Received: 4 July 2019 | Revised: 8 July 2020 | Accepted: 19 September 2020

DOI: 10.1002/nme.6554

RESEARCH ARTICLE

WILEY

Model order reduction for delay systems by iterative interpolation

Dominik Alfke[1] | **Lihong Feng**[2] | **Luigi Lombardi**[3,4] | **Giulio Antonini**[3,4] | **Peter Benner**[2]

[1]Faculty of Mathematics, Technische Universitét Chemnitz, Chemnitz, Germany

[2]Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany

[3]Dipartimento di Ingegneria Industriale e dell'Informazione e di Economia, Universita degli Studi dell'Aquila, Avezzano, Italy

[4]Micron Semiconductor, Avezzano, Italy

Correspondence
Lihong Feng, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstrasse 1, D-39106 Magdeburg, Germany.
Email: feng@mpi-magdeburg.mpg.de

Abstract

Adaptive algorithms for computing the reduced-order model of time-delay systems (TDSs) are proposed in this work. The algorithms are based on interpolating the transfer function at multiple expansion points and greedy iterations for selecting the expansion points. The $\mathcal{L}_\infty$-error of the reduced transfer function is used as the criterion for choosing the next new expansion point. One heuristic greedy algorithm and one algorithm based on the error system and adaptive sub-interval selection are developed. Results on four TDSs with tens of delays from electromagnetic applications are presented and show the efficiency of the proposed algorithms.

KEYWORDS

delay systems, greedy interpolation, model order reduction

1 | INTRODUCTION

Time-delay systems (TDSs) frequently arise in circuit simulation especially in high-frequency applications. In high-speed circuit design, time-delay phenomena appear due to the propagation delays, for instance, caused by the transmission lines in circuit packaging and printed circuit board (PCB) design.[1] In many cases of packaging and PCB design/optimization, propagation delay can dominate circuit performance, while losses can be neglected. In general, distributed interconnects and packaging problems modeled using differential or integral equation-based methods result in large dense system matrices, the simulation of which is computationally expensive. Therefore, model order reduction (MOR) techniques have been proposed as an effective tool to reduce the computational complexity of large models, as they provide a mechanism to generate reduced-order models (ROMs) from detailed descriptions.[2] MOR techniques based on moment-matching are popular in the area of circuit simulation.[3-6]

MOR based on moment-matching for systems without delay[3,4] has been extended to delay systems in References 7-9. It is known that the transfer function of the delay system is not a rational function, such that the standard moment-matching method based on implicit moment-matching cannot be straightforwardly applied. In Reference 7, the exponential factors caused by the delays are first expanded into Taylor series, and then truncated to a certain order. The truncated Taylor series then substitutes all the exponential factors as their approximants. The transfer function can then be approximated

by a rational function, so that the moments of the transfer function can be implicitly computed by the Arnoldi algorithm. To enable applicability of the Arnoldi algorithm, the approximated system in frequency domain is first augmented to a system of dimension $n \times p$, where n is the size (order) of the original system, and p is the order of the truncated Taylor series. The work in Reference 8 extends the idea in Reference 7 to a method which matches the moments at multiple expansion points. The multiple expansion points are selected adaptively according to a binary principle (see also Reference 10). Instead of Taylor series, Reference 9 replaces each exponential factor with an equivalent power series generated by Laguerre polynomials. The power series is then truncated to an order p in order to be able to compute the projection matrix employed for MOR. The resulting approximate system in frequency domain has a similar form as the approximate system obtained by Taylor series expansion.[7] The projection matrix is then computed using the higher order Krylov subspace method proposed in Reference 11 without augmenting the system.

All the methods above try to reuse the conventional algorithms of computing moments by approximating the original non-rational transfer function with a rational function using truncated power series expansion. The resulting moment-matching property is only satisfied for the approximate transfer function and the reduced transfer function.

An iterative rational Krylov algorithm (IRKA) is proposed in Reference 12 for reducing standard linear time invariant systems without delay. The method finds interpolation points, computes projection matrices, and constructs the ROM iteratively and adaptively, so that the transfer function of the ROM is a rational interpolation of the original transfer function. For delay systems, the transfer functions are not rational, and IRKA cannot be directly applied. A recent work in Reference 13 proposes a modified IRKA algorthim, which can be used for reducing the delay systems. The idea is applying IRKA to the rational part of the transfer function. The interpolation points and projection matrices are obtained from the rational approximation. Upon convergence, the projection matrices are applied to the original delay system by Petrov-Galerkin projection to obtain the ROM with delays.

In this work, the original transfer function is *not approximated*. The MOR approach suggested here derives a reduced transfer function which interpolates the *original* transfer function at properly-chosen interpolation points. MOR for delay systems via interpolating the original transfer function previously appeared in Reference 14, where interpolation points are trivially picked. Here we propose an adaptive procedure, where the interpolation points are selected via a greedy algorithm. At each iteration of the greedy algorithm, the point causing the $\mathcal{L}_\infty$ error between the original transfer function and the reduced transfer function is selected and is used to compute new basis vectors which are then combined with the previous basis vectors to compute a new ROM.

Computing the $\mathcal{L}_\infty$-error usually has a computational complexity depending on the order n of the original system, and therefore is time-consuming. In Reference 15, a fast algorithm for computing the $\mathcal{L}_\infty$-norm of a large-scale system is proposed, where the computations are mainly based on the ROM. Motivated by the method, we apply the algorithm in Reference 15 to computing the $\mathcal{L}_\infty$-norm of the error system we need. Besides, we propose a heuristic method for directly computing the $\mathcal{L}_\infty$-error from limited frequency samples. To achieve the required accuracy, it is found that not many samples are needed, so that the computations depending on the original system are not as costly as one might think. The performances of the proposed algorithms are tested by four delay systems with many delays from electromaganetics, showing promising efficiency for MOR of time delay systems. In contrast, only a small system with a single delay is tested in Reference 14.

2 | SOME PRELIMINARIES

We review some theoretical basics on delay systems and interpolation properties of the reduced transfer function for delay systems, which the proposed algorithms in the next section are based on. The delay system we consider is defined as below.

Definition 1. A linear TDS with constant coefficients is a collection of constant system matrices and delays

$$E_0, \dots, E_d, A_0, \dots, A_d \in \mathbb{C}^{n\times n}, \quad B \in \mathbb{C}^{n\times m}, \quad C \in \mathbb{C}^{p\times n}, \\ 0 = \tau_0 < \tau_1 < \dots < \tau_d, \tag{1}$$

associated with the time-domain delay-differential equation, also known as differential-difference equation (DDE). When at least one $\tau_j, j \geq 1$, associated with the state derivative $\dot{x}$ is nonzero, it is called neutral delay-differential equation

(NDDE).

$$\sum_{j=0}^{d} E_j \dot{x}(t-\tau_j) = \sum_{j=0}^{d} A_j x(t-\tau_j) + Bu(t),$$
$$y(t) = Cx(t) \quad \forall\, t \geq 0, \tag{2}$$

with an initial condition

$$x(t) = \Phi(t) \quad \forall\, t \in [-\tau_d, 0], \quad \Phi : [-\tau_d, 0] \to \mathbb{C}^n. \tag{3}$$

The DDE maps an *input* or *control* function $u : [0,\infty) \to \mathbb{C}^m$ to an *output* function $y : [0,\infty) \to \mathbb{C}^p$ using an internal *state* function $x : [-\tau_d, \infty) \to \mathbb{C}^n$. Here n is called the order of the delay system.

The transfer function of the above delay system can be obtained by Laplace transform and has the following form:

$$H(s) = C\left(s\sum_{j=0}^{d} E_j e^{-s\tau_j} - \sum_{j=0}^{d} A_j e^{-s\tau_j} \right)^{-1} B. \tag{4}$$

MOR of DDEs can be achieved by computing ROM via Petrov-Galerkin projection using two projection matrices W, V, and has the same delays as the original system, that is,

$$\begin{aligned} \sum_{j=0}^{d} \hat{E}_j \dot{z}(t-\tau_j) &= \sum_{j=0}^{d} \hat{A}_j z(t-\tau_j) + \hat{B}u(t), \quad \forall\, t \geq 0, \\ \hat{y}(t) &= \hat{C}z(t), \end{aligned} \tag{5}$$

where $\hat{E}_j = W^T E_j V \in \mathbb{R}^{r\times r}, \hat{A}_j = W^T A_j V \in \mathbb{R}^{r\times r}, \hat{B} = W^T B \in \mathbb{R}^{r\times m}, \hat{C} = CV \in \mathbb{R}^{p\times r}$, with $r \ll n$ being the order of the ROM. The original state vector $x(t)$ in Equation (2) can be recovered by the approximation: $x(t) \approx Vz(t)$. Likewise, the transfer function of the ROM is

$$H_r(s) = \hat{C}\left(s\sum_{j=0}^{d} \hat{E}_j e^{-s\tau_j} - \sum_{j=0}^{d} \hat{A}_j e^{-s\tau_j} \right)^{-1} \hat{B}. \tag{6}$$

Definition 2. $L_2^n(-\infty, +\infty)$ is the linear space containing vector-valued functions $f : \mathbb{R} \mapsto \mathbb{R}^n$, endowed with the norm

$$\|f\|_{L_2^n} = \left(\int_{-\infty}^{\infty} ||f(t)||^2 dt \right)^{1/2}.$$

Here and below, $||\cdot||$ denotes the Euclidean vector norm or spectral matrix norm.

Definition 3. The frequency domain $\mathcal{L}_2(\jmath\mathbb{R})$ space contains the matrix-valued functions $F : \mathbb{C} \mapsto \mathbb{C}^{p\times m}$ with finite norm

$$||F||_{\mathcal{L}_2} = \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} ||F(\jmath\omega)||^2 d\omega \right)^{1/2},$$

where $\jmath = \sqrt{-1}$ is the imaginary unit. Here, ω is the angular frequency, its relation with the temporal frequency f (unit: Hz) is $\omega = 2\pi f$. The $\mathcal{L}_\infty$-norm of an operator $M : \mathcal{L}_2 \mapsto \mathcal{L}_2$ is defined as the $\mathcal{L}_2$-induced operator norm as below

Definition 4.

$$||M||_{\mathcal{L}_\infty} := \sup_{||u||_{\mathcal{L}_2} \neq 0} \frac{||Mu||_{\mathcal{L}_2}}{||u||_{\mathcal{L}_2}}, \tag{7}$$

where $u : \mathbb{C} \mapsto \mathbb{C}^m \in \mathcal{L}_2(\jmath\mathbb{R})$ is a vector valued square-integrable function in the Laplace domain, and its $\mathcal{L}_2$-norm is $||u||_{\mathcal{L}_2} = \left(\frac{1}{2\pi}\int_{-\infty}^{\infty} u(\jmath\omega)^T u(\jmath\omega) d\omega\right)^{1/2}$, according to Definition 3.

Definition 4 implies

$$||Mu||_{\mathcal{L}_2} \leq ||M||_{\mathcal{L}_\infty} ||M||_{\mathcal{L}_2}.$$

Consequently the following error bound for the output of the ROM holds:

Theorem 1.

$$||y - y_r||_{\mathcal{L}_2} = ||Hu - H_r u||_{\mathcal{L}_2} \leq ||H - H_r||_{\mathcal{L}_\infty} ||u||_{\mathcal{L}_2}.$$

It can be further proved that

$$||M||_{\mathcal{L}_\infty} = \sup_{\omega\in\mathbb{R}} ||M(\jmath\omega)|| = \sup_{\omega\in\mathbb{R}} \sigma_{\max}(M(\jmath\omega)). \tag{8}$$

Here and below, $\sigma_{\max}(\cdot)$ is the largest singular value of a matrix. Plancherel theorem proves that the L_2^n-norm in time domain and $\mathcal{L}_2$-norm in frequency domain coincide. Therefore, the output error bound in Theorem 1 holds in time and frequency domain due to the Plancherel theorem, so that

$$||y - y_r||_2 = ||Hu - H_r u||_2 \leq ||H - H_r||_{\mathcal{L}_\infty} ||u||_2, \tag{9}$$

that is, the $||\cdot||_2$ in Equation (9) can be the L_2^n-norm in time domain, or the $\mathcal{L}_2$-norm in frequency domain.

From theorem 1 in Reference 14, we can derive the following interpolation theorem for delay systems.

Theorem 2. *Let a delay system be given by $E_0, \ldots, E_d, A_0, \ldots, A_d, \tau_1, \ldots, \tau_d, B, C$ and let $s_0 \in \mathbb{C}$ be a fixed expansion point. Define the sequence of matrices $(\mathcal{K}_k)_{k=0}^{\infty} \in \mathbb{C}^{n\times n}$ by*

$$\mathcal{K}_0 = s_0 \sum_{j=0}^{d} e^{-s_0\tau_j} E_j - \sum_{j=0}^{d} e^{-s_0\tau_j} A_j,$$

$$\mathcal{K}_k = \sum_{j=0}^{d} \frac{(-\tau_j)^{k-1}}{(k-1)!} e^{-s_0\tau_j} E_j + s \sum_{j=0}^{d} \frac{(-\tau_j)^k}{k!} e^{-s_0\tau_j} E_j - \sum_{j=0}^{d} \frac{(-\tau_j)^k}{k!} e^{-s_0\tau_j} A_j \quad \forall\, k \geq 1,$$

which satisfies $\mathcal{K}(\delta) = \sum_{k=0}^{\infty} \mathcal{K}_k \delta^k$, with $s = s_0 + \delta$. Here $\mathcal{K}(s)$ is the matrix associated with the transfer function $H(s)$, that is, $\mathcal{K}(s) = \left(s\sum_{j=0}^{d} E_j e^{-s\tau_j} - \sum_{j=0}^{d} A_j e^{-s\tau_j}\right)$, such that $H(s)=C\mathcal{K}^{-1}(s)B$. Use these to define two other sequences of matrices $(F_k)_{k=0}^{\infty} \subset \mathbb{C}^{n\times m}$ and $(G_k)_{k=0}^{\infty} \subset \mathbb{C}^{n\times p}$ recursively by

$$F_0 = \mathcal{K}_0^{-1} B, \quad F_k = -k!\mathcal{K}_0^{-1}\left(\sum_{i=0}^{k-1} \mathcal{K}_{k-i} F_i\right),$$

$$G_0 = (\mathcal{K}_0)^{-T} C^T, \quad G_k = -k!(\mathcal{K}_0)^{-T}\left(\sum_{i=0}^{k-1} \mathcal{K}_{k-i}^T G_i\right).$$

Let $V, W \in \mathbb{C}^{n\times r}$ be reduction matrices and H_r be the transfer function of the corresponding ROM. Then the following statements hold for all $b \in \mathbb{C}^m$ and $c \in \mathbb{C}^p$:

- *If $F_0 b, \ldots, F_l b \in \text{Range}(V)$, then*

$$H^{(i)}(s_0) b = H_r^{(i)}(s_0) b \quad \forall\, i = 0, \ldots, l.$$

- *If* $G_0c, \dots, G_kc \in \text{Range}(W)$*, then*

$$c^T H^{(i)}(s_0) = c^T H_r^{(i)}(s_0) \quad \forall\, i = 0, \dots, k.$$

- *If* $F_0b, \dots, F_lb \in \text{Range}(V)$ *and* $G_0c, \dots, G_kc \in \text{Range}(W)$*, then*

$$c^T H^{(i)}(s_0)b = c^T H_r^{(i)}(s_0)b \quad \forall\, i = 0, \dots, l+k+1.$$

Here, $(\cdot)^{(i)}$ *denotes the ith derivative of a function.*

Proof. Let $\mathcal{D}_{s_0}^i(f)$ denote the ith derivative of a (vector valued) function $f(s)$ at $s = s_0$. Expanding the state vector $x(s)$ in frequency domain into a power series around s_0, we get $x(\delta) = \sum_{i=0}^{\infty} \frac{D_{s_0}^i[x(s)]}{i!}\delta^i$. Inserting the power series expansions of both $\mathcal{K}(s)$ and $x(s)$ into $\mathcal{K}(s)x(s) = B$, we get

$$D_{s_0}^i[\mathcal{K}^{-1}(s)B] = D_{s_0}^i[x(s)] = F_i, \quad i = 0, \dots, N.$$

Similarly,

$$(D_{s_0}^i[C\mathcal{K}^{-1}(s)])^T = D_{s_0}^i[\mathcal{K}^{-T}(s)C^T] = G_i, \quad i = 0, \dots, N.$$

The results of the theorem are then immediate implications of theorem 1 in Reference 14. ∎

Theorem 2 shows that the reduced transfer function $H_r(s)$ and its ith ($i = 1, \dots, N$) order derivatives are tangential interpolations of the original transfer function $H(s)$ and its corresponding derivatives at the interpolation point s_0, respectively. The vectors b and c are the directions of tangential interpolation. From Theorem 2, we get the following moment-matching properties:

Lemma 1. *If* $F_0, \dots F_l \in \text{Range}(V)$*, then*

$$H^{(i)} = H_r^{(i)}, \quad i = 0, \dots, l.$$

If $G_0, \dots G_k \in \text{Range}(W)$*, then*

$$H^{(i)} = H_r^{(i)}, \quad i = 0, \dots, k.$$

If $F_0, \dots F_l \in \text{Range}(V)$ *and* $G_0, \dots G_k \in \text{Range}(W)$*, then*

$$H^{(i)} = H_r^{(i)}, \quad i = 0, \dots, l+k+1.$$

Proof. $F_i \in \text{Range}(V)$ is equivalent with $F_ie_j \in \text{Range}(V)$, where $e_j \in \mathbb{R}^n$ is the jth unit vector, $j = 1, \dots, p$, therefore we have

$$H^{(i)}e_j = H_r^{(i)}e_j, \quad j = 1, \dots, p, i = 0, \dots, l,$$

which implies

$$H^{(i)} = H_r^{(i)}, \quad i = 0, \dots, l.$$

Similarly, $G_i \in \text{Range}(W)$ is equivalent with $G_ie_j \in \text{Range}(W)$, $j = 1, \dots, m$, therefore we have

$$e_j^T H^{(i)} = e_j^T H_r^{(i)}, \quad j = 1, \dots, m, i = 0, \dots, l,$$

which implies

$$H^{(i)} = H_r^{(i)}, \quad i = 0, \dots, l.$$

Likewise, we can easily obtain the third conclusion. ∎

Remark 1. Due to complicated expressions of the higher order derivatives $F_i, G_i, i > 1$, it is preferred that only the low order derivatives are included in the subspaces Range(V) and Range(W). However, to keep the ROM as accurate as required, multiple interpolation points $s_i, i = 0, \dots, q$, should be used. For each interpolation point s_i, we have the corresponding conclusions in Theorem 2 and Lemma 1. Therefore, we get the following Proposition 1.

Proposition 1. *If* $F_0(s_i) \in \text{Range}(V), i = 1, \dots, l$, *then*

$$H(s_i) = H_r(s_i), \quad i = 1, \dots, l.$$

If $G_0(s_i) \in \text{Range}(W), i = 1, \dots, l$, *then*

$$H(s_i) = H_r(s_i), \quad i = 1, \dots, l.$$

If $F_0(s_i) \in \text{Range}(V)$ *and* $G_0(s_i) \in \text{Range}(W), i = 1, \dots, l$, *then*

$$H'(s_i) = H_r'(s_i), \quad i = 1, \dots, l.$$

Remark 2. For MIMO systems, when $p \neq m$, the two matrices V and W defined by

$$\text{Range}(V) = \text{colspan}\{F_0(s_1), \dots, F_0(s_l)\},$$

and

$$\text{Range}(W) = \text{colspan}\{G_0(s_1), \dots, G_0(s_l)\},$$

may have different column dimensions, which would lead to non-square reduced matrices. The issue can be avoided by, for example, adding certain random orthonormal vectors to the one which has less column size (see Algorithm 2), but still guarantees the interpolation properties. For MIMO systems with numerous inputs and outputs, it is not practical to include the whole input matrix B and output matrix C into the subspaces, rather, nontrivial vectors b and c are introduced, so that tangential interpolation in Theorem 2 is satisfied for each expansion point. Properly choosing tangential vectors b and c has been discussed in Reference 12 for standard linear time invariant systems with no delay, that is, for system in Equation (2) with d=0. Then the transfer function is a rational function, and the interpolation is called rational interpolation. The tangential vectors are chosen using the left and right eigenvectors (corresponding to simple eigenvalue(s)) of the ROM derived at the each iteration of Algorithm 2. The final ROM then satisfies $\mathcal{H}_2$-optimal necessary conditions. For TDSs, there is not a general rule for choosing the tangential vectors. It is not discussed in Reference 14 either. We find that choosing b and c as the left and right singular vectors corresponding to the largest singular value of the reduced transfer function leads to a time-delay ROM with good accuracy. However, no $\mathcal{H}_2$-optimality can be guaranteed.

In this work, we aim to use a greedy algorithm to iteratively select the interpolation (expansion) points $s_i = \jmath\omega$ for the projection matrices V and W in Proposition 1, so that the reduced transfer function and its first derivative interpolate those of the original transfer function, respectively.

3 | GREEDY INTERPOLATION

This section introduces the proposed methods and presents the greedy algorithms for iteratively choosing the interpolation points $s_i, i = 1, \dots, l$.

The idea is summarized in Algorithm 1.

Algorithm 1. Schematic process of greedy interpolation for delay systems

Input: Delay system, frequency interval $[\underline{\omega}, \overline{\omega}]$, maximum number of iterations $k_{\max}$, deflation tolerance ε.
Output: Projection matrices V, W such that $\|H(\jmath\omega) - H_r(\jmath\omega)\| > \varepsilon$ for all $\omega \in [\underline{\omega}, \overline{\omega}]$.

1: Choose initial frequency $\omega_1 \in [\underline{\omega}, \overline{\omega}]$.
2: $V \leftarrow \emptyset$, $W \leftarrow \emptyset$.
3: **for** $k = 1, \ldots, k_{\max}$ **do**
4: Compute V_{s_k}, W_{s_k} based on Proposition 1 so that $H(s)$ and its first derivative are interpolated, respectively, by $H_r(s)$ at $s_k = \jmath\omega_k$.
5: Expand V and W by $V = \text{orth}\{V, V_{s_k}\}$, $V = \text{orth}\{V, W_{s_k}\}$, (e.g., using the modified Gram-Schmidt process with deflation tolerance ε.)
6: Compute new reduced transfer function H_r.
7: Solve the Maximization Problem $\max_{\omega \in [\underline{\omega}, \overline{\omega}]} \|H(\jmath\omega) - H_r(\jmath\omega)\|$ to obtain $\omega_{k+1} \leftarrow \arg\max_{\omega \in [\underline{\omega}, \overline{\omega}]} \|H(\jmath\omega) - H_r(\jmath\omega)\|$.
8: **if** $\|H(\jmath\omega_{k+1}) - H_r(\jmath\omega_{k+1})\| > \varepsilon$ **then**
9: **return** .
10: **end if**
11: **end for**

At each step, a point $s_k = \jmath\omega_k$ which produces the $\mathcal{L}_\infty$-norm of the error ($\mathcal{L}_\infty$-error) is selected as the new interpolation point. Then the corresponding two matrices V_{s_k}, W_{s_k} are computed so that

$$F_0(s_k) \in \text{Range}(V_{s_k}),$$

$$G_0(s_k) \in \text{Range}(W_{s_k}).$$

From Equation (8), when the compact interval $[\omega, \overline{\omega}]$ is large enough, $\max_{\omega \in [\omega, \overline{\omega}]} \|H(i\omega) - H_r(i\omega)\|$ in Step 7 approximates the $\mathcal{L}_\infty$-norm of the error, that is, $\|H - H_r\|_{\mathcal{L}_\infty}$. Step 8 indicates the stopping criteria of the method, that is, the algorithm stops selecting new interpolation points, once the $\mathcal{L}_\infty$-norm of the error is below an error tolerance. This is in agreement with the error bound in Theorem 1.

The remaining issues are the computation of V_{s_k}, W_{s_k} for each interpolation point in Step 4, and solving the optimization problem in Step 8. If the system matrices of the original delay system are real matrices, it is desired that the system matrices of the ROM are also real. Note that according to the definition of the $\mathcal{L}_\infty$-norm, the interpolation points are selected along the imaginary axis, so that V_{s_k}, W_{s_k} are complex matrices. We can instead use

$$\text{Range}(V_{s_k}) := \text{colspan}\{\text{Re}F_0(s_k), \text{Im}F_0(s_k)\},$$

$$\text{Range}(W_{s_k}) := \text{colspan}\{\text{Re}G_0(s_k), \text{Im}G_0(s_k)\}$$

to generate real matrices V_{s_k}, W_{s_k}. This is equivalent to implicitly adding the complex conjugate of s_k as a second interpolation point, due to the equality

$$\text{Range}\left(\text{Re}F_0(s_k), \text{Im}F_0(s_k)\right) = \text{Range}\left(F_0(s_k), F_0(\overline{s_k})\right),$$

where the identity $\overline{F_0(s_k)} = F_0(\overline{s_k})$ is used. The detailed computation of V_{s_k}, W_{s_k} is described in Algorithm 2, where a matrix pair V_{s_k}, W_{s_k} is computed and then orthogonalized against the columns of the current projection matrices V, W. The orthogonalization process $\text{orth}\{\cdot, \cdot\}$ can be done by, for example, the modified Gram-Schmidt process with deflation. Finally, Algorithm 2 provides the updated matrix pair V, W. It actually implements Steps 4 and 5 in Algorithm 1.

Algorithm 2. Update the projection matrices by interpolating at a single point s_k

Input: Current projection matrices V, W, new interpolation point s_k.
Output: Updated projection matrices $V,\ W$ after interpolation at s_k.
1: Setup $K(s_k) = s_k \sum_{j=0}^{d} e^{-s_k\tau_j}E_j - \sum_{j=0}^{d} e^{-s_k\tau_j}A_j$.
2: $F \leftarrow K(s_k)^{-1}B$.
3: $G \leftarrow (K(s_k))^{-T}C^T$.
4: **if** the system matrices are real **then**
5: $F \leftarrow \big(\mathrm{Re}\, F, \mathrm{Im}\, F\big)$,
6: $G \leftarrow \big(\mathrm{Re}\, G, \mathrm{Im}\, G\big)$.
7: **end if**
8: $V = \mathrm{orth}\{V, F\}$.
9: $W = \mathrm{orth}\{W, G\}$.
10: $r_1 \leftarrow$ column number of V, $r_2 \leftarrow$ column number of W.
11: **if** $r_1 > r_2$ **then**
12: Expand W by $r_1 - r_2$ random orthonormal columns.
13: **else if** $r_1 < r_2$ **then**
14: Expand V by $r_2 - r_1$ random orthonormal columns.
15: **end if**

3.1 | Heuristically computing the $\mathcal{L}_\infty$-norm

The key point of Algorithm 1 is solving the optimization problem in Step 7. A heuristic method is simply computing $\|H(\jmath\omega) - H_r(\jmath\omega)\|$ at preassigned samples of ω and selecting one sample which corresponds to the biggest error. Using this method, Step 7 in Algorithm 1 can be modified to, $\omega_{k+1} \leftarrow \arg\max_{\omega\in\Xi}\|H(\jmath\omega) - H_r(\jmath\omega)\|$, where Ξ is a set of samples of $\omega \in \mathbb{R}$, that is, $\Xi \subset \mathbb{R}$. Combining Algorithm 2 and the heuristic optimization technique with Algorithm 1 gives the heuristic greedy interpolation Algorithm 3. Note that the original transfer function $H(\jmath\omega)$ needs to be computed at all samples in Ξ, but only once. During the iterations, those values can be repeatedly used.

Algorithm 3. Heuristic greedy interpolation for delay systems

Input: Delay system, a training set Ξ including evenly distributed samples taken from the frequency interval $[\underline{\omega}, \overline{\omega}]$, maximum number of iterations $k_{\max}$, tolerance ε.
Output: Projection matrices W, V such that $\|H(\jmath\omega) - H_r(\jmath\omega)\| < \varepsilon$ for all $\omega \in [\underline{\omega}, \overline{\omega}]$.
1: Pre-compute $H(\jmath\omega_i)$ for all $\omega_i \in \Xi$.
2: Choose initial frequency $\omega_1 \in [\underline{\omega}, \overline{\omega}]$.
3: $V \leftarrow \emptyset,\ W \leftarrow \emptyset$.
4: **for** $k = 1, \ldots, k_{\max}$ **do**
5: Call Algorithm 2 to expand V and W by interpolating at $s_k = \jmath\omega_k$.
6: Compute new reduced transfer function H_r.
7: Solve the maximization problem $\max_{\omega\in\Xi}\|H(\jmath\omega) - H_r(\jmath\omega)\|$ to obtain $\omega_{k+1} \leftarrow \arg\max_{\omega\in\Xi}\|H(\jmath\omega) - H_r(\jmath\omega)\|$.
8: **if** $\|H(\jmath\omega_{k+1}) - H_r(\jmath\omega_{k+1})\| < \varepsilon$ **then**
9: **return** .
10: **end if**
11: **end for**

Although it is easy to implement, the method might be unreliable, since it is unknown how many samples should be included in Ξ. If too few samples are included in Ξ, ω_{k+1} is probably not the global optimal value. As can also be seen from the simulation results in Section 5, the performance of the method is not stable: it works well for some examples, while

not for others. Often, about 100 samples are sufficient to produce reliable ROMs, though it is not always the case. This means the original transfer function $H(\jmath\omega)$ needs to be evaluated at 100 samples. In the next subsection, we propose a new method for solving the optimization problem, which is theoretically more robust. We will compare these two methods in the following sections.

3.2 | Computing $\mathcal{L}_\infty$-norm from the reduced error system

Solving the optimization problem in Step 7 of Algorithm 1 involves computing the original transfer function at each iteration step, which implies solving a number of linear systems with the large scale matrix $\mathcal{K}(s)$ being the coefficient matrix, that is,

$$\mathcal{K}(s)x_i(s) = b_i,\ i = 1, \dots, p,\ B := (b_1, \dots, b_m),$$

such that $H(s) = CX(s)$, $X(s) := (x_1(s), \dots, x_m(s))$. To avoid computations involving the original large dimension, an iterative algorithm for computing $\|H\|_{\mathcal{L}_\infty}$ is proposed in Reference 15, where $H(s)$ stands for the transfer function of any large-scale system. Note that $H_e(s) := H(s) - H_r(s)$ is actually the transfer function of the following error system

$$\begin{aligned} E_{e_j} &= \begin{pmatrix} E_j & \\ & -E_{j,r} \end{pmatrix}, \quad A_{e_j} = \begin{pmatrix} A_j & \\ & -A_{j,r} \end{pmatrix}, \\ B_e &= \begin{pmatrix} B \\ B_r \end{pmatrix}, \quad C_e = \begin{pmatrix} C & C_r \end{pmatrix}, \quad j = 0, \dots, d, \end{aligned} \tag{10}$$

since

$$H_e(s) = C_e \begin{pmatrix} K(s) & \\ & -K_r(s) \end{pmatrix}^{-1} B_e = H(s) - H_r(s). \tag{11}$$

Therefore the method of computing $\|H\|_{\mathcal{L}_\infty}$ in Reference 15 can also be applied to computing $\|H_e\|_{\mathcal{L}_\infty}$, that is,

$$\|H_e\|_{\mathcal{L}_\infty} = \|H - H_r\|_{\mathcal{L}_\infty} := \sup_{\omega\in\mathbb{R}} \|H(\jmath\omega) - H_r(\jmath\omega)\|. \tag{12}$$

From Equation (12), we see that the optimization problem in Algorithm 1 is equivalent with computing $\|H_e\|_{\mathcal{L}_\infty}$. Following the idea in Reference 15, instead of computing $\|H_e\|_{\mathcal{L}_\infty}$, a local ROM of the error system, noted as $\hat{\Sigma}_e$, is first computed, then $\|H_e\|_{\mathcal{L}_\infty}$ is obtained via $\|\hat{H}_e\|_{\mathcal{L}_\infty}$, where $\hat{H}_e(s)$ is the transfer function of $\hat{\Sigma}_e$. The ROM $\hat{\Sigma}_e$ is updated iteratively, until the stopping criterion is satisfied. Since all the matrices involved in computing $\hat{H}_e(s)$ are of reduced sizes, the computational complexity is expected to be reduced. The ROM of the error system $\hat{\Sigma}_e$ is updated at each iteration step, such that $\sigma_{\max}(\hat{H}_e(s))$ satisfies the following Hermite interpolation conditions at a selected frequency ω_k, that is,

$$\sigma_{\max}(\hat{H}_e(\jmath\omega_k)) = \sigma_{\max}(H_e(\jmath\omega_k)), \quad \sigma'_{\max}(\hat{H}_e(\jmath\omega_k)) = \sigma'_{\max}(H_e(\jmath\omega_k)), \tag{13}$$

where $\sigma'_{\max}(\cdot)$ is the first-order derivative of $\sigma_{\max}$ (as a function of ω) at ω_k. ω_k is the maximizer of $\|\hat{H}_e(\jmath\omega)\|_{\mathcal{L}_\infty}$, that is,

$$\omega_k = \arg\max_{\omega\in\mathbb{R}} \|\hat{H}_e(\jmath\omega)\|.$$

It is proved in Reference 15 that when the above Hermite interpolation properties are satisfied, and when the interpolation points are contained in a bounded interval $\mathcal{I} \in \mathbb{R}$, the sequences ω_k superlinearly converge to a local maximizer of $\|H_e(\jmath\omega)\|_2$. Algorithm 4 describes the details of applying the method from Reference 15 to the computation of $\|H_e\|_{\mathcal{L}_\infty}$.

Remark 3. In Algorithm 4, the projection matrices W, V are initially computed in Steps 2-7, and iteratively updated in Steps 13-19. At each iteration, the reduced transfer function $\hat{H}_e(s)$ computed from W, V is guaranteed to satisfy the Hermitian interpolation property in Equation (13), see lemma 3.1 in Reference 15 for the detailed analysis and proof. $(\cdot)^*$ denotes the conjugate transpose of a matrix. The stopping criterion for Algorithm 4 is

$$|\omega_k - \omega_{k-1}| < \varepsilon \text{ or } k > k_{\max}.$$

Algorithm 4 converges locally. In Reference 15, to reduce the possibility of stagnating at a local maximizer which is not a global one, r_0 initial interpolation points are simultaneously used to compute V_1, W_1, instead of only one. The initial interpolation points are chosen in an interval $[0, \omega_{\max}]$, where $\omega_{\max}$ is problem-dependent. In practical computations, the optimization is also done in $[0, \omega_{\max}]$. This is because $H(\bar{s}) = \overline{H(s)}$ for systems with real-valued matrices. Therefore, the lower bound for ω can be set to zero. The upper bound $\omega_{\max}$ also has to be given, in order to use some optimization solvers, for example, `eigopt` suggested in Reference 15 in Step 12. Since $\omega_{\max}$ is usually problem dependent, it is set to a large number if no better choice is available. Though no global convergence proof is given in Reference 15, the algorithm seems to be fairly robust and no example for failure is known to the authors.

Algorithm 4. Computing $\|H_e\|_{\mathcal{L}_\infty}$ using the method in Reference [15]

Input: The error system in (10), transfer function $H_e(s)$.$\omega_1 \leftarrow$ a random number in $\mathbb{R}$.
Output: $\omega_k = \arg\max_{\omega\in\mathbb{R}} \sigma(\hat{H}_e(\jmath\omega)) \approx \arg\max_{\omega\in\mathbb{R}} \sigma(H_e(\jmath\omega))$
1: Setup $K_e(\jmath\omega_k) = \jmath\omega_k \sum_{j=0}^{d} e^{-\jmath\omega_k\tau_j} E_{e_j} - \sum_{j=0}^{d} e^{-\jmath\omega_k\tau_j} A_{e_j}$, $k = 1, \ldots$.
2: **if** $m = p$ **then**
3: $V_1 \leftarrow K_e(\jmath\omega_1)^{-1} B_e$ and $W_1 \leftarrow (C_e K_e(\jmath\omega_1)^{-1})^*$.
4: **else if** $m>p$ **then**
5: $V_1 \leftarrow K_e(\jmath\omega_1)^{-1} B_e$ and $W_1 \leftarrow (C_e K_e(\jmath\omega_1)^{-1})^* H_e(\jmath\omega_1)$.
6: **else**
7: $V_1 \leftarrow K_e(\jmath\omega_1)^{-1} B_e H_e(\jmath\omega_1)^*$ and $W_1 \leftarrow (C_e K_e(\jmath\omega_1)^{-1})^*$.
8: **end if**
9: $W = W_1$, $V = V_1$.
10: **for** k=2,3,... **do**
11: Form $\hat{H}_e = \hat{C}_e \left(s \sum_{j=0}^{d} \hat{E}_{e_j} e^{-s\tau_j} - \sum_{j=0}^{d} \hat{A}_{e_j} e^{-s\tau_j} \right)^{-1} \hat{B}_e$, where $s = \jmath\omega_k$, $\hat{E}_{e_j} = W^T E_{e_j} V, \hat{A}_{e_j} = W^T A_{e_j} V, \hat{B}_e = W^T B_e, \hat{C}_e = C_e V$.
12: Set $\omega_k = \arg\max_{\omega\in\mathbb{R}} \sigma_{\max}(\hat{H}_e(\jmath\omega))$.
13: **if** $m = p$ **then**
14: $V_k = K_e(\jmath\omega_k)^{-1} B_e$ and $W_k = (C_e K_e(\jmath\omega_k)^{-1})^*$.
15: **else if** $m<p$ **then**
16: $V_k = K_e(\jmath\omega_k)^{-1} B_e$ and $W_k = (C_e K_e(\jmath\omega_k)^{-1})^* H_e(\jmath\omega_k)$.
17: **else**
18: $V_k = K_e(\jmath\omega_k)^{-1} B_e H_e(\jmath\omega_k)^*$ and $W_k = (C_e K_e(\jmath\omega_k)^{-1})^*$.
19: **end if**
20: $W = \text{orth}\{W, W_k\}$, $V = \text{orth}\{V, V_k\}$.
21: **end for**

Combining Algorithms 2 and 4 with Algorithm 1, we get the final greedy algorithm for computing the ROM of the delay system (2) which is detailed in Algorithm 5.

The only difference between Algorithm 3 and Algorithm 5 lies in computing $\|H_e\|_{\mathcal{L}_\infty} := \sup_{\omega\in\mathbb{R}} \|H_e(\jmath\omega)\|$. For Algorithm 3, once $H(\jmath\omega)$ is computed at all samples of ω in Ξ, $\|H_e(\jmath\omega)\|$ can be evaluated quickly within one loop. For Algorithm 5, Algorithm 4 needs to be called to compute $\|H_e(\jmath\omega)\|_{\mathcal{L}_\infty}$, where an extra loop of iteration in Step 10-21 must be implemented.

Algorithm 5. Greedy interpolation algorithm by iteratively expanding W, V

Input: Delay system in (2), frequency interval $[\underline{\omega}, \overline{\omega}]$, maximal number of iterations $k_{\max}$, tolerance ε.
Output: Projection matrices W, V such that $\|H(\jmath\omega) - H_r(\jmath\omega)\| > \varepsilon$ for all $\omega \in [\underline{\omega}, \overline{\omega}]$.

1: Choose initial frequency $\omega_1 \in [\underline{\omega}, \overline{\omega}]$.
2: $V \leftarrow \emptyset$, $W \leftarrow \emptyset$.
3: **for** $k = 1, \dots, k_{\max}$ **do**
4: Call Algorithm 2 to expand V and W by interpolating at $s_k = \jmath\omega_k$.
5: Compute new reduced transfer function H_r.
6: Call Algorithm 4 to solve the maximization problem $\max_{\omega\in\mathbb{R}} \|H(\jmath\omega) - H_r(\jmath\omega)\|$ and obtain $\omega_{k+1} \leftarrow \arg\max_{\omega\in\mathbb{R}} \|H_e(\jmath\omega)\|$.
7: **if** $\|H(\jmath\omega_{k+1}) - H_r(\jmath\omega_{k+1})\|_2 < \varepsilon$ **then**
8: **return** .
9: **end if**
10: **end for**

3.3 | Implementing the algorithm in subintervals

It is observed that Algorithm 4 may not converge until $\hat{H}_e(s)$ approximates $H_e(s)$ sufficiently well, which may result in a ROM $\hat{\Sigma}_e$ of the error system with a large reduced order, especially when the frequency interval $[0, \omega_{\max}]$ is too big. Consequently, the optimization problem associated with its transfer function $\hat{H}_e(s)$ in Step 12 of Algorithm 4 may be too costly. To further improve the efficiency of Algorithm 5, we propose to divide the whole interval into subintervals and implement Algorithm 4 in subintervals, so that the final interpolation point is selected from the local optimal frequencies in the subintervals. When each subinterval is small, a ROM $\hat{\Sigma}_e$ with high accuracy and small reduced order can be built locally, so that the optimization in Step 12 of Algorithm 4 can be quickly implemented in each subinterval. As a result, the overall computational costs are reduced.

Recall that Algorithm 5 iteratively computes interpolation points $\jmath\omega_k$, $k = 1, \dots, K$, so that the reduced transfer function H_r interpolates H at those points. This means the error $H_e(\omega) = \|H(\jmath\omega) - H_r(\jmath\omega)\|$ is zero at these points. This particularly indicates that $H_e(\jmath\omega)$ can never be concave in intervals that contain ω_k as an interior point. For this reason, it is natural to use the interpolation points as the endpoints of the subintervals.

Algorithm 6 modifies the greedy interpolation Algorithm 5 in a way that Algorithm 4 is only called to solve the maximization problem on subintervals $[\omega_{k-1}, \omega_k]$, $k > 1$. We thus get solutions to the maximization problem

$$e^\star_{[\omega_{k-1},\omega_k]} = \|H_e(\omega^\star_{[\omega_{k-1},\omega_k]})\| = \max_{\omega\in[\omega_{k-1},\omega_k]} \|H_e(\jmath\omega)\|, \tag{14}$$

on each subinterval $[\omega_{k-1}, \omega_k]$. The solution to the overall maximization problem can then be found by comparing the maxima $e^\star_{[\omega_{k-1},\omega_k]}$ for the subintervals.

Algorithm 6 brings an additional advantage. In the kth iteration, one new expansion point ω_k (in Step 16) is added, splitting the interval $[\omega_{\bar{\zeta}-1}, \omega_{\bar{\zeta}}]$ into two new subintervals. For all the other intervals, the local maximizers $\omega^\star_{[\omega_{i-1},\omega_i]}$ can be further used as the initial guess for the solution of Equation (14) in the next iteration. When ω_k is found, the indices of some ω_i need to be re-ordered, and the index of the endpoint $\overline{\omega}$ needs to be increased by one, to make $\omega_i \le \omega_{i+1}$ for all $i \le k+1$, this is done in Step 8 of Algorithm 6.

In fact, not all the maximizers of the subintervals need to be updated at each iteration step. Through addition of the new interpolation point ω_k in the kth iteration of Algorithm 6, the interval $[\omega_{\bar{\zeta}-1}, \omega_{\bar{\zeta}}]$ containing ω_k is divided into two new subintervals. On these new subintervals, the shape of $\|H(\jmath\omega) - H_r(\jmath\omega)\|_2$ will differ a lot from the previous iteration, as $\|H(\jmath\omega_k) - H_r(\jmath\omega_k)\|_2 = 0$, while it was a maximum at the previous iteration. Thus, new maximizers $e^\star_{[\omega_{\bar{\zeta}-1},\omega_k]}$, $e^\star_{[\omega_k,\omega_{\bar{\zeta}}]}$ in these new intervals should be computed.

On the other hand, there are also $k-1$ *old* intervals which are not divided, and whose new maxima $e^\star_{[\omega_{i-1},\omega_i]}$ need to be computed too. However, there will be a lot of intervals where $e^\star_{[\omega_{i-1},\omega_i]}$ does not change significantly after an update, especially in the later stages of the algorithm. This observation motivates a *lazy* approach in order to further reduce the

computational efforts. The idea is that instead of computing new maximizers $\omega^{\star}_{[\omega_{i-1},\omega_i]}$ on *all old* intervals, the *old* maximizers are first compared with the maximizers $e^{\star}_{[\omega_{\bar{\zeta}-1},\omega_k]}$, $e^{\star}_{[\omega_k,\omega_{\bar{\zeta}}]}$ in the newly divided subintervals. If any of the *old* maximizers, say $\omega^{\star}_{[\omega_{i_0-1},\omega_{i_0}]}$ once again produces the maximal value $e^{\star}_{[\omega_{i_0-1},\omega_{i_0}]}$ among all of them, the maximization problem in the interval which contains $\omega^{\star}_{[\omega_{i_0-1},\omega_{i_0}]}$ is then solved to obtain an updated new maximizer. The *old* maximizer on the other intervals need not be updated. If it happens that the finally chosen maximal value $e^{\star}_{[\omega_{\bar{\zeta}-1},\omega_{\bar{\zeta}}]}$ is below the error tolerance, then the maximization problems in all the other untouched intervals need to be solved to get their new maximizer in order not to miss the global maximizer. Algorithm 7 illustrates the details of the approach.

Algorithm 6. Greedy interpolation algorithm using subintervals

Input: Delay system in (2), frequency interval $[\underline{\omega}, \overline{\omega}]$, maximal number of iterations $k_{\max}$, tolerance ε.

Output: Projection matrices $W, V \in \mathbb{C}^{n\times r}$ such that the reduced transfer function H_r satisfies $\|H(\jmath\omega) - H_r(\jmath\omega)\| < \varepsilon, \forall\, \omega \in [\underline{\omega}, \overline{\omega}]$.

1: $\omega_0 \leftarrow \underline{\omega}$.
2: $V \leftarrow \emptyset,\ W \leftarrow \emptyset$.
3: $\omega_1 \leftarrow$ the first (randomly) chosen interpolating point.
4: **for** $k = 1, 2, \dots, k_{\max}$ **do**
5: Call Algorithm 2 to expand V and W by interpolation at $s = \jmath\omega_k$.
6: Compute transfer function $H_e(s)$.
7: $\omega_{k+1} = \overline{\omega}$.
8: Reorder the indices i of the endpoints ω_i to make $\omega_0 \le \omega_1 \cdots \le \omega_{k+1}$.
9: **for** each interval $[\omega_{i-1}, \omega_i]$, $1 \le i \le k+1$ **do**
10: call Algorithm 4 to compute the maximizer $\omega^{\star}_{[\omega_{i-1},\omega_i]}$ and maximal value $e^{\star}_{[\omega_{i-1},\omega_i]}$ of $\|H_e(\jmath\omega)\|_2$ inside $[\omega_{i-1}, \omega_i]$, with the initial guess as the previously computed $\omega^{\star}_{[\omega_{i-1},\omega_i]}$, if it exists.
11: **end for**
12: $[\omega_{\bar{i}-1}, \omega_{\bar{i}}] \leftarrow$ the interval with maximal value: $\bar{i} = \arg\max\limits_{1<i\le k+1} e^{\star}_{[\omega_{i-1},\omega_i]}$.
13: **if** $e^{\star}_{[\omega_{\bar{i}-1},\omega_{\bar{i}}]} < \varepsilon$ **then**
14: **return** .
15: **end if**
16: $\omega_{k+1} \leftarrow \omega^{\star}_{[\omega_{\bar{i}-1},\omega_{\bar{i}}]}$.
17: Divide the interval $[\omega_{\bar{i}-1}, \omega_{\bar{i}}]$ into two subintervals: $[\omega_{\bar{i}-1}, \omega_{k+1}], [\omega_{k+1}, \omega_{\bar{i}}]$.
18: **end for**

3.4 | Methods for the optimization problem in Algorithm 4

All the above three algorithms repeatedly need to call Algorithm 4 to solve the optimization problem and find $\arg\max_{\omega\in\mathbb{R}} \|H_e(\jmath\omega)\|$. The question arises what method to use for the maximization problem: Step 12 in Algorithm 4. We propose two possibilities: one is the software `eigopt` suggested and used in the original paper,[15] the other possibility is a sample approach analogous to what is used in Algorithm 3.

- `Eigopt` is a software package based on evaluating the objective function and its derivative at quite a large number of points, and then using the so-called *quadratic support functions* to estimate $\|H_e(\jmath\omega)\|_2$ in a certain frequency range.

  The advantages of using `Eigopt` is the theoretical validation in Reference 15, as well as the exploitation of the first derivative of the objective function, which we can easily compute according to Lemma 2 below. Extremely high computational cost is the most obvious disadvantage. Please refer to the results in Table 2 in Section 5.
- A sample approach. We evaluate the objective function $\|\hat{H}_e(\jmath\omega)\|_2$ at N samples in a sample set Ξ. Then we use $\arg\max_{\omega\in\Xi} \|\hat{H}_e(\jmath\omega)\|$ to approximate $\arg\max_{\omega\in\mathbb{R}} \|H_e(\jmath\omega)\|$. The only difference of this approach from Step 8 of the heuristic Algorithm 3 is that instead of computing $\|H_e(\jmath\omega)\|_2$ which involves computing $H(\jmath\omega)$ at all samples in Ξ, $\|\hat{H}_e(\jmath\omega)\|_2$ with reduced dimension is computed. Here, the sample set Ξ might be different from that used in

Algorithm 3, depending on the interval in which the optimization is implemented. For Algorithm 5, the samples are taken from the whole interval, so Ξ could be the same as the one for Algorithm 3. For the other algorithms, Ξ only contains samples in a subinterval where the optimization is implemented. The number of samples in such a set Ξ can be much smaller.

Algorithm 7. Greedy interpolation algorithm on selected subintervals

Input: Delay system in (2), frequency interval $[\underline{\omega}, \overline{\omega}]$, maximal number of iterations $k_{\max}$, tolerance ε.
Output: Projection matrices $V, W \in \mathbb{C}^{n\times r}$ such that $\|H(i\omega) - H_r(i\omega)\|_2 > \varepsilon, \forall\, \omega \in [\underline{\omega}, \overline{\omega}]$.

$V \leftarrow \emptyset$, $W \leftarrow \emptyset$, $r \leftarrow 0$.
$\omega_1 \leftarrow$ the first randomly chosen interpolation point.
$\omega_{-1} \leftarrow \underline{\omega}$, $\omega_0 \leftarrow \overline{\omega}$.
$\mathcal{I} \leftarrow \{[\omega_{-1}, \omega_0]\}$. *% Set of all subintervals.*
$\bar{i} = 0$. *% $\bar{i}$ defines the interval in $\mathcal{I}$ containing the next interpolation point.*
for $k = 1, 2, \ldots$ **do**
  Use Algorithm 2 to expand V and W by interpolating at $s = \jmath\omega_k$.
  Divide the interval $[\omega_{\bar{i}-1}, \omega_{\bar{i}}]$ into two intervals $[\omega_{\bar{i}-1}, \omega_k]$ and $[\omega_k, \omega_{\bar{i}}]$ and delete $[\omega_{\bar{i}-1}, \omega_{\bar{i}}]$ from $\mathcal{I}$.
  Call Algorithm 4 to compute the maximizers $\omega^\star_{[\omega_{\bar{i}-1},\omega_k]}$ and $\omega^\star_{[\omega_k,\omega_{\bar{i}}]}$ and maximal values $e^\star_{[\omega_{\bar{i}-1},\omega_k]}$ and $e^\star_{[\omega_k,\omega_{\bar{i}}]}$ in the intervals $[\omega_{\bar{i}-1}, \omega_k]$ and $[\omega_k, \omega_{\bar{i}}]$, respectively.
  Compare $e^\star_{[\omega_{\bar{i}-1},\omega_k]}$ and $e^\star_{[\omega_k,\omega_{\bar{i}}]}$ with other $e^\star_{[\omega_{i-1},\omega_i]}$ in the non-divided intervals and choose a maximum.
  Define the new interval with the newly chosen maximum:$[\omega_{\bar{i}-1}, \omega_{\bar{i}}] \leftarrow$ the interval in $\mathcal{I}$ with the maximum.
  while $[\omega_{\bar{i}-1}, \omega_{\bar{i}}]$ is one of the non-divided intervals and its maximizer is not yet updated **do**
    Use Algorithm 4 to recompute the maximizer$\omega^\star_{[\omega_{\bar{i}-1},\omega_{\bar{i}}]}$ in $[\omega_{\bar{i}-1}, \omega_{\bar{i}}]$.
    Compare all $e^\star_{[\omega_{i-1},\omega_i]}$ of all intervals and choose a new maximum.
    Repeat Step 12.
  end while
  if $e^\star_{[\omega_{\bar{i}-1},\omega_{\bar{i}}]} < \varepsilon$ **then**
    Use Algorithm 4 to update $\omega^\star_{[\omega_{i-1},\omega_i]}$ and $e^\star_{[\omega_{i-1},\omega_i]}$ for all subintervals in $\mathcal{I}$ where these values have not been updated yet.
    Choose a new maximum: $[\omega_{\bar{i}-1}, \omega_{\bar{i}}]$.
    if $e^\star_{[\omega_{\bar{i}-1},\omega_{\bar{i}}]} < \varepsilon$ **then**
      return
    end if
  end if
  $\omega_{k+1} \leftarrow \omega^\star_{[\omega_{\bar{i}-1},\omega_{\bar{i}}]}$.
end for

Lemma 2. *Consider a complex matrix function $M : \mathbb{C} \mapsto \mathbb{C}^{p\times m}$ that is analytic in the point $i\omega_0$, $\omega_0 \in \mathbb{R}$. Suppose the largest singular value $\sigma_{\max}(M(\jmath\omega_0))$ is simple, and let u_0 and v_0 be the left and right singular vectors for that singular value, with $\|u_0\|_2 = \|v_0\|_2 = 1$. Then the singular value function*

$$\varphi : \Omega \to \mathbb{R}, \quad \varphi(\omega) = \|M(\jmath\omega)\|_2 = \sigma_{\max}(M(\jmath\omega)), \tag{15}$$

defined on a small real neighborhood $\Omega \subset \mathbb{R}$ of ω_0, is two times differentiable in ω_0 and its derivatives are given by

$$\frac{\mathrm{d}\varphi}{\mathrm{d}\omega}(\omega_0) = -\mathrm{Im}(u_0^* M' v_0) \quad \text{and}$$
$$\frac{\mathrm{d}^2\varphi}{\mathrm{d}\omega^2}(\omega_0) = -\mathrm{Re}(u_0^* M'' v_0) - \begin{pmatrix} M'v_0 \\ M'^* u_0 \end{pmatrix}^* \begin{pmatrix} \varphi(\omega_0) I_p & -M \\ -M^* & \varphi(\omega_0) I_m \end{pmatrix}^+ \begin{pmatrix} M'v_0 \\ M'^* u_0 \end{pmatrix},$$

with the abbreviations $M = M(\jmath\omega_0)$, $M' = \frac{\mathrm{d}M}{\mathrm{d}s}(\jmath\omega_0)$, *and* $M'' = \frac{\mathrm{d}^2M}{\mathrm{d}s^2}(\jmath\omega_0)$. *Here,* $(\cdot)^+$ *denotes the Moore-Penrose pseudo-inverse of a matrix.*

Proof. The eigenvalues of the matrix function

$$Z : \Omega \to \mathbb{C}^{(m+p)\times(m+p)}, \quad Z(\omega) = \begin{pmatrix} 0 & M(\jmath\omega) \\ M(\jmath\omega)^* & 0 \end{pmatrix},$$

are equal to the singular values of $M(\jmath\omega)$ as well as their negative counterparts, so $\varphi(\omega)$ is a simple eigenvalue of $Z(\omega)$. The differentiability and derivative formulas are proven by the results on eigenvalue differentiation in References 16 and 17. ∎

4 | COMPARISON OF COMPUTATIONAL COSTS

In this section, we analyze the computational costs of Algorithm 3 and Algorithm 7 to show the main difference between the two methods. The computational costs of Algorithms 5 and 6 can also be analogously estimated. For simplicity, we assume the delay system has the same number of inputs and outputs, that is, $m = p$.

- The Heuristic Algorithm 3: Although the original transfer function needs to be computed at each sample in Ξ, this is done only once, and can be repeatedly used in the *for* loop in Algorithm 3. This means the full system (original transfer function) needs to be evaluated at least N times, where N is the cardinality of the set Ξ. Note that at Step 5 of Algorithm 3, we need to compute V_{s_k}, W_{s_k} by Algorithm 2, where in Steps 2-3, $m + p = 2m$ linear systems of full dimension need to be solved. The computational costs of solving the linear systems are almost equal to evaluating the original transfer function at s_k twice. Therefore, the total number of full system evaluations in Algorithm 3 by using the Heuristic optimization method is $2K + N$, where K is the total number of iterations of Algorithm 3. In the next section, we find that in most cases $N = 100$ is sufficient to give a good ROM with error less than $\varepsilon = 1 \times 10^{-4}$. Note that to solve the maximization problem in Step 7, the ROM also needs to be evaluated for N times at each iteration. Finally, there will be KN ROM evaluations.
- Algorithm 7: as discussed in Section 3.3, Algorithm 7 implements the interpolation method on selected subintervals, and is the most efficient choice compared to Algorithms 5 and 6. We use this algorithm as an example to analyze the computational costs. Algorithm 4 actually contributes to the main computational costs, because computation of the projection matrices V_i, W_i in Steps 2-8 and Steps 13-19 include the matrix computations of the FOM. Computational costs for a single pair of V_i, W_i are equivalent to two original transfer function evaluations. If Algorithm 4 takes l_1 (on average) iterations to converge, then it needs $2l_1$ evaluations of the full system (transfer function) in each subinterval and at each iteration of Algorithm 7. Assume that l_2 (on average) subintervals have been updated at each iteration step, and $\tilde{K}$ iterations have been taken by Algorithm 7, then the total number of full system evaluations should be $2(l_1 * l_2 + 1)\tilde{K}$ by considering the full system evaluation in Step 7 when expanding V, W from Algorithm 2. The optimization problem (Step 12) in Algorithm 4 only involves ROM evaluations. If $\tilde{N}$ ROM evaluations need to be done in Step 12 of Algorithm 4, then there will be $\tilde{N} * l_1 * l_2 * \tilde{K}$ ROM evaluations in total.

In the next section, we compare the proposed Algorithm 3 and Algorithms 5-7 by their applications to four delay systems from electromagnetic (EM)/circuit design.

5 | NUMERICAL RESULTS

This section demonstrates the simulation results of Algorithm 3 and Algorithms 5-7 for solving four delay systems, which come from modelling of EM systems through the PEEC method[18] and have tens to hundred of delays. Algorithms 5-7 present three different ways of computing the approximate $\mathcal{L}_\infty$-norm, respectively, that is, Algorithm 5 considers the whole frequency interval as a single interval; Algorithm 6 divides the frequency interval into subintervals, and implements optimization (Algorithm 4) in each subinterval; Algorithm 7 divides the frequency interval into subintervals, but only

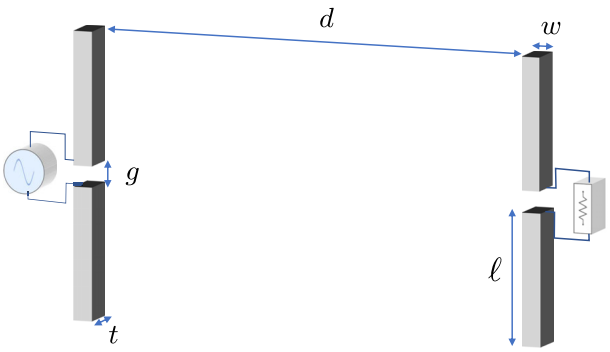


FIGURE 1 Transmitting and receiving dipole antennas

TABLE 1 Comparison of the proposed algorithms for the dipole antennas

Method name	Runtime	r	Validated error	FOM eval.	ROM eval.
H-greedy, $N = 50$	1567s	28	1.3e-3	64	350
H-greedy, $N = 100$	3507s	32	2.2e-05	116	800
OI-greedy, $N = 50$	27,716s	28	5.6e-04	70	1785
OI-greedy, $N = 100$	78,174s	32	2.2e-05	92	4646
ASI-greedy, $N = 5$	12,797s	28	2.7e-3	250	750
ASI-greedy, $N = 10$	25,872s	28	3.6e-05	316	1738
SSI-greedy, $N = 5$	8941s	28	3.3e-3	196	546
SSI-greedy, $N = 10$	20,552s	28	3.6e-05	262	1364

implements optimization (Algorithm 4) in selected subintervals. For convenience of comparison, we name Algorithm 3 and Algorithms 5-7 as follows,

- H-greedy: Algorithm 3 (Heuristic).
- OI-greedy: Algorithm 5 (One interval).
- ASI-greedy: Algorithm 6 (All subintervals).
- SSI-greedy: Algorithm 7 (Selected subintervals).

The results for Example 1 and the smaller model of Example 2 are obtained on a laptop with Intel(R) Core (TM) i7-5500U CPU@2.40GHz. Simulations for the larger model of Example 2 and Example 3 are run on a computer server with 4 Intel Xeon E7-8837 CPUs running at 2.67 GHz. 1TB main memory, split into four 256 GB partitions.

Example 1 (Transmitting and receiving dipole antennas). As the first model, a couple of transmitting and receiving dipole antennas is considered, see Figure 1. The length, width, and thickness of the dipole conductors are $\ell = 10$ cm, $w = 1$ mm, and $t = 1$ mm, respectively; the gap between the dipole conductors is $g = 1$ mm and the distance from transmitter to receiver is $d = 10$ cm. Both the dipoles are terminated on a 73 Ω resistor. The delayed partial element equivalent circuit (PEEC) method[18] has been used to describe the electromagnetic problem in a circuit form which is described by the DDEs in (2). The number of unknowns for the dipoles geometry is $n = 4016$ with 74 different delays besides the delay-less portion of the system. The model has 2 inputs and 2 outputs.

We first compare Algorithm 3 and Algorithms 5-7 in Table 1. The termination criterion for all algorithms is chosen as $\|H(j\omega) - H_r(j\omega)\|_2 < \varepsilon$ for all $\omega \in [0, \omega_{\max}]$, with tolerance $\varepsilon = 1 \times 10^{-4}$. For this model, $\omega_{\max} = 2 \times 10^{10}$. Algorithm 4 is commonly used in Algorithms 5-7. For the optimization problem in Step 12 of Algorithm 4, we use the sample approach described in Section 3.4. $\varepsilon = 1 \times 10^{-4}$ is also used as the stopping criterion of Algorithm 4 (see Remark 3).

In Table 1, and all the tables below (if applicable),

- r is the order of the ROM.

TABLE 2 Comparison of optimization solver used in Algorithm 4

Method name	Runtime	r	Validated error	FOM eval.	ROM eval.
Eigopt	530,180s	32	3.4739e-05	388	18,802
Sample approach, $N = 10$	20,552s	28	3.6301e-05	262	1342

- Runtime: the total runtime of each algorithm.
- Validated error: The reduction results are validated on 1000 sample points between 0 and $\omega_{\max}$. The validated error is the maximum value of $\|H(\jmath\omega) - H_r(\jmath\omega)\|_2$ on the 1000 sample points.
- FOM eval.: number of full system evaluations as analyzed in Section 4.
- ROM eval.: number of ROM evaluations as analyzed in Section 4.
- N: number of samples in Ξ taken from a certain frequency interval. For H-greedy or OI-greedy, the frequency interval is the whole interval $[0, \omega_{\max}]$. For ASI-greedy or SSI-greedy, the samples are taken from a single subinterval $[\omega_{i-1}, \omega_i] \in [0, \omega_{\max}]$, therefore N is much smaller.
- Iter. i: ith iteration of an algorithm.
- Inter.ω: the interpolation frequency ω; Inter.f: the interpolation frequency f, $\omega = 2\pi f$. For each model, $\jmath\omega$ is the interpolation point selected at each iteration. For ease of notation, we only list the values of ω or f in the relevant tables below.
- Offline time: the time of constructing the ROM.

We see that for $N = 50$, validated errors reveal that the ROMs computed by both H-greedy and OI-greedy have errors larger than the error tolerance. Similar cases happen to the ASI-greedy and SSI-greedy for $N = 5$. This means using a heuristic optimization method by deciding the optimizer from a finite number of candidate samples cannot always give good results. In order to ensure obtaining valid ROMs, dense enough sample sets need to be used. For all the algorithms above, $N = 100$ for the whole interval, or $N = 10$ for each subinterval are sufficient to give results with errors below the error tolerance.

Among the algorithms giving accurate results, the H-greedy algorithm with $N = 100$ needs much less time than the other algorithms, therefore, is most efficient. This can be seen from the number of FOM evaluations and ROM evaluations needed by each algorithm shown in Table 1. In Section 4, we have analyzed the FOM and ROM evaluations involved in H-greedy and SSI-greedy, respectively. From Table 1, we see SSI-greedy in fact needs more FOM evaluations than H-greedy. Besides, many ROM evaluations have been done by SSI-greedy. Therefore, it is not surprising that it takes more time than H-greedy. Comparing OI-greedy with ASI-greedy and SSI-greedy, we see that OI-greedy has many more ROM evaluations than the other two algorithms, though it needs less FOM evaluations. Consequently, OI-greedy takes much longer to converge. ASI-greedy outperforms OI-greedy, which indicates the advantage of doing optimization in each subinterval over optimization in the whole large interval. SSI-greedy is more efficient than ASI-greedy, showing the efficiency of implementing optimization only in the selected subintervals.

Next, we compare the two optimization solvers used in Algorithm 4, the algorithm commonly employed by OI-greedy, ASI-greedy, and SSI-greedy. We use SSI-greedy (Algorithm 7) to show the results listed in Table 2.

The sample approach leads to much less runtime than `eigopt`. It is clear from the number of FOM and ROM evaluations used by each algorithm. `eigopt` needs many more ROM evaluations than the sample approach, indicating that ROM evaluations cannot be neglected when they reach a certain amount. The validated errors show both solvers give rise to accurate ROMs. Figure 2A shows the $\|H(\jmath\omega) - H_r(\jmath\omega)\|_{\mathcal{L}_\infty}$ error decay computed by each algorithm.

ASI-greedy and SSI-greedy iterate less than the other two algorithms. This may illustrate the benefit of performing optimization in subintervals. OI-greedy needs the same number of iterations as H-greedy. This, nevertheless, does not mean it converges faster, since it has one more level of inner iterations (Algorithm 4) than H-greedy. Similarly, ASI-greedy and SSI-greedy consume the same number of iterations and more interesting is that their error decay behaves exactly the same. This means, both algorithms have chosen the same interpolation points ω_k at each iteration. It justifies the robustness of adaptively selecting the subintervals by the SSI-greedy. As optimization is only done in the selected subintervals, SSI-greedy converges much faster than ASI-greedy as shown in Table 1.

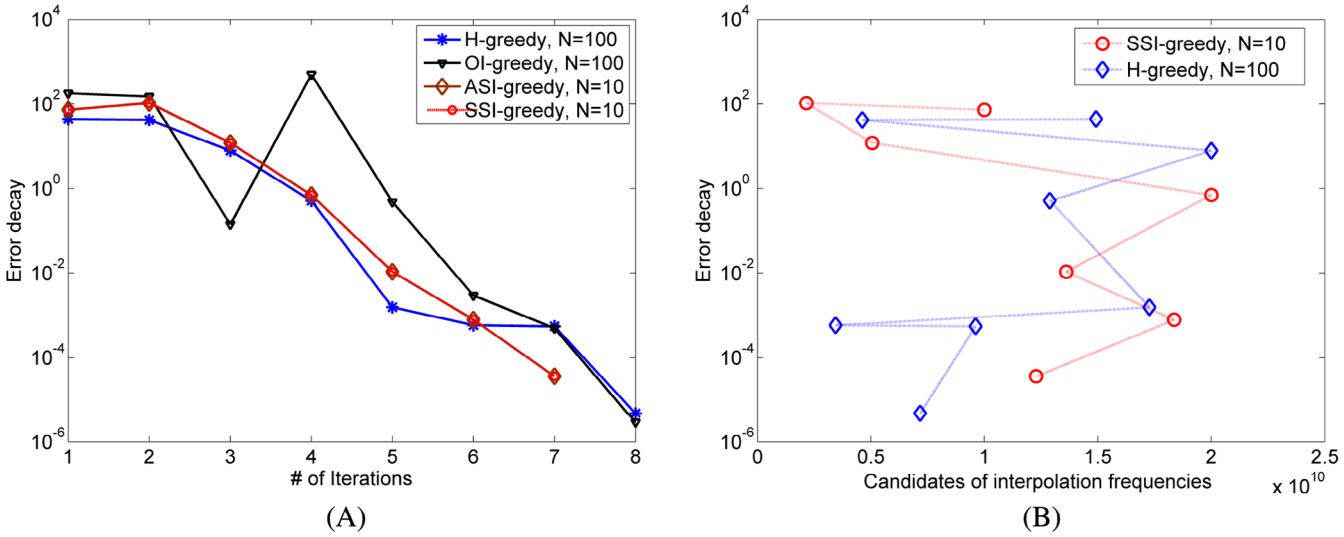


FIGURE 2 $\mathcal{L}_\infty$ error behavior, dipole antennas

TABLE 3 Selected intervals by SSI-greedy ($N = 10$) for the dipole antennas

Iterations	Inter. ω	Current all intervals	Selected intervals	Local maximizers
Iter. 1	1e10	$\mathcal{I}_1, \mathcal{I}_2$	$\mathcal{I}_1, \mathcal{I}_2$	2.2e9, 1.4e10
Iter. 2	2.2e09	$\mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4$	$\mathcal{I}_3, \mathcal{I}_4$	1e4, 5e9, 1.4e10
Iter. 3	5.e09	$\mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_5, \mathcal{I}_6$	$\mathcal{I}_2, \mathcal{I}_5, \mathcal{I}_6$	1e4, 4.2e9, 7.4e9, 2e10
Iter. 4	2e10	$\mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_5$-$\mathcal{I}_7$	$\mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_7$	1e4, …, 1.4e10, 2e10
Iter. 5	1.4e10	$\mathcal{I}_3, \mathcal{I}_5$-$\mathcal{I}_9$	$\mathcal{I}_5, \mathcal{I}_6, \mathcal{I}_8, \mathcal{I}_9$	1e4, …, 1.8e10, 2e10
Iter. 6	1.8e10	$\mathcal{I}_3, \mathcal{I}_5$-$\mathcal{I}_8, \mathcal{I}_{10}, \mathcal{I}_{11}$	$\mathcal{I}_8, \mathcal{I}_{10}, \mathcal{I}_{11}$	1e4, …, 1.2e10, …
Iter. 7	1.2e10	$\mathcal{I}_3, \mathcal{I}_5$-$\mathcal{I}_7, \mathcal{I}_9$-$\mathcal{I}_{13}$	$\mathcal{I}_3, \mathcal{I}_5$-$\mathcal{I}_7, \mathcal{I}_{10}$-$\mathcal{I}_{13}$	5.5e8, 3.8e9, 7.4e9, …

Figure 2B plots the error decay w.r.t. the selected frequency interpolation points at each iteration of H-greedy and SSI-greedy, respectively. Note that instead of $\jmath\omega$, only the frequency ω is plotted. The midpoint $\jmath\omega$ with $\omega = 1 \times 10^{10}$ is taken as the initial interpolation point for SSI-greedy, while H-greedy uses $\jmath\omega$ with $\omega = \arg\max_{\omega \in [0, \omega_{\max}]} \|H(\jmath\omega)\|_2$ as the initial interpolation point. Further different points are selected according to the different errors of the reduced transfer function produced by the two algorithms at each iteration. Define the subintervals (of ω) divided by the interpolation points selected by SSI-greedy with $N = 10$ according to their appearances as

$$\begin{aligned}
&\mathcal{I}_1 = [1e4, 1e10], && \mathcal{I}_2 = [1e10, 2e10], && \mathcal{I}_3 = [1e4, 2.2e09], && \mathcal{I}_4 = [2.2e09, 1e10],\\
&\mathcal{I}_5 = [2.2e09, 5.e09], && \mathcal{I}_6 = [5.e09, 1e10], && \mathcal{I}_7 = [2e10, 2e10], && \mathcal{I}_8 = [1e10, 1.4e10],\\
&\mathcal{I}_9 = [1.4e10, 2e10], && \mathcal{I}_{10} = [1.4e10, 1.8e10], && \mathcal{I}_{11} = [1.8e10, 2e10], && \mathcal{I}_{12} = [1e10, 1.2e10],\\
&\mathcal{I}_{13} = [1.2e10, 1.4e10].
\end{aligned}$$

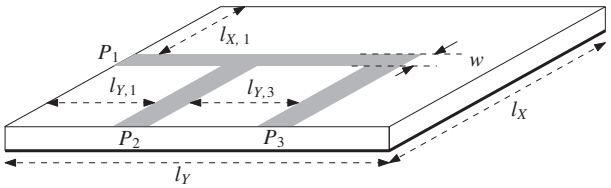
Table 3 shows which subintervals have been selected for updating at each iteration of SSI-greedy. The local maximizers are those points at which the error $\|\hat{H}_e(s)\|_2$ is maximized in their corresponding subintervals. The number of the local maximizers is equal to the number of current total subintervals. Note that, once a new interpolation point is selected, the interval which includes the point is divided into two subintervals. The interval will be replaced by the two subintervals in the later iterations.

At each iteration, if a subinterval is not updated, its corresponding local maximizer remains unchanged. For example, at Iter.2, $\mathcal{I}_2$ is not updated, so its local maximizer 1.4e10 remains unchanged. At Iter.3, $\mathcal{I}_3$ is not updated, so its local

TABLE 4 Runtime for the dipole antennas

FOM runtime 21,580s over 1000 frequency samples.			
Method	**ROM runtime**	**Offline time**	**Speed-up**
H-greedy, $N = 100$	19.5s	3507s	6.1
SSI-greedy, $N = 10$	4s	20,552s	1.1
FOM runtime 215,200s over 10,000 frequency samples.			
Method	**ROM runtime**	**Offline time**	**Speed-up**
H-greedy, $N = 100$	140s	3507s	59
SSI-greedy, $N = 10$	119s	20,552s	10

FIGURE 3 The three-port microstrip power-divider circuit



maximizer 1*e*4 remains unchanged. With the number of subintervals increasing, the number of local maximizers also increases. To keep the table within the page size, we do not list all the local maximizers at Iter. 4-7, except the one which is selected as the interpolation frequency for the next iteration. It is clear that the interpolation frequencies in the second column of Table 3 are selected from the local maximizers from the last iteration. Except for the first iteration, the number of updated subintervals is always less than the number of all the subintervals, which further justifies the advantage of SSI-greedy over ASI-greedy.

In Table 4, we compare the runtime spent on computing the transfer function $H(s)$ of the FOM at 1000 frequency points with the runtime of computing the reduced transfer function $H_r(s)$ at the same frequencies. We use the two algorithms: H-greedy and SSI-greedy for comparison. It can be seen that simulating the ROM is much faster than simulating the FOM. By including the offline time of constructing the ROM, we have achieved the best speed-up factor of 6.1. However, the SSI-greedy has only a little speed-up for this model. Nevertheless, if the FOM needs to be evaluated at more frequency points, for example, over 10,000 frequency points, then speed-ups of 59 and 10 are obtained, respectively. Much better results have been obtained for the following splitter model.

Example 2 (Three port splitter). The second example is a three-port microstrip power-divider circuit has been modeled. The structure is shown in Figure 3 (P_1, P_2 and P_3 denote the ports). The dimensions of the circuit are [20, 20, 0.5] mm in the $[x, y, z]$ directions and the width of the microstrips is set as 0.8 mm. Furthermore, the dimensions l_{X1}, $l_{Y\,1}$, and $l_{Y\,3}$ are 9, 7.2, and 7.2 mm, respectively. The relative dielectric constant is $\epsilon_r = 2.2$. As before, the delayed PEEC method[18] has been used to describe the electromagnetic problem and cast it in a circuit form which admits a DDE model (2). We use two different mesh sizes to derive two systems with different numbers of unknowns. The first system has $n = 5019$ unknowns with 88 different delays besides the delay-less portion of the system. The frequency range of this system is $[0, \omega_{\max}] = [0, 2\pi \times 10^{10}]$. The second system has $n = 10,626$ unknowns with 93 delays. The frequency range of the second system is $[0, \omega_{\max}] = [0, 4\pi \times 10^{10}]$.

Tables 5 and 6 show simulation results for the three port splitter. Here, the results of OI-greedy and ASI-greedy are not presented, as they are much less efficient than SSI-greedy. The results of SSI-greedy using the optimization solver `eigopt` is not shown either, because of its many inner-loop iterations taken by Algorithm 4 and the resulting extremely long runtime. Unfortunately, H-greedy cannot provide sufficiently accurate ROMs for the three-port splitter model in general, except for the smaller system with $N = 100$. The results are not surprising for this heuristic algorithm. If N is further increased, better results can be obtained. However, this requires some *try-and-error* approach and is not fully automatic. SSI-greedy produces ROMs meeting the error requirement $\epsilon = 1 \times 10^{-4}$ for both systems of this model. H-greedy needs less FOM and ROM evaluations and therefore uses less runtime than SSI-greedy.

Trends of the error changes $\|H(j\omega) - H_r(j\omega)\|_{\mathcal{L}_\infty}$ of each algorithm for the smaller system are plotted in Figure 4A. For each algorithm, the two different cases of different N have almost the same error decay in the first few 2-3 iterations,

TABLE 5 Comparison of the proposed algorithms for the splitter with $n = 5019$

Method name	Runtime	r	Validated error	FOM eval.	ROM eval.
H-greedy, $N = 50$	2295s	30	5.1e-03	66	400
H-greedy, $N = 100$	4500s	48	6.7949e-05	116	400
SSI-greedy, $N = 5$	8267s	48	8.2827e-05	288	816
SSI-greedy, $N = 10$	9850s	48	8.1736e-05	282	1463

TABLE 6 Comparison of the proposed algorithms for the splitter with $n = 10,626$

Method name	Runtime	r	Validated error	FOM eval.	ROM eval.
H-greedy, $N = 50$	18,291s	60	1.6	70	500
H-greedy, $N = 100$	37,204s	66	1e-02	122	1100
SSI-greedy, $N = 5$	97,662s	84	4.3159e-05	552	1572
SSI-greedy, $N = 10$	121,830s	78	7.2937e-05	600	3157

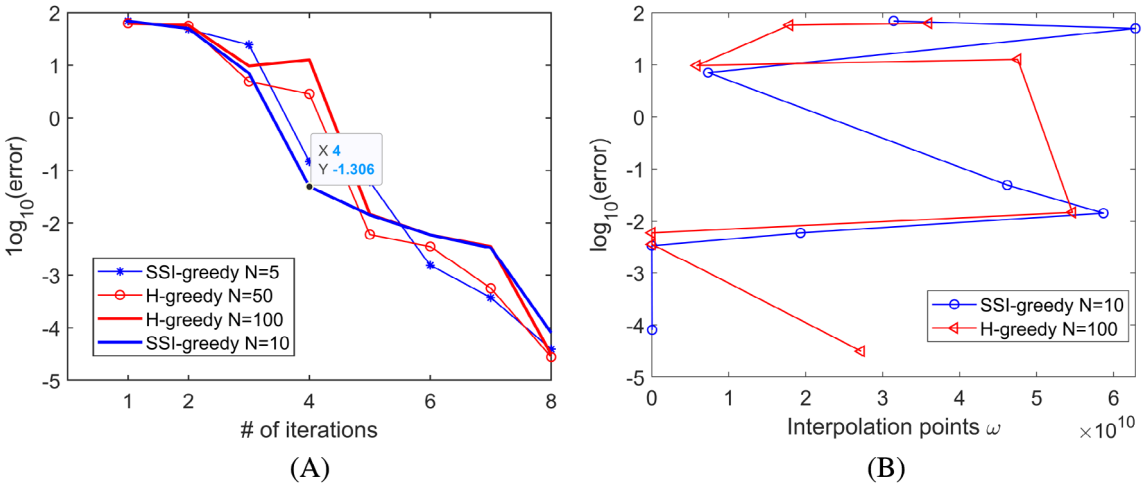


FIGURE 4 $\mathcal{L}_\infty$ error behavior, splitter with $n = 5,019$

implicating the same interpolation points ω_k have been computed. Figure 4B indicates the error decay w.r.t. the selected interpolation points at each iteration of H-greedy and SSI-greedy, respectively.

Analogously, we list in Table 7 the interpolation frequencies, selected subintervals, as well as the local maximizers computed by SSI-greedy for the splitter model with smaller size $n = 5019$. The subintervals (of f) appearing in the SSI-greedy iteration are

$$\begin{aligned}
&\mathcal{I}_1 = [1e4, 5e9], && \mathcal{I}_2 = [5e9, 1e10], && \mathcal{I}_3 = [1e10, 1e10], && \mathcal{I}_4 = [1e4, 1.2e9],\\
&\mathcal{I}_5 = [1.2e9, 5e9], && \mathcal{I}_6 = [5e9, 7.3e9], && \mathcal{I}_7 = [7.3e9, 1e10], && \mathcal{I}_8 = [7.3e9, 9.3e9],\\
&\mathcal{I}_9 = [9.3e9, 1e10], && \mathcal{I}_{10} = [1.2e9, 3e9], && \mathcal{I}_{11} = [3e9, 5e9] && \mathcal{I}_{12} = [1e4, 1e4],\\
&\mathcal{I}_{13} = [1e4, 6.5e6], && \mathcal{I}_{14} = [6.5e6, 1.2e9].
\end{aligned}$$

To avoid repetition, we do not explain the results in detail. By comparing column 3 and column 4, it can be seen that optimization is performed only in some selected subintervals at most iteration steps.

In Table 8, we compare the runtime spent on computing the transfer function $H(s)$ of the FOM with the runtime of computing $H_r(s)$ at 1000 frequency points. Simulating the ROM is finished within seconds. Again, H-greedy has much

TABLE 7 Selected intervals by SSI-greedy ($N = 10$) for the splitter with $n = 5019$

Iterations	Inter.f	Current all intervals	Selected intervals	Local maximizers
Iter. 1	5e9	$\mathcal{I}_1$, $\mathcal{I}_2$	all	1.2e9, 1e10
Iter. 2	1e10	$\mathcal{I}_1$-$\mathcal{I}_3$	all	1.2e9, 7.3e9, 1e10
Iter. 3	1.2e9	$\mathcal{I}_2$-$\mathcal{I}_5$	$\mathcal{I}_2$, $\mathcal{I}_4$, $\mathcal{I}_5$	1e4, 3e9, 7.3e9, 1e10
Iter. 4	7.3e9	$\mathcal{I}_2$-$\mathcal{I}_7$	$\mathcal{I}_4$-$\mathcal{I}_7$	1e4, 9.3e9, … , 1e10
Iter. 5	9.3e9	$\mathcal{I}_3$-$\mathcal{I}_6$, $\mathcal{I}_8$-$\mathcal{I}_9$	$\mathcal{I}_5$-$\mathcal{I}_6$, $\mathcal{I}_8$-$\mathcal{I}_9$	1e4, 3e9, … , 1e10
Iter. 6	3e9	$\mathcal{I}_3$-$\mathcal{I}_4$, $\mathcal{I}_6$, $\mathcal{I}_8$-$\mathcal{I}_{11}$,	$\mathcal{I}_4$, $\mathcal{I}_{10}$-$\mathcal{I}_{11}$	1e4, … ,9.7e9
Iter. 7	1e4	$\mathcal{I}_3$-$\mathcal{I}_6$, $\mathcal{I}_8$-$\mathcal{I}_9$, $\mathcal{I}_{11}$-$\mathcal{I}_{12}$	$\mathcal{I}_4$, $\mathcal{I}_{12}$	1e4, 6.5e6, … ,1e10
Iter. 8	6.5e6	$\mathcal{I}_3$, $\mathcal{I}_6$, $\mathcal{I}_8$-$\mathcal{I}_{14}$	all	1e4, 1.7e5, …

TABLE 8 Runtime for the splitter

FOM with $n = 5019$, runtime 30,775s over 1000 frequency samples.			
Method	**ROM runtime**	**Offline time**	**Speed-up**
H-greedy, N=50	11.7s	2283s	17
H-greedy, N=100	10s	4490s	8.5
SSI-greedy, $N = 5$	11s	8256s	7.9
SSI-greedy, $N = 10$	11s	9839s	5
FOM with $n = 10,626$, runtime 257,670s over 1000 frequency samples.			
Method	**ROM runtime**	**Offline time**	**Speed-up**
H-greedy, N=50	16.2s	18,275s	14.1
H-greedy, N=100	16.5s	37,187s	6.9
SSI-greedy, $N = 5$	26.8s	97,635s	2.6
SSI-greedy, $N = 10$	20s	121,810s	2.1

better speed-ups than SSI-greedy. However, the ROMs with the best speed-ups by H-greedy are not reliable as shown in Tables 5 and 6: the ROM errors are still much larger than the error tolerance.

We compare the accuracy of the transfer functions computed from the ROMs with the original transfer function in Figure 5. Since the two models are multi-input and multi-output, the transfer functions are matrices. We only compare the transfer function from input port 1 to output port 1, and we have similar observations on the transfer functions associated with other input-outputs. The magnitudes of the transfer functions are plotted. It is seen that all ROMs there produce accurate transfer functions for both models, though the $\mathcal{L}_\infty$-error of the ROM obtained by H-greedy with $N = 100$ is still larger than the error tolerance (see the validated error in Table 6). This is because the $\mathcal{L}_\infty$-error is neither the error of the magnitude nor the error of the phase of the reduced transfer function. However, it contributes to the error bound for the output error in Equation (9), when (varying) inputs are applied to the system and can be an indicator for choosing the expansion points.

Example 3 (A multilayer irregular power bus model). The geometrical details of the model are described in Figure 6. The thicknesses of conductors and dielectrics are $t = 50$ μm and $h = 700$ μm, respectively. The relative permittivity of the dielectric is $\varepsilon_r = 4.1$ and the conductivity of conductors is $\sigma = 5.8 \times 10^7 \frac{S}{m}$. There are six input ports and six output ports. All the ports are terminated with 10Ω resistors. The delayed PEEC method[18] has been used to describe the electromagnetic problem which results in a DDE model with $n = 13,048$ unknowns and 118 different delays. The frequency range of the model is $[0, \omega_{\max}] = [0, 4\pi \times 10^9]$.

Table 9 presents the computational results of the proposed algorithms. To avoid redundancy, we do not show the selected intervals and local maximizers of the SSI-greedy algorithm, as those in Tables 3 and 7. SSI-greedy still takes much longer time than H-greedy to converge. However, it achieves better accuracy than H-greedy as can be seen from the

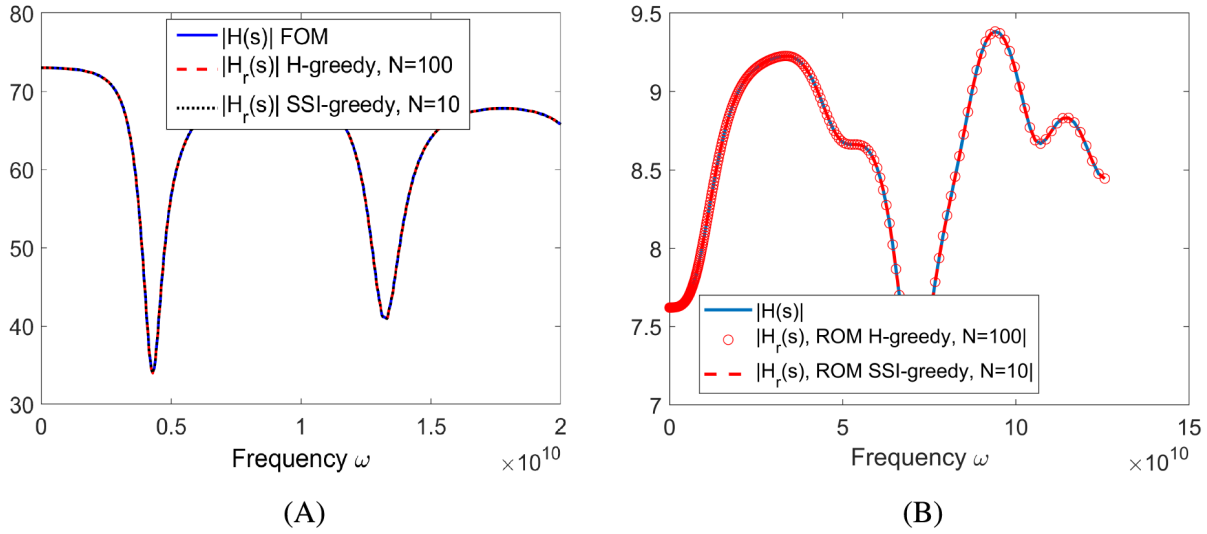


FIGURE 5 Comparison of the transfer function from port 1 to port 1. Left: dipole antennas, right: splitter with $n = 10,626$

TABLE 9 Comparison of the proposed algorithms for the power bus model with $n = 13,048$

Method name	Runtime	r	Validated error	FOM eval.	ROM eval.
H-greedy, $N = 50$	36,605s	96	8.7e-2	66	400
H-greedy, $N = 100$	77,511s	132	4.5e-03	122	1100
SSI-greedy, $N = 5$	167,300s	156	1e-03	464	1314
SSI-greedy, $N = 10$	232,110s	156	2.3-03	560	2937

TABLE 10 Runtime for the power bus model

FOM with $n = 13,048$, runtime 487,590s over 1000 frequency samples.			
Method	**ROM runtime**	**Offline time**	**Speed-up**
H-greedy, N=50	41.7s	36,563s	13.3
H-greedy, N=100	78.9s	77,432s	6.2
SSI-greedy, $N = 5$	108s	167,202s	2.9
SSI-greedy, $N = 10$	109s	232,001s	2.1

validated errors, though they are all larger than the error tolerance. It is worth pointing out that SSI-greedy with $N = 5$ gets even better accuracy than it with $N = 10$. This implicates that more robust and efficient optimization solvers (Step 12) used in Algorithm 4 should be proposed in the future.

In Figure 7, we plot the transfer function of the original model and those of the ROMs computed with H-greedy and SSI-greedy, respectively. It is shown that both reduced transfer functions match the original transfer function very well, though the desired error tolerance is not met everywhere.

In Table 10, we compare the runtime spent on computing the transfer function $H(s)$ of the FOM with the runtime of the proposed algorithms. Computing $H(s)$ needs almost 6 days, while computing $H_r(s)$ at the same 1000 frequency points is done within a few seconds. Taking the offline time into consideration, we have achieved speed-up factors between 2 and 14. Finally, it should be pointed out that for all the examples, the offline time is done only once. If further simulations need to be implemented, for example, by varying the inputs, the ROM can be further used as surrogate without being reconstructed, and only online ROM simulations are implemented.

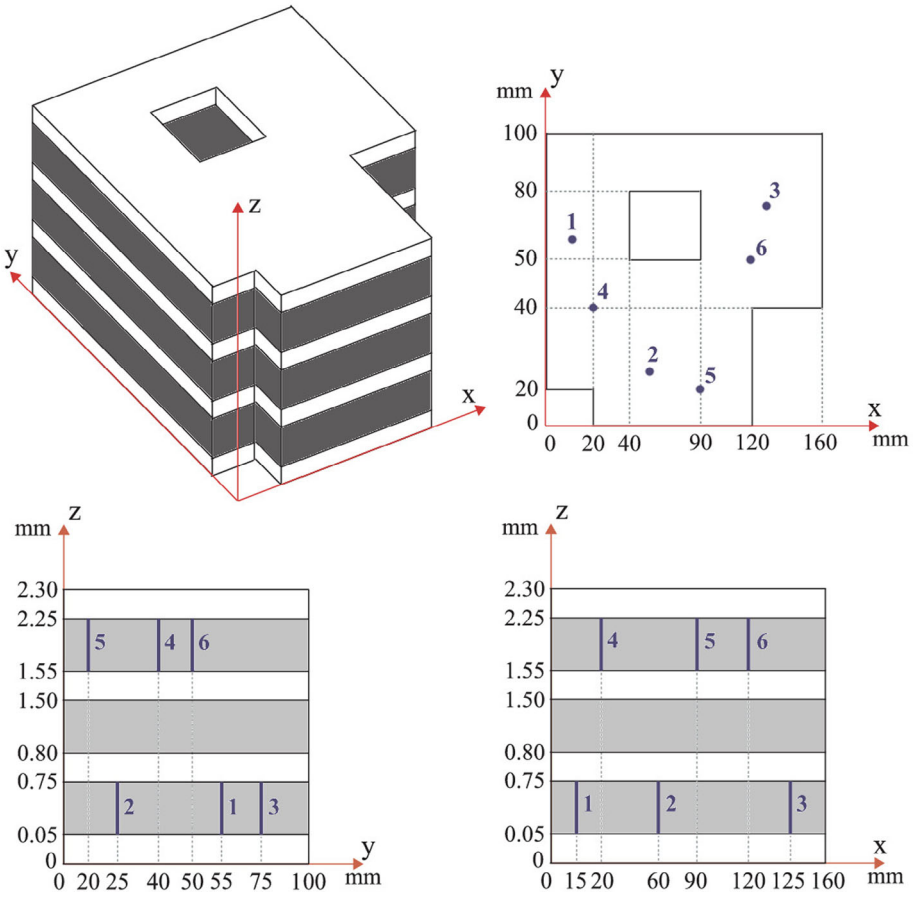


FIGURE 6 The multilayer irregular power bus

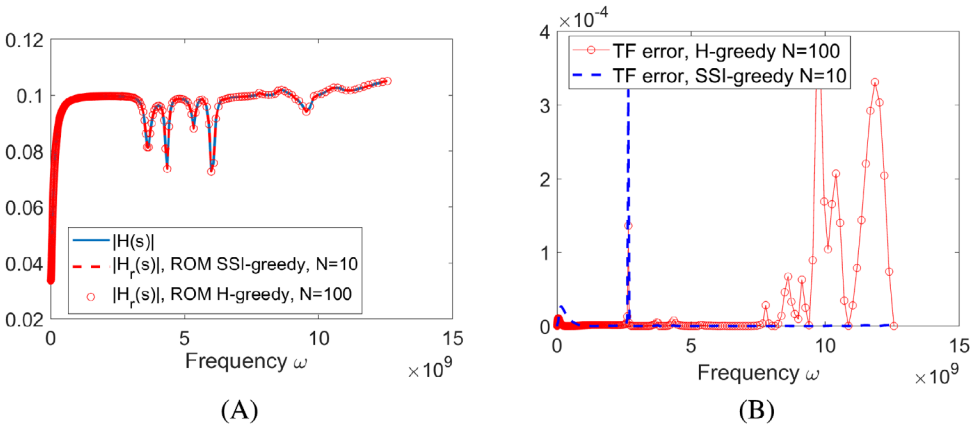


FIGURE 7 Comparison of the transfer function for the power bus model from port 1 to port 1. Left: Magnitudes of the transfer functions. Right: Magnitude of the error functions

6 | CONCLUSIONS

Simulating large-scale systems with many delays is a tough task. In this work, greedy interpolation algorithms for reduced order modeling of time delay systems with many delays are proposed. The interpolation points are selected iteratively according to the $\mathcal{L}_\infty$-error. Several techniques are proposed for computing the $\mathcal{L}_\infty$-error. While the heuristic algorithm H-greedy is faster than the other algorithms based on optimization, the SSI-greedy algorithm is more accurate. In general, the proposed algorithms H-greedy and SSI-greedy have gained noteworthy accelerations as compared to full simulations. As future work, on the one hand, more practical and fast-to-compute error estimators will be developed in order to replace the $\mathcal{L}_\infty$-error bound to improve the efficiency of the greedy algorithms. On the other hand, the efficiency of Algorithm 4 for solving large-scale systems should be further improved to make the SSI-greedy algorithm more attractive in practical applications.

ACKNOWLEDGEMENT

This work was supported by DFG Grant AN-693/1-1. Open access funding enabled and organized by Projekt DEAL.

ORCID

Lihong Feng https://orcid.org/0000-0002-1885-3269
Peter Benner https://orcid.org/0000-0003-3362-4103

REFERENCES

1. Achar R, Nakhla M. Simulation of high-speed interconnects. *Proc IEEE*. 2001;89(5):693-728.
2. Tan SX-D, He L. *Advanced Model Order Reduction Techniques in VLSI Design*. Cambridge, MA: Cambridge University Press; 2007.
3. Odabasioglu A, Celik M, Pileggi LT. PRIMA: passive reduced-order interconnect macromodeling algorithm. *IEEE Trans Comput-Aid Des Integrat Circ Syst*. 1998;17(8):645-654.
4. Feldmann P, Freund RW. Efficient linear circuit analysis by Padé approximation via the Lanczos process. *IEEE Trans Comput-Aid Des Integrat Circ Syst*. 1995;14:639-649.
5. Liu P, Li Z-F, Han G-B. Application of asymptotic waveform evaluation to eigenmode expansion method for analysis of simultaneous switching noise in printed circuit boards (PCBs). *IEEE Trans Electromagn Compat*. 2006;48(3):485-492.
6. F. Ferranti, G. Antonini, G., T. Dhaene, and L. Knockaert. Guaranteed passive parameterized model order reduction of the partial element equivalent circuit (PEEC) method. *IEEE Trans Electromagn Compat*. 52(4): 974–984, 2010.
7. Tseng W, Chen C, Gad E, Nakhla M, Achar R. Passive order reduction for RLC circuits with delay elements. *IEEE Trans Adv Packag*. 2007;30(4):830-840.
8. Ferranti F, Nakhla M, Antonini G, Dhaene T, Knockaert L, Ruehli A. Multipoint full-wave model order reduction for delayed PEEC models with large delays. *IEEE Trans Electromagn Compat*. 2011;53(4):959-967.
9. Samuel ER, Knockaert L, Dhaene T. Model order reduction of time-delay systems using a Laguerre expansion technique. *IEEE Trans Comput-Aid Design Integrat Circ Syst*. 2014;61(6):1815-1823.
10. Feng L, Korvink J, Benner P. A fully adaptive scheme for model order reduction based on moment matching. *IEEE Trans Compon Packag Manuf Technol*. 2015;5(12):1872-1884.
11. Tsuyoshi K, Sanjay G. *Moment Matching Theorems for Dimension Reduction of Higher-Order Dynamical Systems Via Higher-Order Krylov Subspaces Report No UCB/SEMM-2008/04*. Berkeley, CA: University of California; 2008.
12. Gugercin S, Antoulas AC, Beattie CA. H2 model reduction for large-scale linear dynamical systems. *SIAM J Matrix Anal Appl*. 2008;30(2):609-638.
13. Sinani K, Gugercin S, Beattie C. A structure-preserving model reduction algorithm for dynamical systems with nonlinear frequency dependence. *IFAC-PapersOnline*. 2016;49(9):56-61.
14. Beattie CA, Gugercin S. Interpolatory projection methods for structure-preserving model reduction. *Syst Control Lett*. 2009;58:225-232.
15. Aliyev N, Benner P, Mengi E, Voigt M. Large-scale computation of $\mathscr{L}_\infty$-norms by a greedy subspace method. *SIAM J Matrix Anal Appl*. 2017;38(4):1496-1516.
16. Andrew AL, Chu K-WE, Lancaster P. Derivatives of eigenvalues and eigenvectors of matrix functions. *SIAM J Matrix Anal Appl*. 1993;14(4):903-926.
17. Mengi E, Yildirim EA, Kilic M. Numerical optimization of eigenvalues of Hermitian matrix functions. *SIAM J Matrix Anal Appl*. 2014;35(2):699-724.
18. Ruehli AE, Antonini G, Jiang L. *Circuit Oriented Electromagnetic Modeling Using the PEEC Techniques*. Hoboken, NJ: Wiley-IEEE Press; 2017.

How to cite this article: Alfke D, Feng L, Lombardi L, Antonini G, Benner P. Model order reduction for delay systems by iterative interpolation. *Int J Numer Methods Eng*. 2021;122:684–706. https://doi.org/10.1002/nme.6554