j^* \circ \widetilde{\mathbf{y}})) - \sum_{j=1}^{p} \widetilde{\ell}_j \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{w}}_j^* \circ \widetilde{\mathbf{y}})) + \mathbf{VRy}, \qquad (25)$$

amounts to $\mathcal{O}((p+q)(\alpha + \beta + 1)N)$ FLOPs.

Proof From the result in Corollary 1, we can write

$$(\mathbb{S} - x\mathbb{L})\mathbf{y} = \sum_{j=1}^{q} \widetilde{\mathbf{v}}_j \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{r}}_j^* \circ \widetilde{\mathbf{y}})) - \sum_{j=1}^{p} \widetilde{\ell}_j \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{w}}_j^* \circ \widetilde{\mathbf{y}})) + \mathbf{VRy}$$

$$+ \sum_{j=1}^{q} \widetilde{\mathbf{v}}_j \circ ((\mathcal{C} - \widetilde{\mathcal{C}})(\widetilde{\mathbf{r}}_j^* \circ \widetilde{\mathbf{y}})) - \sum_{j=1}^{p} \widetilde{\ell}_j \circ ((\mathcal{C} - \widetilde{\mathcal{C}})(\widetilde{\mathbf{w}}_j^* \circ \widetilde{\mathbf{y}}))$$

$$= \sum_{j=1}^{q} \widetilde{\mathbf{v}}_j \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{r}}_j^* \circ \widetilde{\mathbf{y}})) - \sum_{j=1}^{p} \widetilde{\ell}_j \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{w}}_j^* \circ \widetilde{\mathbf{y}})) + \mathbf{VRy} + \mathbf{E}\widetilde{\mathbf{y}},$$

where $\boldsymbol{E} := \sum_{j=1}^{q} \operatorname{diag}(\widetilde{\mathbf{v}}_j)(\mathcal{C} - \widetilde{\mathcal{C}})\operatorname{diag}(\widetilde{\mathbf{r}}_j^*) - \sum_{j=1}^{p} \operatorname{diag}(\widetilde{\ell}_j)(\mathcal{C} - \widetilde{\mathcal{C}})\operatorname{diag}(\widetilde{\mathbf{w}}_j^*)$. Therefore,

$$\|\boldsymbol{E}\| \leq \|\mathcal{C} - \widetilde{\mathcal{C}}\| \left(\sum_{j=1}^{q} \|\operatorname{diag}(\widetilde{\mathbf{v}}_j)\| \|\operatorname{diag}(\widetilde{\mathbf{r}}_j^*)\| + \sum_{j=1}^{p} \|\operatorname{diag}(\widetilde{\ell}_j)\| \|\operatorname{diag}(\widetilde{\mathbf{w}}_j^*)\| \right)$$

$$= \|\mathcal{C} - \widetilde{\mathcal{C}}\| \left(\sum_{j=1}^{q} \|\widetilde{\mathbf{v}}_j\|_\infty \|\widetilde{\mathbf{r}}_j\|_\infty + \sum_{j=1}^{p} \|\widetilde{\ell}_j\|_\infty \|\widetilde{\mathbf{w}}_j\|_\infty \right)$$

$$\leq (p+q)\|\mathcal{C} - \widetilde{\mathcal{C}}\| \max{}^2\{\|\mathbf{V}\|_\infty, \|\mathbf{R}\|_\infty, \|\mathbf{L}\|_\infty, \|\mathbf{W}\|_\infty\}.$$

This proves the first part of Proposition 1. To conclude, by making use of the property that the matrix-vector product with a (α, β)-HSS matrix costs $\mathcal{O}((\alpha + \beta)N)$ FLOPs and that **VR** has rank q, a direct computation shows that the number of operations needed to perform (25) amounts to $\mathcal{O}((p+q)(\alpha+\beta+1)N)$ FLOPs, which proves the second claim in Proposition 1. $\square$

As before, analogous results can be obtained for $\mathbb{L}_r$ and $\mathbb{S}_r$ satisfying (22) and (23), respectively.

Proposition 1 shows that, whenever $\|\mathcal{C} - \widetilde{\mathcal{C}}\|$ is small, the matrix-vector product $(\mathbb{S} - x\mathbb{L})\mathbf{y}$ can be well-approximated by the expression in (25) while dramatically reducing the computational complexity from $\mathcal{O}(N^2)$ FLOPs to $\mathcal{O}((p+q)(\alpha+\beta+1)N)$ FLOPs. However, when this approximation is used within our favorite iterative procedure for computing a partial SVD of $\mathbb{S} - x\mathbb{L}$, the inexactness introduced by neglecting the term $\boldsymbol{E}\widetilde{\mathbf{y}}$ should be taken into account.

The use of inexact matrix-vector products within certain iterative procedures has been the subject of numerous research papers: Krylov techniques for solving linear systems and matrix equations [6, 24, 32, 40, 43], eigenvalue problems [13, 39], or an inexact variant of the Lanczos bidiagonalization for the computation of some leading singular triplets of a generic matrix function $f(\mathbf{A})$ can be found in [14]. With the goal to decrease the computational cost of the overall procedure, these studies show that the accuracy of the matrix-vector product can be *relaxed* (becoming more and more inaccurate) as iterations proceed. In our framework, the inexactness introduced by approximating $(\mathbb{S} - x\mathbb{L})\mathbf{y}$ with (25) is fixed throughout the entire iterative procedure and mainly depends on $\|\mathcal{C} - \widetilde{\mathcal{C}}\|$, which is often small, as shown in Example 1. Therefore, the approximation

$$(\mathbb{S} - x\mathbb{L})\mathbf{y} \approx \sum_{i=1}^{q} \widetilde{\mathbf{v}}_i \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{r}}_i^* \circ \widetilde{\mathbf{y}})) - \sum_{i=1}^{p} \widetilde{\ell}_i \circ (\widetilde{\mathcal{C}}(\widetilde{\mathbf{w}}_i^* \circ \widetilde{\mathbf{y}})) + \mathbf{VRy},$$

does not greatly affect the accuracy of the computed singular triplets (see Sect. 4). Moreover, in our case, we do not need an accurate approximation of the singular triplets of $\mathbb{S} - x\mathbb{L}$. The main goal is to have meaningful spaces spanned by the computed left and right singular vectors so that the obtained reduced model $(\mathbf{E}, \mathbf{A}, \mathbf{B}, \mathbf{C})$ inherits the desired approximation properties. Moreover, as shown in [21, Corollary 1.4], [2, Proposition 8.25], in the case of noise-free measurements of a low-order rational function, even general projectors, not necessarily obtained from the SVD, can be employed for identifying the underlying function.

Remark 6 In Remark 3 we suggested to use the value $2(p+q)\pi_C$, where π_C is the numerical rank of $\mathcal{C}$, to decide on the number n of singular triplets of $\mathbb{S} - x\mathbb{L}$ needed for the reduced

model. For interlaced partitions, as it is the case with ODD&EVEN and ODD&EVEN (REAL) (see Table 1), the numerical (standard) rank of the Cauchy matrix is large, in general. Hence, the value $n = 2(p + q) \max\{\alpha, \beta\}$ may instead be employed for the computation of a meaningful reduced model whenever $\mathcal{C}$ can be well-approximated by a (α, β)-HSS matrix $\widetilde{\mathcal{C}}$[6]. Moreover, the HSS-rank of $\widetilde{\mathcal{C}}$ is obtained as a byproduct of the construction of $\widetilde{\mathcal{C}}$.

Remark 7 If $\mathcal{C}$ admits an accurate approximation in terms of an (α, β)-HSS matrix $\widetilde{\mathcal{C}}$, the expression in Theorem 1 shows that $\mathbb{L}$ can also be well-approximated by a HSS matrix $\widetilde{\mathbb{L}}$ whose rank is at most $((p + q)\alpha, (p + q)\beta)$. Even though the computational cost of $\widetilde{\mathbb{L}}\mathbf{y}$ would still be $\mathcal{O}((p + q)(\alpha + \beta)N)$ FLOPs, using the HSS approximation $\widetilde{\mathbb{L}}$ of $\mathbb{L}$ may be very advantageous whenever linear systems with $\mathbb{L}$ need to be solved (see, e.g., the procedure presented in [12] for the pseudospectra computation of $\mathbb{S} - x\mathbb{L}$). Indeed, as mentioned in Sect. 3.2, the computation of the inverse $\widetilde{\mathbb{L}}^{-1}$ of $\widetilde{\mathbb{L}}$ costs $\mathcal{O}((p + q)^3(\alpha + \beta)^3 N)$ FLOPs. Once $\widetilde{\mathbb{L}}^{-1}$ is computed, we need only $\mathcal{O}((p + q)(\alpha + \beta)N)$ FLOPs to perform $\widetilde{\mathbb{L}}^{-1}\mathbf{y}$.

Remark 8 We would like to mention that we did not observe any numerical issue related to the matrix-vector product (25) and its stability during our vast numerical testing. Moreover, one may want to perform the matrix-vector product by $\widetilde{\mathcal{C}}$ in a parallel environment to achieve better computational performance. The `strumpack` package[7] may be employed to this end. See also, e.g., [37]. However, such a parallel approach has not been used in the numerical experiments presented in Sect. 4.

4 Numerical Results

In this section we present numerical experiments illustrating the potential of the proposed approach.

In Example 2, we compare our approach to standard procedures employed in the Loewner framework. Recall that the main steps in the standard approach involve forming the full Loewner and shifted Loewner matrices $\mathbb{L}$ and $\mathbb{S}$ and computing the SVD of $\mathbb{S} - x\mathbb{L}$. This SVD can be either computed in full, followed by keeping only the n dominant singular vectors, or only these n singular vectors can be obtained by means of an iterative procedure, where the matrix-vector product with $\mathbb{S} - x\mathbb{L}$ is needed[8]. In the following, we report the overall running time, considering the *construction* step (CONSTRUCTION), i.e., the computation of $\mathbb{L}$ and $\mathbb{S}$ in the standard approach and of $\widetilde{\mathcal{C}}$ in our approach, as well as the *reduction* step (REDUCTION), involving the SVD computation followed by projection to obtain the reduced matrices in (15). In terms of memory requirements, for our approach, this involves the allocation of $\widetilde{\mathcal{C}}$ in the HSS format, while for the standard approach, we report the storage required for $\mathbb{L}$ and $\mathbb{S}$.

In Table 2 we recall the computational cost of the construction and reduction steps of both the standard approach, based on either a full or a partial SVD, and the novel one presented in this paper along with their memory requirements.

Lastly, the accuracy of the reduced models is reported in terms of the normalized $\mathcal{H}_2$-error:

$$\mathcal{H}_2 - \text{error} = \sqrt{\frac{\sum_{j=1}^{N} \|\mathbf{H}_j - \mathbf{H}(i\omega_j)\|_F^2}{\sum_{j=1}^{N} \|\mathbf{H}_j\|_F^2}},$$

---

[6] As before, the value $4(p + q) \max\{\alpha, \beta\}$ should be preferred whenever $\mathbb{L}_r$ and $\mathbb{S}_r$ solve (22) and (23), respectively.

[7] https://portal.nersc.gov/project/sparse/strumpack/

[8] We employ the MATLABfunctions `svd` and `svds`, respectively.

Table 2 Computational cost of the construction (CONSTRUCTION) and reduction (REDUCTION) steps of the different approaches we test along with their storage demand (STORAGE). The computational cost of the construction of $\widetilde{\mathcal{C}}$ can be found, e.g., in [28, Table 1]

	CONSTRUCTION	REDUCTION	STORAGE
Full svd	$\mathcal{O}(N^2(p+q))$	$\mathcal{O}(N^3)$	$\mathcal{O}(N^2)$
svds w/ $\mathbb{S} - x\mathbb{L}$	$\mathcal{O}(N^2(p+q))$	$\mathcal{O}(nN^2)$	$\mathcal{O}(N^2)$
svds w/ $\widetilde{\mathcal{C}}$	$\mathcal{O}(\max\{\alpha,\beta\}N\log N)$	$\mathcal{O}(n(p+q)(\alpha+\beta+1)N)$	$\mathcal{O}((\alpha+\beta+p+q)N)$

where $\|\cdot\|_F$ denotes the Frobenius norm. Similar results in terms of accuracy are attained for the $\mathcal{H}_\infty$-error, however, we decided not to document them here, for the sake of brevity.

In Example 3, we compare our novel strategy to the one presented in [18], which makes use of the low-rank ADI-Galerkin method for computing the Loewner matrix as the solution to (11). Such a scheme computes low-rank approximations to the dense Loewner matrix to speed-up the SVD computation, however, the memory constraints originating from the allocation of $\mathbb{L}$ and $\mathbb{S}$ are still present.

Results were obtained by running MATLABR2020b [29] on a MacBook Pro with an Intel Core i9 processor running at 2.3GHz using 16GB of RAM. All computations involving HSS matrices employed the hm-toolbox [28] with the default settings and the threshold for off-diagonal truncation set to 10^{-14}.

Example 2 We consider a synthetic problem for which we can control the order of the original system (n), the number of inputs and outputs ($p = q$), as well as the number of measurements (N). The system dynamics is generated randomly, with poles in complex conjugate pairs. In particular:

– the real part of the poles is random with mean -10^4 and standard deviation $-2 \cdot 10^3$; the imaginary part is also random, with mean 10^4 and standard deviation 10^6.
– residues associated to each pole are rank-1 matrices, obtained as outer products between two random vectors, both having the real part with mean 0 and standard deviation 10, while the imaginary part has mean 0 and standard deviation 10^2.

Measurement points $\{s_j = i\omega_j\}_{j=1}^{j=N}$ are logarithmically distributed between 10^4 and 10^7 rad/sec. Last, but not least, random noise with a signal-to-noise ratio $SNR = 100$ was added to the transfer function evaluation $\mathbf{H}(i\omega_j)$ to obtain the measurement matrices $\mathbf{H}_j$. We adopt the ODD&EVEN (REAL) partition of the frequencies as it achieves satisfactory approximation results while eliminating complex arithmetic. Tangential directions are chosen as unit vectors (rows and columns of the identity matrix of size p).

We compare the proposed approach to the traditional Loewner framework, in which the Loewner and shifted Loewner matrices $\mathbb{L}$ and $\mathbb{S}$ are formed and the full SVD of $\mathbb{S} - x\mathbb{L}$ is computed, as well as the alternative approach in which, after building $\mathbb{L}$ and $\mathbb{S}$, a partial SVD of $\mathbb{S} - x\mathbb{L}$ using the Matlab svds function is computed for various instances of the data set described above for different values of N, p, and n. The command svds was employed with the left starting vector $\widetilde{\mathbf{v}}_1$ (same notation as in Theorem 1) instead of a random starting vector, which is the default setting.

Figure 2 presents the memory requirements for storing the Loewner and shifted Loewner matrices $\mathbb{L}$ and $\mathbb{S}$ (in red), as opposed to storing the HSS approximation $\widetilde{\mathcal{C}}$ in our approach (in blue), along with the storage needed to allocate the data in $\boldsymbol{\Lambda}_r$, $\boldsymbol{M}_r$, $\mathbf{R}_r$, $\mathbf{L}_r$, $\mathbf{V}_r$, $\mathbf{W}_r$, for increasing values in the number of inputs and outputs p (in black). We point out that for values

Fig. 2 Example 2. Memory
requirements in Megabytes to
store $\mathbb{L}$, $\mathbb{S}$, $\widetilde{\mathcal{C}}$, and the data
matrices (Λ_r, M_r, $\mathbf{V}_r$, $\mathbf{W}_r$, $\mathbf{L}_r$,
$\mathbf{R}_r$) for different values of N and
p
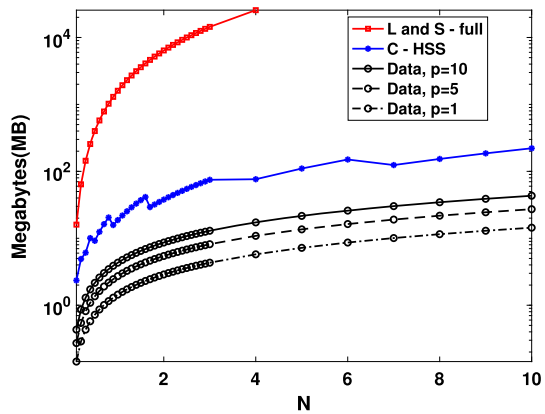
of N larger than 40 000, we were not able to allocate the full matrices $\mathbb{L}$ and $\mathbb{S}$ on the employed
laptop (this value, however, depends on the available RAM memory of the machine). For
instances when these matrices can be allocated, Fig. 2 shows that the memory requirements
for the proposed approach are always much lower than for the standard scheme. Moreover, in
contrast to what happens to the memory required for the data matrices, the storage demanded
by the allocation of $\widetilde{\mathcal{C}}$ in HSS format is independent of p.

We report the results of the comparison between the different approaches in terms of run
time in Table 3 for the number of measurements N varying between 1 000 to 100 000, the
number of inputs and outputs p taking values 1, 5 and 10, and the number of poles being 50
or 100. The "–" is used to indicate the instances for which we were not able to compute the
reduced model (15): for $N > 50\,000$, we cannot allocate the full matrices $\mathbb{L}$ and $\mathbb{S}$, and for
$N = 30\,000, 40\,000$ we could not compute the full SVD of $\mathbb{S} - x\mathbb{L}$. Such constraints are not
relevant to our proposed strategy. It is pertinent to remark the following:

1. the CPU time of the full SVD approach does not depend on p and n, only on N, as
   expected from Table 2: indeed, the cost of building $\mathbb{L}$ and $\mathbb{S}$ is quadratic in N whereas the
   full SVD demands $\mathcal{O}(N^3)$ FLOPs; the full SVD approach is rarely the fastest method (it
   can happen for very modest values of N in the considered range);
2. the CPU time of the full assembly of $\mathbb{S} - x\mathbb{L}$ followed by the `svds` Matlab command
   does not depend on p, only on N and n, as expected from Table 2: the construction of $\mathbb{L}$
   and $\mathbb{S}$ costs $\mathcal{O}(N^2)$ FLOPs, whereas the computational effort for the partial SVD depends
   on n, leading to a more demanding procedure for large n; it is usually the fastest approach
   for (very) modest values of N in the considered range and $p > 1$;
3. the HSS rank of the Cauchy matrix approximation $\widetilde{\mathcal{C}}$ only depends on the frequency
   samples, hence on N because, in our scenario, the sampling interval is the same, but the
   distribution of points inside the interval is different for each N; there may be instances
   when, for the same samples, the HSS rank of $\widetilde{\mathcal{C}}$ may produce slightly different results
   due to the randomness induced by the adaptive cross approximation procedure used in
   constructing $\widetilde{\mathcal{C}}$ (for instance, for $N = 50\,000$, $p = 5$, $n = 50$ and $n = 100$, the rank is
   28, while for the rest of the values considered for n and p, the rank is 27); moreover, the
   HSS rank increases with N;
4. our proposed approach is as accurate as the first two approaches, highlighting the fact
   that the HSS approximation $\widetilde{\mathcal{C}}$ does not lead to significant losses in the approximation

Table 3 Example 2. Computational time (in seconds) and $\mathcal{H}_2$-error achieved by each approach for different values of N (number of samples), p (number of inputs and outputs), and n (order of the underlying system and of the model) on the employed laptop

N	p	n	Full `svd`		`svds` w/ $\mathbb{S}-x\mathbb{L}$		`svds` w/ $\widetilde{\mathcal{C}}$		
			Time (s)	$\mathcal{H}_2$-error	Time (s)	$\mathcal{H}_2$-error	hssrank($\widetilde{\mathcal{C}}$)	Time (s)	$\mathcal{H}_2$-error
1 000	1	50	0.28	3.62e-10	**0.20**	3.62e-10	15	1.07	3.62e-10
3 000			10.69	3.71e-10	3.51	3.71e-10	19	**3.16**	3.71e-10
5 000			20.79	3.7e-10	12.29	3.7e-10	21	**6.99**	3.7e-10
10 000			158.41	3.71e-10	68.40	3.71e-10	22	**15.55**	3.71e-10
15 000			554.98	3.73e-10	209.31	3.73e-10	24	**22.86**	3.73e-10
29 000			4674.35	3.74e-10	1590.37	3.74e-10	26	**52.33**	3.74e-10
30 000			–	–	1827.45	3.74e-10	26	**50.68**	3.74e-10
40 000			–	–	11214.41	3.74e-10	27	**71.44**	3.74e-10
50 000			–	–	–	–	27	**91.88**	3.74e-10
100 000			–	–	–	–	30	**189.71**	3.75e-10
1 000	1	100	**0.21**	9.63e-11	0.22	9.63e-11	15	1.59	9.64e-11
3 000			10.66	1.01e-10	5.84	1.01e-10	19	**5.65**	1.01e-10
5 000			20.62	1.01e-10	18.89	1.01e-10	21	**12.77**	1.01e-10
10 000			156.76	1.01e-10	93.16	1.01e-10	22	**28.47**	1.02e-10
50 000			–	–	–	–	27	**155.46**	1.03e-10
100 000			–	–	–	–	30	**321.84**	1.03e-10
1 000	5	50	0.27	3.71e-10	**0.19**	3.71e-10	15	1.20	3.72e-10
3 000			10.69	3.17e-10	**3.56**	3.17e-10	19	4.59	3.17e-10
5 000			20.81	3.01e-10	12.29	3.01e-10	21	**9.51**	3.02e-10
10 000			157.75	2.92e-10	68.43	2.92e-10	22	**19.67**	2.92e-10
50 000			–	–	–	–	28	**107.74**	2.88e-10
100 000			–	–	–	–	30	**230.17**	2.88e-10
1 000	5	100	0.26	2.52e-10	**0.25**	2.52e-10	15	2.21	2.52e-10
3 000			10.69	1.41e-10	**5.90**	1.41e-10	19	8.62	1.41e-10
5 000			20.78	1.35e-10	18.97	1.35e-10	21	**17.73**	1.35e-10
10 000			157.75	1.33e-10	93.58	1.33e-10	22	**36.84**	1.33e-10
50 000			–	–	–	–	28	**197.72**	1.31e-10
100 000			–	–	–	–	30	**421.02**	1.3e-10
1 000	10	50	0.26	6.57e-10	**0.18**	6.57e-10	15	1.61	6.57e-10
3 000			11.05	3.2e-10	**3.67**	3.2e-10	19	5.58	3.2e-10
5 000			20.83	2.73e-10	12.25	2.73e-10	21	**10.94**	2.73e-10
10 000			159.13	2.68e-10	69.34	2.68e-10	22	**23.09**	2.68e-10
50 000			–	–	–	–	27	**132.64**	2.56e-10
100 000			–	–	–	–	30	**293.55**	2.54e-10
1 000	10	100	**0.24**	5.27e-10	0.24	5.27e-10	15	2.88	5.25e-10
3 000			10.68	1.78e-10	**5.94**	1.78e-10	19	10.45	1.78e-10
5 000			20.84	1.73e-10	**18.97**	1.73e-10	21	20.50	1.73e-10

Table 3 continued

| N | p | n | Full svd | | svds w/ $\mathbb{S} - x\mathbb{L}$ | | svds w/ $\widetilde{\mathcal{C}}$ | | |
			Time (s)	$\mathcal{H}_2$-error	Time (s)	$\mathcal{H}_2$-error	hssrank($\widetilde{\mathcal{C}}$)	Time (s)	$\mathcal{H}_2$-error
10 000			157.48	1.65e-10	93.63	1.65e-10	22	**42.97**	1.65e-10
50 000			–	–	–	–	27	**248.91**	1.58e-10
100 000			–	–	–	–	30	**552.66**	1.58e-10
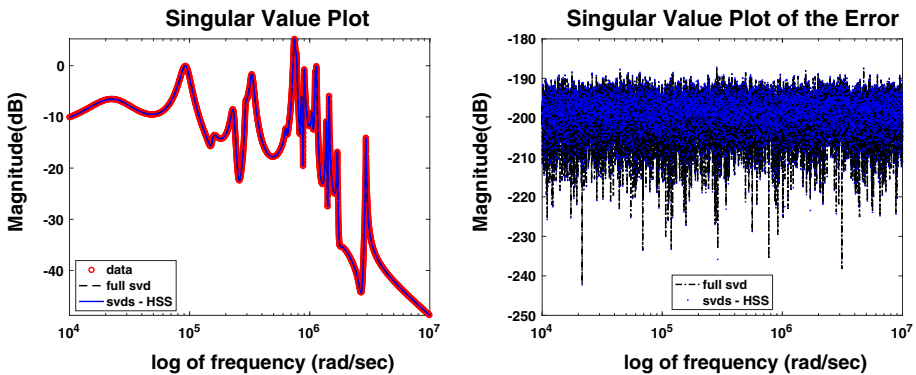
Fig. 3 Example 2. Left: Frequency response obtained with the standard and proposed approaches (in black and blue, respectively) versus the measurements (in red) for $p = 1$, $n = 50$, and $N = 10\,000$. Right: Error plots for the models obtained with the standard and proposed approaches (in black and blue, respectively) for $p = 1$, $n = 50$, and $N = 10000$ (Color figure online)

properties of the reduced model (15); clearly, our approach cannot be more accurate than the traditional Loewner framework, especially when the full SVD is performed;

5. last, but not least, the CPU time of the proposed solution depends linearly on p, n and $N \log N$ (Table 2), thus being the fastest method for large values of N; moreover, no memory constraints are present for N up to 100 000.

Figure 4 (left) we plot the computational time of the three approaches for $p = 1, n = 50$, and different values of N. Even though these are the same results as those reported in Table 3, Fig. 4 (left) clearly shows the $\mathcal{O}(N^3)$ trend of the full SVD scheme versus the $\mathcal{O}(N^2)$ trend of the svds scheme versus the $\mathcal{O}(N)$ behaviour of the proposed approach. Figure 4 (right) we depict, on a logarithmic scale, the running time of the proposed procedure for $n = 50$ and different values of N and p, clearly exhibiting a linear dependency on p and an $N \log N$ dependency with respect to N.

Example 3 In this example we compare the novel strategy presented in this paper to the fast Loewner SVD scheme illustrated in [18]. We consider the same data set as the one in Example 2, this time with $SNR = 120$ and a random $\mathbf{D} \in \mathbb{R}^{p \times p} \neq \mathbf{0}$. Due to the fact that the models resulting from the Loewner framework have $\mathbf{D} = \mathbf{0}$, a realization of size $n + p$ is needed to approximate the system with $\mathbf{D} \neq \mathbf{0}$ [26, 30].

In [18], a Galerkin-ADI method is applied to the Sylvester equation (11) satisfied by the Loewner matrix. At the k-th iteration, a low-rank approximation $P_k L_k Q_k^*$, P_k, $Q_k \in \mathbb{C}^{N \times \bar{k}}$, $L_k \in \mathbb{C}^{\bar{k} \times \bar{k}}$, to $\mathbb{L}$ is thus computed. If $U_k S_k V_k^* = L_k$ denotes the SVD of L_k, then the matrices $P_k U_k$ and $V_k^* Q_k^*$ can be used in place of $\mathbf{X}_n$ and $\mathbf{Y}_n$ in (15) to compute the reduced model.
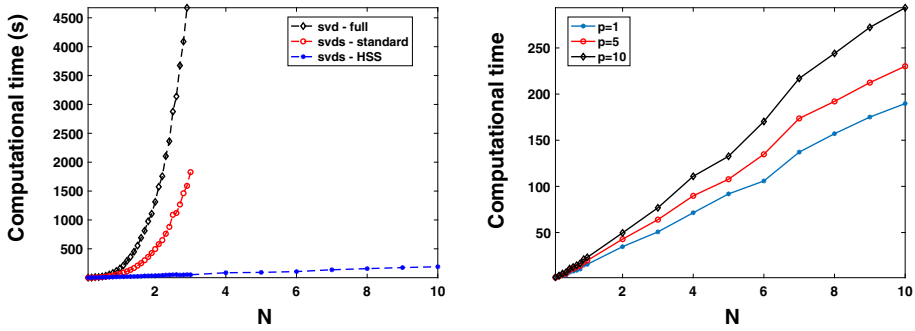
Fig. 4 Example 2. Left: Computational time achieved by the different approaches for $p = 1$, $n = 50$, and N. Right: Computational time achieved by our novel procedure for $n = 50$, and different values of N and p

The method is stopped whenever the norm of the residual matrix $\boldsymbol{M} P_k L_k Q_k^* - P_k L_k Q_k^* \boldsymbol{\Lambda} - \boldsymbol{VR} + \boldsymbol{LW}$, consisting of the left-hand side of the Sylvester equation with $\mathbb{L}$ replaced by its low-rank approximation $P_k L_k Q_k^*$, is smaller than a certain threshold ε. In the results that follow we employ $\varepsilon = 10^{-4}$, as done in [18]. At each iteration step, the SVD of L_k is truncated to keep only the $n + p$ significant values.

We consider the HALF&HALF partition of the frequencies as this is the best scenario for the scheme coming from [18]. The HALF&HALF partition often leads to a rather fast convergence of the Galerkin-ADI method in terms of number of iterations so that a quite small approximation space is constructed. If different partitions were used, the Galerkin-ADI method could be equipped with a quite involved divide-and-conquer scheme; see [18]. On the other hand, as illustrated in Example 1, the HALF&HALF partition leads to higher values of the HSS-rank of $\widetilde{\mathcal{C}}$ than for the EVEN&ODD partition with a consequent increment in the computational efforts of our scheme. In addition, as for [18], our tests employed complex arithmetic and did not solve the corresponding Sylvester equation (22) for real-coefficient matrices.

In Table 4 we report the results for $p = 10$, $n = 50$, and different values of N. Notice that even though the Galerkin-ADI approach efficiently computes the approximation spaces, the construction of the reduced model (15) still requires the allocation of both $\mathbb{L}$ and $\mathbb{S}$. Therefore, also for the Galerkin-ADI scheme severe memory constraints hold and for $N > 30\,000$, we are not able to allocate the $\mathbb{L}$ and $\mathbb{S}$ matrices with complex entries on the machine used for running the tests.

Even though the Galerkin-ADI approach is faster for $N < 20\,000$, the computed approximation spaces are quite poor. Indeed, the computed reduced models are always 7 orders of magnitude less accurate than the ones constructed by our approach. The paper [18] validates the Galerkin-ADI scheme on a system with randomly generated poles for various orders n and number of samples N but does not mention the accuracy of the resulting models. Moreover, in terms of CPU time, our results are comparable to the ones in [18] when considering the computational time solely of the Galerkin-ADI iteration, disregarding the steps involving building the full matrices and projecting these to obtain the reduced model.

The remarkable difference in the accuracy attained by the two approaches make any sort of computational comparison rather pointless. However, we would like to point out that the computational time of the Galerkin-ADI approach grows quadratically with N due to the need to assemble and store the full Loewner and shifted Loewner matrices, while an $N \log N$

Table 4 Example 3. Number of iterations, computational time (in seconds) solely of the Galerkin-ADI iteration scheme together with the total time (including building the data, the full Loewner and shifted Loewner matrices and the projection step) as well as the $\mathcal{H}_2$-error achieved by the Galerkin-ADI approach. In comparison, we list the HSS-rank, the total time (in seconds) as well as the $\mathcal{H}_2$-error of the novel scheme presented in this paper for different values of N (number of samples), $p = 10$, and $n = 50$

| | Galerkin-ADI with $\varepsilon = 10^{-4}$ | | | | svds w/ $\widetilde{\mathcal{C}}$ | | |
| | # of | Scheme | Total | | | Total | |
N	Iter.	Time(s)	Time (s)	$\mathcal{H}_2$-error	hssrank($\widetilde{\mathcal{C}}$)	Time (s)	$\mathcal{H}_2$-error
5 000	5	2.93	9.54	1.55e-2	42	100.21	2.06e-9
10 000	5	5.4	28.61	2.76e-2	46	174.04	1.81e-9
15 000	5	10.80	149.02	4.06e-2	49	280.11	1.31e-9
20 000	5	14.67	357.01	9.45e-2	50	340.42	1.17e-9
25 000	6	23.38	691.94	1.18e-2	52	426.34	9.55e-10
30 000	6	30.67	1229.48	1.17e-1	52	540.20	9.06e-10

Table 5 Example 3. Number of iterations, computational time (in seconds) solely of the Galerkin-ADI iteration scheme together with the total time (including building the data, the full Loewner and shifted Loewner matrices and the projection step) as well as the $\mathcal{H}_2$-error achieved by the Galerkin-ADI approach for $N = 5\,000$, $p = 10$, and $n = 50$

ε	# of Iter.	Scheme Time(s)	Total Time(s)	$\mathcal{H}_2$-error
10^{-4}	5	2.93	9.54	1.55e-2
10^{-12}	51	1648.02	1655.01	2.20e-3

dependency of the computational cost of our novel approach can be evidenced once again from the timings reported in Table 4.

Several ideas could be implemented to improve the accuracy of the models obtained with the Galerkin-ADI approach. In order to have the fairest comparisons with respect to our novel approach, each of these ideas will be tested separately to explore all the possibilities to enhance the Galerkin-ADI approach from [18].

First, the tolerance ε for solving the Sylvester equation via Galerkin-ADI can be chosen to a value comparable to the noise level for an SNR of 120, namely $\varepsilon = 10^{-12}$. Results are detailed in Table 5 only for the case $N = 5\,000$, $p = 10$, and $n = 50$ as the trend is obvious from this one example. While the accuracy of the model has slightly improved with respect to results obtained for $\varepsilon = 10^{-4}$, the number of iterations has also considerably increased, leading to matrices L_k of much larger dimensions for which the SVD $L_k = U_k S_k V_k^*$ becomes costly. Hence, the CPU cost of the scheme has exploded and is no longer viable. In any case, even for a tolerance value close to the noise level, the accuracy of the model is several orders of magnitude worse than with our proposed technique (10^{-3} versus 10^{-9}).

Second, it is always advisable to compute the projection subspaces from a linear combination of $\mathbb{S}$ and $\mathbb{L}$, namely $\mathbb{S} - x\mathbb{L}$ rather than only $\mathbb{L}$, as the Loewner matrix $\mathbb{L}$ encodes the strictly rational part and the addition of $\mathbb{S}$ provides all the information on the system, including its polynomial part (the **D**-term). We apply the low-rank Galerkin-ADI method to the Sylvester equation fulfilled by $\mathbb{S} - x\mathbb{L}$ thus computing a matrix $P_k Z_k Q_k^*$ such that $P_k Z_k Q_k^* \approx \mathbb{S} - x\mathbb{L}$. Results are detailed in Table 6 for the case $\varepsilon = 10^{-4}$, $N = 5\,000$, $p = 10$, and $n = 50$. For all instances considered, results were comparable in terms of CPU

Table 6 Example 3. Number of iterations, computational time (in seconds) solely of the Galerkin-ADI iteration scheme together with the total time (including building the data, the full Loewner and shifted Loewner matrices and the projection step) as well as the $\mathcal{H}_2$-error achieved by the Galerkin-ADI approach on the Sylvester equations satisfied by $\mathbb{L}$, $\mathbb{S}$ and $\mathbb{S} - x\mathbb{L}$, for $\varepsilon = 10^{-4}$, $N = 5\,000$, $p = 10$, and $n = 50$

	# of Iter.	Scheme Time(s)	Total Time(s)	$\mathcal{H}_2$-error
$\mathbb{L}$	5	2.93	9.54	1.55e-2
$\mathbb{S}$	6	3.96	10.51	5.22e-2
$\mathbb{S} - x\mathbb{L}$, $x = f(1)$	4	2.97	9.51	9.23e-2
$\mathbb{S} - x\mathbb{L}$, $x = f(N/2)$	4	2.89	9.34	9.23e-2
$\mathbb{S} - x\mathbb{L}$, $x = f(N)$	4	2.95	9.55	9.23e-2

Table 7 Example 3. Number of iterations, computational time (in seconds) solely of the Galerkin-ADI iteration scheme together with the total time (including building the data, the full Loewner and shifted Loewner matrices and the projection step) as well as the $\mathcal{H}_2$-error achieved by the Galerkin-ADI approach. In comparison, we list the HSS-rank, the total time (in seconds) as well as the $\mathcal{H}_2$-error of the novel scheme presented in this paper for different values of N (number of samples), $p = 10$, and $n = 50$ when employing real arithmetic

| | Galerkin-ADI with $\varepsilon = 10^{-4}$ | | | | svds w/ $\widetilde{\mathcal{C}}$ | | |
N	# of Iter.	Scheme Time(s)	Total Time (s)	$\mathcal{H}_2$-error	hssrank($\widetilde{\mathcal{C}}$)	Total Time (s)	$\mathcal{H}_2$-error
$\mathbb{L}$ complex	5	2.93	9.54	1.55e-2	42	100.21	2.06e-9
$\mathbb{L}$	3	1.76	50.43	1.03e-2	24	81.56	2.05e-9
$\mathbb{S}$	6	3.61	52.42	2.66e-3	24	80.97	2.05e-9
$\mathbb{S} - x\mathbb{L}$, $x = f(1)$	7	18.57	65.84	2.53e-3	24	80.84	2.05e-9
$\mathbb{S} - x\mathbb{L}$, $x = f(N/2)$	5	15.13	63.63	1.87e-2	24	82.11	2.05e-9
$\mathbb{S} - x\mathbb{L}$, $x = f(N)$	6	16.47	66.72	9.01e-2	24	82.95	2.05e-9

time to those obtained when considering solely the Sylvester equation satisfied by $\mathbb{L}$ in the Galerkin-ADI iteration (listed in the first line of Table 6 for reference), while in terms of accuracy, they are slightly worse. For this example, the sole benefit of using a linear combination $\mathbb{S} - x\mathbb{L}$ might be the system identification properties as, in principle, a sharp drop in the SVD of Z_k reveals the degree of the underlying system.

The third avenue worth exploring is employing real arithmetic and the corresponding Sylvester equations (22) and (23). Table 7 shows the results obtained using real arithmetic, both for the Galerkin-ADI scheme, as well as our proposed method. For reference, the first line in Table 7 lists the results previously obtained in complex arithmetic. For the method in [18], the cost of the scheme has mostly increased, due to more complicated Sylvester equations in (22) and (23). The CPU cost of building the data matrices, the full Loewner and shifted Loewner matrices, has also increased, yielding a total cost far superior to that obtained in complex arithmetic. In some instances, the accuracy has improved slightly. On the other hand, the real arithmetic causes the HSS-rank of the Cauchy matrix approximation to be much smaller with a remarkable impact on the CPU time and almost no effects on the model accuracy when using our novel approach.

We conclude this example by mentioning that the use of a *hybrid* approach may be fruitful. In particular, our novel approach can be employed to avoid storing the large and dense Loewner and shifted Loewner matrices. Then, the Galerkin-ADI scheme can be used

Table 8 Example 3. Computational time (in seconds) of the three individual steps in the *hybrid* approach: setting up of the data matrices, the Galerkin-ADI iteration scheme and projection to obtain the reduced model, together with the total time as well as the $\mathcal{H}_2$-error for $N = 5000$, $p = 10$, and $n = 50$ in real arithmetic

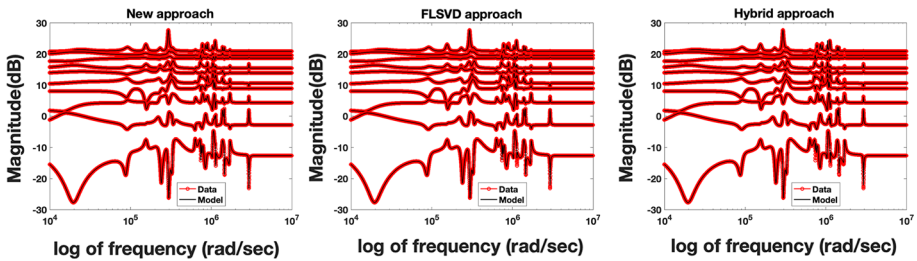| Data matrices Time(s)	Galerkin-ADI Scheme Time(s)	Projection Time(s)	Total Time(s)	$\mathcal{H}_2$-error
0.8	1.76	4.65	7.21	2.1e-2

Fig. 5 Example 3. Frequency response of the model (in black) and the measurements (in red) for $N = 5\,000$, $p = 10$, and $n = 50$ using our proposed approach, Galerkin-ADI as in [18] and the hybrid approach, employing real arithmetic
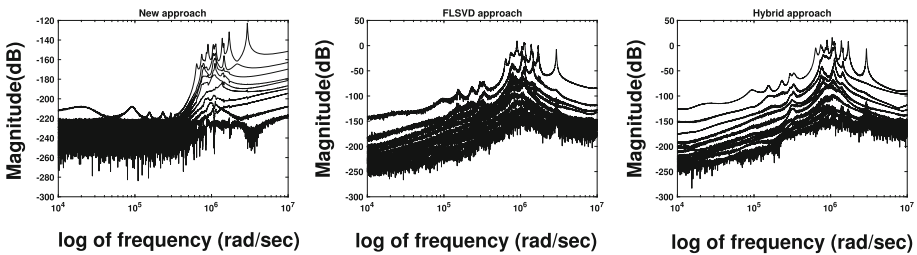
Fig. 6 Example 3. Error plots for $N = 5\,000$, $p = 10$, and $n = 50$ using our proposed approach, Galerkin-ADI as in [18] and the hybrid approach, employing real arithmetic

to compute the first dominant singular vectors of $\mathbb{S} - x\mathbb{L}$, instead of employing svds, thus also being able to identify the order of the underlying system. However, the accuracy will not be comparable to that of our proposed approach. We implemented this idea and list the CPU times of the various steps in Table 8 together with the resulting accuracy for Galerkin-ADI applied to solving the Sylvester equation (22) for $\mathbb{L}$ in real arithmetic with $\varepsilon = 10^{-4}$ for $N = 5\,000$, $p = 10$, and $n = 50$. Plots of the responses of our proposed approach, together with the Galerkin-ADI scheme as proposed in [18] and the hybrid approach are shown in Fig. 5. Even though the general shape of the response is well captured, some resonances are not modeled accurately, as expected from the much higher model errors reported earlier. This can be noticed better from the error plots in Fig. 6.

5 Conclusion

By exploiting the Cauchy-like structure of the Loewner and shifted Loewner matrices, a novel strategy for reducing the computational costs and the memory requirements of the Loewner framework has been proposed. In particular, the use of the HSS-format leads to tremendous

savings in the storage demand and computational efforts of the overall scheme. Indeed, except for the construction of $\widetilde{\mathcal{C}}$ whose cost is polylogarithmic in N, both the memory requirements and the computational cost of iteratively performing the SVD now linearly depend on the cardinality of the considered data set.

The success of our procedure strongly relies on the capability of representing the Cauchy matrix $\mathcal{C}$ in terms of an HSS-matrix $\widetilde{\mathcal{C}}$ with low (α, β) rank of the off-diagonal blocks. Even though we restricted ourselves to showing how different, but common, partitions of the frequencies affect the HSS-rank of $\widetilde{\mathcal{C}}$, a thorough analysis of their connection may be beneficial. This interesting, but tricky study will need to take into account several and diverse aspects like the compressibility in the HSS format of the matrix $\mathcal{C}$, the conditioning of $\mathbb{L}$ and $\mathbb{S}$, and the approximation properties of the underlying partition of the frequencies.

We have always computed $\widetilde{\mathcal{C}}$ at high accuracy. Results very similar to the one reported in the previous sections are obtained also with 10^{-12} as low-rank truncation threshold. However, we believe that the employment of more inexact, and thus with a lower rank, HSS-representations of $\mathcal{C}$ and its effects on the accuracy of the overall scheme may be another interesting research direction which is worth pursuing depending on the application at hand.

The strategy presented in this paper can be applied to more sophisticated problems as long as the Loewner and shifted Loewner matrices maintain a Cauchy-like structure. In particular, our approach can be employed with minor modifications in model order reduction of parametrized [21], linear switched [16], and bilinear systems [1].

Declarations

Conflicts of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethical approval The research presented in this paper is based upon work supported by the National Science Foundation under Grant No. DMS-1439786 while both the authors were in residence at the Institute for Computational and Experimental Research in Mathematics (ICERM) in Providence, RI, during the *Model and Dimension Reduction in Uncertain and Dynamic Systems* program. Even though the second half of the program had to be performed virtually due to the restrictions caused by the COVID-19 pandemic, we are extremely grateful to the organizers of the program and the whole staff of ICERM for doing whatever possible to maintain an exciting, fruitful, and high-quality working environment.
The datasets and algorithms generated during and/or analysed during the current study are available from the corresponding author on reasonable request. Moreover, the approach presented in this paper will be included in the `hm-toolbox` in the near future.

References

1. Antoulas, A.C., Gosea, I.V., Ioniţă, A.C.: Model reduction of bilinear systems in the Loewner framework. SIAM J. Sci. Comput. **38**, B889–B916 (2016)
2. Antoulas, A.C., Lefteriu, S., Ioniţă, A.C.: A tutorial introduction to the Loewner framework for model reduction, ch. 8, pp. 335–376
3. Baglama, J., Reichel, L.: Augmented implicitly restarted Lanczos bidiagonalization methods. SIAM J. Sci. Comput. **27**, 19–42 (2005)
4. Bebendorf, M.: Approximation of boundary element matrices. Numer. Math. **86**, 565–589 (2000)
5. Beckermann, B., Townsend, A.: Bounds on the singular values of matrices with displacement structure. SIAM Rev. **61**, 319–344 (2019)
6. Bouras, A., Frayssé, V.: Inexact matrix-vector products in Krylov methods for solving linear systems: a relaxation strategy. SIAM J. Matrix Anal. Appl. **26**, 660–678 (2005)
7. Carrier, J., Greengard, L., Rokhlin, V.: A fast adaptive multipole algorithm for particle simulations. SIAM J. Sci. Stat. Comput. **9**, 669–686 (1988)
8. Chandrasekaran, S., Dewilde, P., Gu, M., Lyons, W., Pals, T.: A fast solver for HSS representations via sparse matrices, SIAM J. Matrix Anal. Appl. **29**, 67–81 (2006/07)
9. Chandrasekaran, S., Gu, M., Sun, X., Xia, J., Zhu, J.: A superfast algorithm for Toeplitz systems of linear equations. SIAM J. Matrix Anal. Appl. **29**, 1247–1266 (2007)
10. Derakhtenjani, A.S., Candanedo, J.A., Chen, Y., Dehkordi, V.R., Athienitis, A.K.: Modeling approaches for the characterization of building thermal dynamics and model-based control: a case study, Science and Technology for the. Built Environ. **21**, 824–836 (2015)
11. Drineas, P., Mahoney, M.W., Muthukrishnan, S.: Relative-error CUR matrix decompositions. SIAM J. Matrix Anal. Appl. **30**, 844–881 (2008)
12. Embree, M., Ioniţă, A.C.: Pseudospectra of Loewner matrix pencils, To appear, Realization and Model Reduction of Dynamical Systems: A Festschrift in Honor of the 70th Birthday of Thanos Antoulas (2019)
13. Freitag, M.A., Spence, A.: Convergence theory for inexact inverse iteration applied to the generalised nonsymmetric eigenproblem, Electron. Trans. Numer. Anal. **28**, 40–64 (2007/08)
14. Gaaf, S.W., Simoncini, V.: Approximating the leading singular triplets of a large matrix function. Appl. Numer. Math. **113**, 26–43 (2017)
15. Gohberg, I., Olshevsky, V.: Fast algorithms with preprocessing for matrix-vector multiplication problems. J. Complexity **10**, 411–427 (1994)
16. Gosea, I.V., Petreczky, M., Antoulas, A.C.: Data-driven model order reduction of linear switched systems in the Loewner framework. SIAM J. Sci. Comput. **40**, B572–B610 (2018)
17. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comput. Phys. **73**, 325–348 (1987)
18. Hochman, A.: Fast singular-value decomposition of Loewner matrices for state-space macromodeling, in 2015 IEEE 24th Electrical Performance of Electronic Packaging and Systems (EPEPS), pp. 177–180 (2015)
19. Hochstenbach, M.E.: A Jacobi–Davidson type SVD method. SIAM J. Sci. Comput. **23**, 606–628 (2001)
20. Horn, R., Johnson, C.: Topics in Matrix Analysis. Cambridge Univ. Press (1991)
21. Ioniţă, A.C.: Lagrange rational interpolation and its applications to approximation of large-scale dynamical systems, PhD thesis, Rice University (2013)
22. Karachalios, D., Gosea, I., Antoulas, A.: Data-driven approximation methods applied to non-rational functions. Proc. Appl. Math. Mech. **18**, 1 (2018)
23. Kressner, D., Massei, S., Robol, L.: Low-rank updates and a divide-and-conquer method for linear matrix equations. SIAM J. Sci. Comput. **41**, A848–A876 (2019)
24. Kürschner, P., Freitag, M.: Inexact methods for the low rank solution to large scale Lyapunov equations, BIT Numerical Mathematics (2020)
25. Larsen, R.: Lanczos bidiagonalization with partial reorthogonalization. DAIMI Rep. Ser. **27**, 1 (1998)
26. Lefteriu, S., Antoulas, A.C.: A new approach to modeling multiport systems from frequency-domain data. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **29**, 14–27 (2010)
27. Massei, S., Palitta, D., Robol, L.: Solving rank-structured Sylvester and Lyapunov equations. SIAM J. Matrix Anal. Appl. **39**, 1564–1590 (2018)
28. Massei, S., Robol, L., Kressner, D.: hm-toolbox: MATLAB software for HODLR and HSS matrices. SIAM J. Sci. Comput. **42**, C43–C68 (2020)
29. MATLAB, version 9.9.0.1467703 (R2020b), The MathWorks Inc., Natick, Massachusetts (2020)
30. Mayo, A.J., Antoulas, A.C.: A framework for the solution of the generalized realization problem. Linear Algebra Appl. **405**, 634–662 (2007)
31. Nakatsukasa, Y.: Fast and stable randomized low-rank matrix approximation (2020)ArXiv preprint: arXiv:2009.11392

32. Palitta, D., Kürschner, P.: On the convergence of low-rank Krylov methods. Numer. Algorithm **88**, 1383–1417 (2021)
33. Pan, V.Y.: Fast approximate computations with Cauchy matrices, polynomials and rational functions, in Computer Science - Theory and Applications, Hirsch, E.A., Kuznetsov, S.O., Pin, J.-É., Vereshchagin, N.K. eds., Cham, Springer International Publishing, pp. 287–299 (2014)
34. Pan, V.Y.: Transformations of matrix structures work again. Linear Algebra Appl. **465**, 107–138 (2015)
35. Peeters, B., Van der Auweraer, H., Guillaume, P., Leuridan, J.: The PolyMAX frequency-domain method: a new standard for modal parameter estimation? Shock. Vib. **11**, 395–409 (2004)
36. Poussot-Vassal, C., Quero, C., Vuillemin, P.: Data-driven approximation of a high fidelity gust-oriented flexible aircraft dynamical model, IFAC-PapersOnLine, 51 , pp. 559 – 564. 9th Vienna International Conference on Mathematical Modelling (2018)
37. Rouet, F.-H., Li, X.S., Ghysels, P., Napov, A.: A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization. ACM Trans. Math. Softw. **42**, 1 (2016)
38. Sahouli, M., Dounavis, A.: Iterative Loewner matrix macromodeling using CUR decomposition for noisy frequency responses. In: 2019 IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), pp. 1–3 (2019)
39. Simoncini, V., Eldén, L.: Inexact Rayleigh quotient-type methods for eigenvalue computations. BIT **42**, 159–182 (2002)
40. Simoncini, V., Szyld, D.B.: Theory of inexact Krylov subspace methods and applications to scientific computing. SIAM J. Sci. Comput. **25**, 454–477 (2003)
41. Stoll, M.: A Krylov–Schur approach to the truncated SVD. Linear Algebra Appl. **436**, 2795–2806 (2012)
42. The MORwiki Community, MORwiki - Model Order Reduction Wiki. http://modelreduction.org
43. van den Eshof, J., Sleijpen, G.L.G.: Inexact Krylov subspace methods for linear systems. SIAM J. Matrix Anal. Appl. **26**, 125–153 (2004)
44. Vandebril, R., Van Barel, M., Golub, G., Mastronardi, N.: A bibliography on semiseparable matrices. Calcolo **42**, 249–270 (2005)
45. Vogel, J., Xia, J., Cauley, S., Balakrishnan, V.: Superfast divide-and-conquer method and perturbation analysis for structured eigenvalue solutions. SIAM J. Sci. Comput. **38**, A1358–A1382 (2016)
46. Xi, Y., Xia, J., Cauley, S., Balakrishnan, V.: Superfast and stable structured solvers for Toeplitz least squares via randomized sampling. SIAM J. Matrix Anal. Appl. **35**, 44–72 (2014)
47. Xia, J., Chandrasekaran, S., Gu, M., Li, X.S.: Fast algorithms for hierarchically semiseparable matrices. Numer. Linear Algebra Appl. **17**, 953–976 (2010)
48. Xia, J., Chandrasekaran, S., Gu, M., Li, X.S.: Superfast multifrontal method for large structured linear systems of equations. SIAM J. Matrix Anal. Appl. **31**, 1382–1411 (2010)