Parametric Dominant Pole Algorithm for Parametric Model Order Reduction

Maryam Saadvandi
Karl Meerbergen
Wim Desmet

Report TW 625, March 2013



KU Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Parametric Dominant Pole Algorithm for Parametric Model Order Reduction

Maryam Saadvandi
Karl Meerbergen
Wim Desmet

Report TW 625, March 2013

Department of Computer Science, KU Leuven

Abstract

Standard model order reduction techniques attempt to build reduced order models of large scale systems with similar input-output behavior over a wide range of input frequencies as the full system models. The method known as the dominant pole algorithm has previously been successfully used in combination with model order reduction techniques to approximate standard linear time-invariant dynamical systems and second order dynamical systems as well as nonlinear time-delay systems. In this paper, we show that the dominant pole algorithm can be adapted for parametric systems where these parameters usually have physical meaning. There are two approaches for finding dominant poles. These algorithms are illustrated by the second order numerical examples.

Keywords : parametric system, parametric transfer function, modal approximation, residue, parametric eigenvalue problem, parametric model reduction, dominant poles algorithm.

Parametric Dominant Pole Algorithm for Parametric Model Order Reduction

Maryam Saadvandi Karl Meerbergen Wim Desmet

March 2013

Abstract

Standard model order reduction techniques attempt to build reduced order models of large scale systems with similar input-output behavior over a wide range of input frequencies as the full system models. The method known as the dominant pole algorithm has previously been successfully used in combination with model order reduction techniques to approximate standard linear time-invariant dynamical systems and second order dynamical systems as well as nonlinear time-delay systems. In this paper, we show that the dominant pole algorithm can be adapted for parametric systems where these parameters usually have physical meaning. There are two approaches for finding dominant poles. These algorithms are illustrated by the second order numerical examples.

Keywords: parametric system, parametric transfer function, modal approximation, residue, parametric eigenvalue problem, parametric model reduction, dominant poles algorithm.

1 Introduction

Consider the single-input single-output system, formulated in the Laplace or frequency domain, as follows:

$$\begin{cases} A(s)x(s) &= bu(s) \\ y(s) &= c^*x(s) \end{cases} \tag{1}$$

where $A(s)$ is an $n \times n$ matrix with large n. Function $u(s)$ is the input and $y(s)$ the output. Vector x is called the state vector. Matrix A is a function of s and is linear, e.g., $A(s) = A_0 + sA_1$, for linear systems of ODEs, or quadratic, e.g., $A(s) = K - s^2M$ or $A(s) = K + isC - s^2M$, for the analysis of vibrations or even nonlinear as, e.g., the delay differential equation, where $A(s) = A - sM + s^{-\tau s}B$ and τ is the delay.

The evaluation of $y(s)$ is usually expensive when A is large since a linear system of large size has to be solved. The goal of Model Order Reduction (MOR) is to develop a low order model of the same form as (1) but with a much smaller matrix so that its evaluation is cheap. Of course, the model should be built so that the output of the reduced model and the original one are close. There are basically three classes of MOR methods. Moment matching or Padé type methods are very popular in circuit design and vibrations [1, 2, 10]. They match high order moments around a central point or a selection of points. Krylov methods and rational Krylov methods belong to this class of methods. Balanced truncation methods can also be seen as a class of rational Krylov method. They are usually more expensive, but build smaller models for the same error level as Krylov-Padé methods. There are various algorithms for large scale problems, based on rational Krylov spaces [1, 14, 16, 17].

Finally, modal approximation is very popular in the study of vibrations, but also for the reduction of models of power lines [18, 19, 24]. The dominant pole algorithm method approximates the solution as a sum of linear rational functions of which the poles are eigenvalues of the system. The dominant pole algorithm with subspace projection [19, 21] is related to the Jacobi-Davidson method for computing eigenvalue problems. We discuss it in more detail in Section 2. There are variants for polynomial [20] and nonlinear problems [23].

In our case, y does not only depend on s but also on parameters, which we denote by $\gamma = (\gamma_1, \ldots, \gamma_p) \in \Gamma \subset \mathbb{R}^p$, that arise in A, b or c. These parameters usually have a direct physical meaning. Such parametric studies appear in uncertainty quantification or design optimization. The classical approach is to build a reduced model for each value of $\gamma \in \Gamma$ for which the system is evaluated. This is probably highly inefficient since these systems are built independently. Parametric model order reduction, on the other hand, allows for cheaply evaluating $y(s, \gamma)$ for a wide range of values of γ. Padé type methods such as PIMTAP [11, 12, 13] and TAP [11] build reduced models whose output matches multivariate moments with the exact output function. Interpolatory reduced models [5] obtained by interpolating subspaces generated at a selection of interpolation points in the parameter space or by merging these subspaces and using them for evaluating the system output in other than the interpolation points [4]. In this paper, we will iteratively compute the k parametric dominant poles. We consider two approaches. In the first approach, we compute the parameter dependent poles one by one, i.e., all parameters are taken into account together. We will use interpolation in the parameter space to achieve this. In the second approach, the dominant eigenpairs are computed for a selection of interpolation points in the parameter space, independently from each other. As for the iterative rational Krylov algorithm [4] and Krylov-Padé methods [28], the transfer function is Hermite interpolated in the iteration points in the Laplace domain and the parameter domain.

The paper is organized as follows. In Section 2 the dominant pole algorithm for nonparametric problems is reviewed. Section 3 introduces the dominant pole algorithm adapted for the parametric case. Numerical examples are shown in Section 4 and the conclusions are given in Section 5.

2 The dominant pole algorithm

The transfer function of system (1), $H : \mathbb{C} \to \mathbb{C}$, is defined as

$$H(s) = c^* A(s)^{-1} b.$$

The poles of the system (1) are the poles of $H(s)$. These form a subset of the eigenvalues $\lambda \in \mathbb{C}$. Define an eigentriplet (λ_j, x_j, y_j) of $A(s)$ as:

$$\begin{cases} A(\lambda_j)x_j & = 0 \qquad x_j \neq 0 \\ A(\lambda_j)^* y_j & = 0 \qquad y_j \neq 0 \end{cases} \tag{2}$$

where λ_j is an eigenvalue and $x_j, y_j \in \mathbb{C}$ are corresponding right and left eigenvectors. Note that the number of eigentriplets depends on the type of problem: when $A(s) = K - s^2 M$ with M symmetric positive definite and K symmetric, then there are n linearly independent right eigenvectors with associated λ_j^2 value. When $A(s) = K + isC - s^2 M$, then there are $2n$ eigenvalues λ_j and if $A(s)$ originates from the delay differential equation then the system has infinite number of eigenvalues.

The Dominant Pole Algorithm (DPA) computes 'dominant poles' of the system. We now explain this concept and how a reduced model is built from those. We assume the transfer function $H(s)$ can be expressed as

$$H(s) = \sum_j \frac{R_j}{s - \lambda_j},$$

where the sum is taken over all eigenvalues and where

$$R_j = \frac{(c^* x_j)\left(y_j^* b\right)}{y_j^* \frac{\mathrm{d}A(\lambda_j)}{\mathrm{d}\lambda_j} x_j}$$

is called the residue [8, 9]. The weighted residue is defined as

$$\rho_j = \frac{|R_j|}{|\mathrm{Re}(\lambda_j)|}.$$

The poles are sorted following decreasing weighted residue, i.e.,

$$\rho_1 \geq \rho_2 \geq \rho_3 \geq \cdots \geq \rho_k.$$

We call $\lambda_1, \ldots, \lambda_k$ the k dominant poles of the system. An approximation of transfer function $H(s)$ is obtained from $k \ll n$ terms, as in

$$H(s) \approx \widehat{H}(s) = \sum_{j=1}^{k} \frac{R_j}{s - \lambda_j},$$

The reduced model is constructed by using the matrices $X, Y \in \mathbb{R}^{n \times k}$ whose columns span the k right and left dominant eigenvectors, respectively. The reduced model is defined as

$$\begin{cases} \widehat{A}(s)\widehat{x}(s) & = & \widehat{b}u(s) \\ \widehat{y}(s) & = & \widehat{c}^* \widehat{x}(s), \end{cases} \tag{3}$$

where $\widehat{A}(s) = Y^* A(s) X \in \mathbb{R}^{k \times k}$ and $\widehat{b} = Y^* b$, $\widehat{c} = X^* c \in \mathbb{R}^k$ where $k \ll n$ with $y(s) \approx \widehat{y}(s)$. The transfer function of the reduced system (3) becomes

$$\widehat{H}(s) = \widehat{c}^* \widehat{A}(s)^{-1} \widehat{b}.$$

It has the same k dominant poles as the original function. It is therefore expected that $\widehat{H}(s)$ is a good approximation of $H(s)$ near the 'peaks' of the transfer function.

The dominant poles are eigenvalues of the eigenvalue problem (2). There are many methods for solving such eigenvalue problems. In this particular case, we are only interested in the eigenvalues with large weighted residue. The DPA is designed for finding these eigenvalues. The poles of the transfer function are, of course, the roots of $1/H(s) = 0$. The DPA uses the Newton method for finding these roots [19, 20]. As for the Jacobi-Davidson eigenvalue solver, which can be seen as Newton's method with subspace projection, we equip the DPA with subspace projection [19]. This makes the method more reliable and also allows us to compute more than one dominant pole. In order to avoid that the algorithm converges to the same dominant pole several times, deflation or locking can be used, see [19, 22] for first order, see [15, 20] for second order systems and see [23] for nonlinear systems.

The Subspace Accelerated Dominant Pole Algorithm (SADPA) is presented in Algorithm 1.

Algorithm 1 (SADPA)
INPUT: System matrix A, initial value s
OUTPUT: Dominant poles Λ, corresponding dominant right and left eigenvectors X and Y

1: Let $V = [\,]$ and $W = [\,]$.
2: **while** not all desired poles are computed **do**
3: Solve v from $A(s)v = b$.
4: Solve w from $A(s)^*w = c$.
5: Orthonormalize v against the columns of V and add to V.
6: Orthonormalize w against the columns of W and add to W.
7: Define the projected system matrix $\widetilde{A} = W^*AV$.
8: Solve the small scale eigenvalue problem $\widetilde{A}(\widetilde{\lambda}_j)z_j = 0$ and $t_j^*\widetilde{A}(\widetilde{\lambda}_j) = 0$ with eigenpairs sorted following decreasing weighted residue.
9: Compute the right and left Ritz vectors $\widetilde{x}_j = Vz_j$ and $\widetilde{y}_j = Wt_j$.
10: **if** $\|A(\widetilde{\lambda}_1)\widetilde{x}_1\|$ and $\|\widetilde{y}_1^*A(\widetilde{\lambda}_1)\|$ are small enough **then**
11: $\Lambda = [\Lambda\ \widetilde{\lambda}_1]$, $X = [X\ \widetilde{x}_1]$ and $Y = [Y\ \widetilde{y}_1]$.
12: `Deflate` $\widetilde{\lambda}_1$
13: **end if**
14: Select s among the Ritz values
15: **end while**

In Step 5, v is orthonormalized against the columns of V and then added to V. We do a similar operation for w in the next step. In Step 7, system (1) is projected using V and W. In Step 8, we compute the Ritz values, i.e., the eigenvalues of projected system $\widetilde{A}(s)$. The number of columns of V and W is small, otherwise the storage cost becomes quite high. As a result, we use QR type algorithms for solving the projected eigenvalue problem. If the number of columns of V and W would become high, it is possible to use thick restarting as for the Jacobi-Davidson method, see, e.g., [25]. The associated right and left Ritz vectors are $\widetilde{x}_j = Vz_j$ and $\widetilde{y}_j = Wt_j$ respectively. The residue can be computed as

$$\widetilde{R}_j = \frac{(c^*\widetilde{x}_j)(\widetilde{y}_j^*b)}{\widetilde{y}_j^*A'(\widetilde{\lambda}_j)\widetilde{x}_j} = \frac{(\widetilde{c}^*z_j)(t_j^*\widetilde{b})}{t_j^*\widetilde{A}'(\widetilde{\lambda}_j)z_j}$$

where $\widetilde{c} = V^*c$ and $\widetilde{b} = W^*b$. The $\widetilde{R}_j$ are thus cheaply computed. The Ritz triples are sorted following decreasing weighted residue. In Step 10, we check whether the most dominant Ritz value is close enough to an eigenvalue. If this is the case, the eigentriple is accepted as dominant pole and X and Y are expanded. Note that deflation is required to prevent the dominant pole from being recomputed. For the technical details we refer to the literature, as discussed above. The next value of s selected in Step 14 is the most dominant Ritz value that has not been deflated. In this way, we expect that the first k dominant poles are selected one by one.

3 The parametric dominant pole algorithm

We now want to build a reduced model based on modal truncation for the parametric system

$$\left\{ \begin{array}{rcl} A(s,\gamma)x(s) & = & b(\gamma)u(s) \\ y\,(s) & = & c(\gamma)^* x\,(s)\,, \end{array} \right. \tag{4}$$

where the system matrix, A, input and output vectors, b and c, depend on parameters $\gamma \in \Gamma \subset \mathbb{R}^p$. The transfer function in the Laplace domain is

$$H\,(s,\gamma) = c(\gamma)^* A(s,\gamma)^{-1} b(\gamma) = \sum_{j=1} \frac{R_j(\gamma)}{s - \lambda_j(\gamma)},$$

where the poles are similarly eigenvalues of

$$\begin{array}{rcll} A\,(\lambda_j(\gamma),\gamma)\,x_j(\gamma) = 0 & , & x_j(\gamma) \neq 0, \\ y_j(\gamma)^* A\,(\lambda_j(\gamma),\gamma) = 0 & , & y_j(\gamma) \neq 0. \end{array} \tag{5}$$

The difference with Section 2 is that all variables now depend on the parameters. From eigenvalue perturbation theory, e.g., [27], we know that the eigenvalues are continuous functions of the parameters. The eigenvectors are continuous too, as long as the eigenvalues are not defective. As a result, the eigenvalues are surfaces in $\mathbb{C} \times \Gamma$.

3.1 Parametric systems

When applying DPA to (4), we solve on each iteration the two linear systems

$$\left\{ \begin{array}{rcl} A\,(s(\gamma),\gamma)\,v(\gamma) & = & b(\gamma) \\ A\,(s(\gamma),\gamma)^*\,w(\gamma) & = & c(\gamma), \end{array} \right. \tag{6}$$

and keep $v(\gamma)$ and $w(\gamma)$ in subspaces V and W. Here $s(\gamma)$ corresponds to the γ dependent Ritz value that was selected in Step 14 of Algorithm 1. Since γ is a continuous variable, we have to approximate v and w in some way. There are various possibilities for this. The approximation will have to be developed such that only a few columns need to be added to V and W, but so that v and w are well approximated for all $\gamma \in \Gamma$.

To achieve this, we will only compute v and w for a small number of γ's and use those as an 'interpolatory' reduced model for the other values of γ. We focus on the computation of v only, since the computation of w is similar. The choice of interpolation points is by itself a hard issue. See, e.g., sparse grids [6] or lattice rules [7] to name a few possibilities. More advanced methods determine the interpolation point dynamically, e.g., the reduced basis method [3]. We will show in Section 4 which rules we used for our numerical examples. Assume we have m interpolation points in Γ. Assume that the dominant pole $\lambda_1(\gamma)$ is computed for all interpolation points $\gamma^{(j)}$, $j = 1, \ldots, m$. Let X and Y collect the right and left eigenvectors for all m parameter values. Then, the dominant poles of (4), for $\gamma \in \Gamma$, are Ritz values computed from the reduced problem

$$\widehat{A}(s,\gamma)\widehat{x}_1(\gamma) = \widehat{\lambda}_1(\gamma)\widehat{x}_1(\gamma).$$

For values of γ, different from the interpolation points, $\widehat{\lambda}$ is an approximation to the exact dominant eigenvalue. The quality of this approximation depends on the number of interpolation points but also on the problem. When the eigenvector x_1 is a smooth function of γ, a few interpolation points may be sufficient to well approximate λ_1 in the entire domain Γ.

Theorem 1 *Let A be differentiable towards γ for $\gamma \in \Gamma$ and $s \in \mathbb{C}$. Let $(\lambda_1^{(j)}, x_1^{(j)}, y_1^{(j)})$ be the dominant Ritz triplet for $\gamma = \gamma^{(j)}$ and let this triple be simple. Let $X = [x_1^{(1)}, \ldots, x_1^{(m)}]$ and $Y = [y_1^{(1)}, \ldots, y_1^{(m)}]$. Then there is a Ritz value $\widehat{\lambda}_1$, computed from $\widehat{A} = Y^* A X$, so that*

$$\widehat{\lambda}_1(\gamma^{(j)}) = \lambda_1(\gamma^{(j)}).$$

If, in addition, λ_1 is a simple eigenvalue, then

$$\nabla_\gamma \widehat{\lambda}_1(\gamma^{(j)}) = \nabla_\gamma \lambda_1(\gamma^{(j)}).$$

Proof. From the definitions of X, Y and $\widehat{A}$, it is easy to see that $\widehat{\lambda}_1(\gamma^{(j)}) = \lambda_1(\gamma^{(j)})$, $j = 1, \ldots, m$, i.e. $\widehat{\lambda}_1$ interpolates λ_1. The gradient of λ_1 towards γ is a vector with the components $\partial\lambda_1/\partial\gamma_i$, $i = 1, \ldots, p$. Since λ_1 is simple and A is differentiable in γ, the gradient exists. The derivative of λ_1 can be computed as follows. From $A(\lambda_1)x_1 = 0$ and $y_1^* A(\lambda_1) = 0$, we derive

$$\begin{aligned} \frac{\partial A}{\partial \gamma_i}x_1 + \frac{\partial A}{\partial \lambda}\frac{\partial \lambda_1}{\partial \gamma_i}x_1 + A\frac{\partial x_1}{\partial \gamma_i} &= 0 \\ y_1^*\frac{\partial A}{\partial \gamma_i}x_1 + y_1^*\frac{\partial A}{\partial \lambda}\frac{\partial \lambda_1}{\partial \gamma_i}x_1 &= 0 \\ \frac{\partial \lambda_1}{\partial \gamma_i} &= -\frac{y_1^*\frac{\partial A}{\partial \gamma_i}x_1}{y_1^*\frac{\partial A}{\partial \lambda}x_1}\ . \end{aligned}$$

Since λ_1 is a simple eigenvalue, this derivative is unique. Similarly, with $\widehat{A}(\widehat{\lambda}_1, \gamma)z_1 = 0$ and $t_1^*\widehat{A}(\widehat{\lambda}_1, \gamma) = 0$ note that

$$\frac{\partial \widehat{\lambda}_1}{\partial \gamma_i} = -\frac{t_1^*\frac{\partial \widehat{A}}{\partial \gamma_i}z_1}{t_1^*\frac{\partial \widehat{A}}{\partial \lambda}z_1}\ .$$

For $\gamma = \gamma^{(j)}$, we have that $x_1(\gamma^{(j)}) = Vz_1(\gamma^{(j)})$ and $y_1(\gamma^{(j)}) = Wt_1(\gamma^{(j)})$. Therefore as a result we have that

$$\begin{aligned} t_1^*\tfrac{\partial \widehat{A}}{\partial \gamma_i}z_1 = \quad t_1^*\tfrac{W^*\partial AV}{\partial \gamma_i}z_1 = \quad y_1^*\tfrac{\partial A}{\partial \gamma_i}x_1, \\ t_1^*\tfrac{\partial \widehat{A}}{\partial \lambda}z_1 = \quad t_1^*\tfrac{W^*\partial AV}{\partial \lambda}z_1 = \quad y_1^*\tfrac{\partial A}{\partial \lambda}x_1. \end{aligned}$$

So

$$\frac{\partial \widehat{\lambda}_1}{\partial \gamma_i} = \frac{\partial \lambda_1}{\partial \gamma_i}.$$

This ends the proof of the theorem. □

Note that the derivatives are an almost free result, regardless of the number of parameters. The eigenvectors do not have a similar property. This property does, of course, not only hold for the first dominant poles, and can be generalized to all eigenvalues, provided the gradient exists.

For Krylov and rational Krylov methods, the interpolated reduced models approximate the transfer function by Hermite interpolation in points in the $\mathbb{C} \times \Gamma$ space. We have a similar property when we interpolate in the computed dominant poles. The following theorem is shown for the case that (5) is linear.

Theorem 2 *Let the columns of X and Y span the vectors $x \neq 0$ and $y \neq 0$ respectively, where (λ, x, y) is an eigentriple of (5) for a given γ. Assume that (5) is linear and the eigenvalue λ is simple and that $y^*x \neq 0$ and $y^*(\partial A/\partial s)x \neq 0$. Assume that A, b, c are differentiable in γ and the derivatives are finite. Then*

$$\lim_{s\to\lambda} |s-\lambda|^2 \frac{\partial H}{\partial \gamma} = \lim_{s\to\lambda} |s-\lambda|^2 \frac{\partial \widehat{H}}{\partial \gamma}\ .$$

Proof. Since A is linear in s, we can write: $A = A_0 + (s-\lambda)A_1$. We have that $y^*x \neq 0$ and $y^*A_1x \neq 0$. We first prove that

$$\lim_{s\to\lambda}(s-\lambda)A^{-1}b = \lim_{s\to\lambda}(s-\lambda)X\widehat{A}^{-1}\widehat{b}\ . \tag{7}$$

Define

$$g(s) = \begin{cases} (s-\lambda)A^{-1}b & , \quad s \neq \lambda \\ \eta x & , \quad s = \lambda \end{cases}$$

where $\eta = y^*b/(y^*A_1x)$. We can similarly define

$$\widehat{g}(s) = \begin{cases} (s-\lambda)X\widehat{A}^{-1}\widehat{b} & , \quad s \neq \lambda \\ \eta x & , \quad s = \lambda. \end{cases}$$

We now have to prove that these functions are continuous in $s = \lambda$. We give the proof for g.

Now, let $g(s) = \zeta x + v$ with ζ such that $y^*v = 0$. We will prove that ζ tends to η and v to zero. Then,

$$(A_0 + (s-\lambda)A_1)(\zeta x + v) = (s-\lambda)b \ .$$

We now reorganize the terms so that the right hand side is orthogonal to y^*

$$A_0 v + (s-\lambda)A_1((\zeta - \eta)x + v) = (s-\lambda)(b - \eta A_1 x) \tag{8}$$

Hence the left hand side is also orthogonal to y. This implies

$$y^*A_0 v + (s-\lambda)y^*A_1((\zeta-\eta)x + v) = 0$$

or, since $y^*A_0 = 0$:

$$y^*A_1((\zeta-\eta)x + v) = 0 \quad , i.e., \quad |\eta - \zeta| \leq \frac{\|A_1^* y\|\|v\|}{|y^*A_1x|} \ .$$

Then, from (8) we derive that

$$\left(I - \frac{A_1 x y^*}{y^*A_1 x}\right)(A_0 + (s-\lambda)A_1)\left(I - \frac{xy^*}{y^*x}\right)v = (s-\lambda)(b - \eta A_1 x) \ . \tag{9}$$

The matrix in (9) is nonsingular in the range of $(I - xy^*/y^*x)$, therefore, v is unique and $\|v\|$ is bounded and proportional to $|s-\lambda|$, which tends to zero. This proves (7).

We can similarly prove that

$$\lim_{s\to\lambda}(s-\lambda)c^*A^{-1} = \lim_{s\to\lambda}(s-\lambda)\widehat{c}^*\widehat{A}^{-1}Y^* \ .$$

From $H = c^*A^{-1}b$, we have

$$\frac{\partial H}{\partial \gamma_i} = (A^{-1}c)^*\frac{\partial A}{\partial \gamma_i}(A^{-1}b) + \frac{\partial c}{\partial \gamma_i}(A^{-1}b) + (A^{-1}c)^*\frac{\partial b}{\partial \gamma_i}.$$

As $\partial b/\partial\gamma_i$ and $\partial c/\partial\gamma_i$ are bounded, we have

$$\lim_{s\to\lambda}|s-\lambda|^2\frac{\partial c}{\partial\gamma_i}(A^{-1}b) = \lim_{s\to\lambda}|s-\lambda|^2(A^{-1}c)^*\frac{\partial b}{\partial\gamma_i} = 0,$$

so, we readily derive that

$$\begin{aligned} \lim_{s\to\lambda}|s-\lambda|^2\frac{\partial H}{\partial\gamma_i} &= \lim_{s\to\lambda}|s-\lambda|(A^{-1}c)^*\left(\frac{\partial A}{\partial\gamma_i}\right)|s-\lambda|(A^{-1}b) \\ \lim_{s\to\lambda}|s-\lambda|^2\frac{\partial \widehat{H}}{\partial\gamma_i} &= \lim_{s\to\lambda}|s-\lambda|(Y\widehat{A}^{-1}\widehat{c})^*\left(\frac{\partial A}{\partial\gamma_i}\right)|s-\lambda|(X\widehat{A}^{-1}\widehat{b}) \ . \end{aligned}$$

Combining this with (7) proves the theorem. □

Note that Theorem 2 generalizes to the second order case by using a linearization as in [20].
The gradient of the transfer functions of the original and reduced models have the same behaviour near the dominant poles. We can therefore conclude that the parametric reduced model obtained from the DPA satisfies a Hermite interpolation property.

3.2 Algorithms

We consider two approaches. Both algorithms solve (6) for a small number of γ's, say, $\gamma^{(j)}$, $j = 1, \ldots, m$. Recall that the poles are continuous surfaces in the $\mathbb{C} \times \Gamma$ space. In the first approach, we attempt to exploit this fact. The DPA is applied to the parametric system and the dominant poles are found as a function of γ, one by one. An algorithm for this approach, called the Interpolatory Dominant Pole Algorithm (IDPA) is shown in the algorithm below.

Algorithm 2 (IDPA (Interpolatory DPA))
INPUT: System matrix A, initial values $s^{(1)}, \ldots, s^{(m)}$
OUTPUT: Dominant poles Λ, corresponding dominant right and left eigenvectors X and Y

Let $V = [\,]$ and $W = [\,]$.
Set of indices of converged eigenvalues $\mathcal{J} = \{1, \ldots, m\}$.
while not all desired poles are computed **do**
  for $j = 1, \ldots, m$ **do**
    if j is a member of $\mathcal{J}$ **then**
      Select $s^{(j)}$ as the most dominant undeflated Ritz value for $\gamma = \gamma^{(j)}$.
      Solve $v^{(j)}$ from $A(s^{(j)}, \gamma^{(j)})w^{(j)} = b(\gamma^{(j)})$.
      Solve $w^{(j)}$ from $A(s^{(j)}, \gamma^{(j)})^* w^{(j)} = c(\gamma^{(j)})$.
      Orthonormalize $v^{(j)}$ against the columns of V and add to V.
      Orthonormalize $w^{(j)}$ against the columns of W and add to W.
      Define the projected system matrix $\widetilde{A} = W^* A V$.
      Solve the small scale eigenvalue problem $\widetilde{A}(\widetilde{\lambda}_j) z_j = 0$
      for $i \in \mathcal{J}$ **do**
        if the dominant Ritz pair for $\gamma^{(i)}$ is accurate enough **then**
          Add the right Ritz vector to X and the left Ritz vector to Y.
          Deflate the dominant eigentriplet.
          Remove i from $\mathcal{J}$
        end if
      end for
      if $\mathcal{J}$ is empty **then**
        Initial set of indices $\mathcal{J} = \{1, \ldots, m\}$ for the next dominant pole.
      end if
    end if
  end for
end while

We want to exploit the fact that eigenvalues are continuous functions of γ. Let us assume for a moment that the surfaces do not intersect, i.e., all eigenvalues are simple or semisimple. We compute such surfaces one by one. In each iteration, the parametric eigenvalue problem is projected and Ritz triplets are computed. Following Theorem 1, the Ritz values Hermite interpolate the eigenvalues in the interpolation points. For efficiency reasons, we use the index set $\mathcal{J}$. This set of indices indicates for which interpolation points, the system has to be evaluated. By default, all interpolation points are used. When a Ritz triplet has converged for interpolation point $\gamma^{(j)}$, say, the solution at $\gamma^{(j)}$ will not change and, therefore, it does not make much sense to recompute it. When the eigenvalues for all m interpolation points are computed we move to the next surface. When there are multiple eigenvalues, the situation is more complex. Nevertheless, we still use the same strategy in this case in our numerical examples in Section 4. Deflation is needed to prevent converged Ritz triplets from being reselected as s and hence to be recomputed. The principle of deflation is that the residue of converged Ritz pairs is set to zero so that those can no longer be selected as next s. For the technical details we refer to the literature [20, 23]. In our numerical experiments we used the deflation strategy from [20].

In the second approach, we compute the dominant poles for each interpolation point $\gamma^{(j)}$ independently. Since the eigenvalues and eigenvectors are continuous functions of the parameters, it makes sense to use the solution of previous parameters to compute a starting value and include

the subspace of Ritz vectors as well. This idea is implemented in the following algorithm which is called Continuation Dominant Pole Algorithm (CDPA1).

Algorithm 3 (CDPA1 (Continuation DPA))
INPUT: System matrix A, initial value $s^{(1)}$.
OUTPUT: Dominant poles Λ, corresponding dominant right and left eigenvectors X and Y
1: Let $X = [\,]$ and $Y = [\,]$.
2: **for** $j = 1, \ldots, m$ **do**
3: Use the algorithm 1 for computing k dominant poles using X and Y as initial bases for $\gamma = \gamma^{(j)}$. Denote the k dominant poles by $\Lambda^{(j)}$ and associated right and left eigenbases by $X^{(j)}$ and $Y^{(j)}$, respectively. For $j > 1$, use $\lambda_1^{(j-1)}$ as initial value of $s^{(j)}$.
4: Merge the bases $X^{(j)}$ and $Y^{(j)}$ to X and Y respectively.
5: **end for**

The disadvantage of this algorithm is that the initial V and W become very large in Step 3, when j is large. To prevent this, we introduce a projection step right before Step 3, so that the initial V and W correspond to the Ritz bases obtained by projection on the current X and Y. This strategy is implemented in the following algorithm (CDPA2).

Algorithm 4 (CDPA2 (Continuation DPA))
INPUT: System matrix A, initial value s
OUTPUT: Dominant poles Λ, corresponding dominant right and left eigenvectors X and Y
1: Let $X = [\,]$ and $Y = [\,]$.
2: **for** $j = 1, \ldots, m$ **do**
3: Solve the projected eigenvalue problem $\widehat{A}(\widehat{\lambda}, \gamma^{(j)})x = 0$ and select k dominant Ritz triplets, if they exist.
4: Collect the k right dominant Ritz vectors in V and the k left dominant Ritz vectors in W.
5: Use the dominant pole algorithm for computing k dominant poles using V and W as initial bases for $\gamma = \gamma^{(j)}$ and $\widehat{\lambda}_1$ as initial s. Denote the k dominant poles by $\Lambda^{(j)}$ and associated right and left eigenbases by $X^{(j)}$ and $Y^{(j)}$ respectively.
6: Merge the bases $X^{(j)}$ and $Y^{(j)}$ to X and Y respectively.
7: **end for**

Note that for $j = 1$, X and Y are empty and that for $j = 2$, V and W span the same subspaces for both Algorithms 3 and 4.

4 Numerical examples

We now illustrate the algorithms for three second order parametric dynamical systems. The system matrix takes the form

$$A(s, \gamma) = s^2 M(\gamma) + sC(\gamma) + K(\gamma),$$

where $M, C, K \in \mathbb{R}^{n \times n}$ are mass matrix, damping matrix and stiffness matrix, respectively, and potentially depend on p parameters $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_p)$.

4.1 Aluminum plate

The model which we illustrate, consists of an aluminum A2 plate (420×594mm, $E = 70$GPa, $\nu = 0.3, \rho = 2700\text{kg/m}^3, t = 3$mm). A grounded torsional spring with variable stiffness was added to allow for a distinction between fixed (zero displacements and zero bending moments along the boundary) and clamped boundary conditions (zero displacements and rotations along the boundary). However, since perfect clamping does not exist, it is parametrised using boundary springs. Moreover, if the clamping frame allows for (small) movements, there can also be (a limited amount of) damping. The spring constant is $]0, \infty[$ Nm/rad (closer to clamped/fixed). The used Finite Element model is valid for the frequency range 0 to 735Hz ($[0, 2\pi 735\text{rad/s}]$),

using 10 elements per wavelength. In this frequency range, the first 15 modes for the fixed case and the first 12 modes for the clamped case are located. The model consists of 2684 elements and, since we are dealing with plate elements, 6 DOFs per node, thus 15480 DOFs for the whole plate. Of these 15480 DOFs, 606 are constrained for the fixed case and 1212 for the clamped case. This model leads to stiffness, damping and mass matrices K_0, C_0, M_0, respectively. Matrices $K_{\text{bnd}}, C_{\text{bnd}}$ are related to boundary springs and dampers, with size 14876×14876. We define the mass, damping and stiffness matrices for the whole system in the following way:

$$\begin{aligned} M &= M_0, \\ C &= C_0 + \gamma_1 C_{\text{bnd}} \\ K &= K_0 + \gamma_2 K_{\text{bnd}}, \end{aligned}$$

where $\gamma = (\gamma_1, \gamma_2)$ are two parameters of the system which can be arbitrary function of a general other parameters. The free parameters (γ_1, γ_2) are defined by

$$\begin{aligned} &\gamma_1 \in [10^{-4}, 10^{-2}], \\ &\gamma_2 = \frac{-100}{\alpha_2 - 1} - 101, \qquad \alpha_2 \in [0, 1). \end{aligned} \tag{10}$$

Input and output vectors, $b, c \in \mathbb{R}^{14876}$ are equal to zero except their 1245th components which are equal to -1.

4.2 Selection of the interpolation points

We now explain how the interpolation points can be chosen for a 2D parameter domain. The classical approach for selecting interpolation points in Γ is to discretize the γ_1 and γ_2 axes independently in, e.g., m_1 and m_2 points respectively. This would lead to a grid of $m_1 \cdot m_2$ interpolation points. The discretization is usually equidistantly chosen in a linear or logarithmic fashion. For 2D problems, such an approach is still feasible, but for more parameters, this would lead to an explosion of interpolation points. As we mentioned higher, it is usually more efficient to use other techniques, such as sparse grids or Quasi Montecarlo methods. In this paper, we use a lattice rule that is generated as follows [7]. We assume that the number of interpolation points, m, is a Fibonacci number and Z is the Fibonacci number coming just before m, i.e. we have the sequence of Fibonacci numbers

$$1, 1, 2, 3, 5, 8, \ldots, Z, m \ .$$

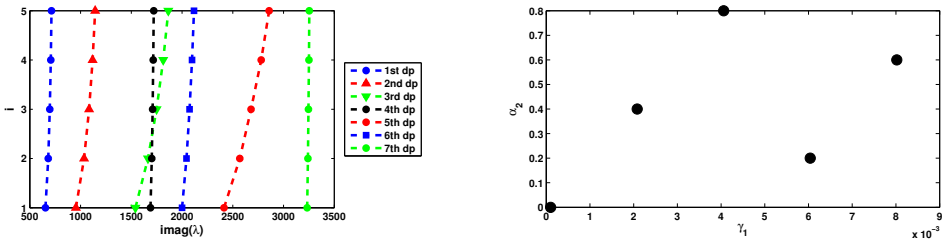
We define the m lattice points as

$$\alpha^{(j)} = \begin{bmatrix} \alpha_1^{(j)} \\ \alpha_2^{(j)} \end{bmatrix} = \frac{\operatorname{mod}\left((j-1)\begin{bmatrix} 1 \\ Z \end{bmatrix}, m\right)}{m}, \qquad j = 1, \ldots, m \tag{11}$$

where the function mod stands for the modulo as the remainder after the division with m. The points are chosen so that $\alpha \in [0, 1]^2$. The system parameter γ, defined in (10), is computed as follows from α:

$$\begin{aligned} \gamma_1 = 10^{-4} + \alpha_1(10^{-2} - 10^{-4}), \qquad & \alpha_1 \in [0, 1) \\ \gamma_2 = \frac{-100}{\alpha_2 - 1} - 101, \qquad & \alpha_2 \in [0, 1). \end{aligned}$$

Figure 1(a) shows the seven dominant poles corresponding to the five point lattice rule (11) shown in Figure 1(b). The vertical axis in Figure 1(a) corresponds to the index of $\gamma^{(i)}, i = 1, \ldots, 5$. We can see that the first pole does not change significantly. For the third and fourth poles the behaviour clearly becomes more parameter dependent.



(a) Dominant poles (b) five point lattice

Figure 1: Dominant eigenvalues in function of the parameters.

Table 1: Five interpolation points as defined by lattice rule for the first example.

	α_1	α_2	γ_1	γ_2
$\gamma^{(1)}$	0	0	0.0001	-1
$\gamma^{(2)}$	0.6	0.2	0.0060	24
$\gamma^{(3)}$	0.2	0.4	0.0021	65.67
$\gamma^{(4)}$	0.8	0.6	0.0080	149
$\gamma^{(5)}$	0.4	0.8	0.0041	399

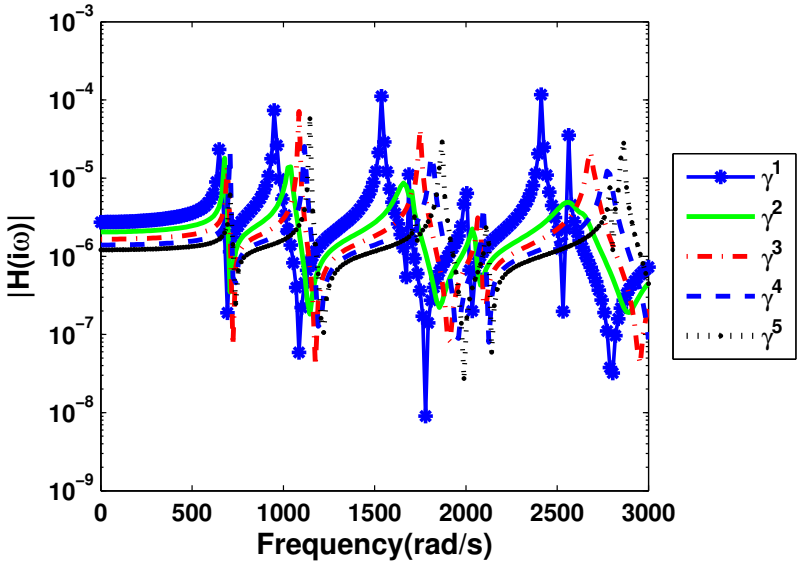


Figure 2: Exact transfer functions for the five lattice interpolation points (first example).

Table 2: Convergence history of Algorithm 2.

Iteration	pole	Iteration	pole	Iteration	pole
10	$\lambda_1^{(5)}$	17	$\lambda_2^{(2)}$	23	$\lambda_3^{(2)}$
11	$\lambda_1^{(1)}$	18	$\lambda_2^{(3)}$	27	$\lambda_3^{(1)}$
13	$\lambda_1^{(3)}$	19	$\lambda_2^{(4)}$	28	$\lambda_3^{(3)}$
14	$\lambda_1^{(4)}$	20	$\lambda_2^{(5)}$	30	$\lambda_3^{(5)}$
15	$\lambda_1^{(2)}$	21	$\lambda_2^{(1)}$	32	$\lambda_3^{(4)}$
Iteration	pole	Iteration	pole	Iteration	pole
33	$\lambda_4^{(1)}$	40	$\lambda_6^{(1)}$	47	$\lambda_5^{(2)}$
34	$\lambda_6^{(2)}$	41	$\lambda_4^{(2)}$	48	$\lambda_5^{(3)}$
36	$\lambda_3^{(4)}$	42	$\lambda_6^{(3)}$	49	$\lambda_5^{(4)}$
37	$\lambda_3^{(5)}$	44	$\lambda_7^{(5)}$	50	$\lambda_5^{(5)}$
39	$\lambda_3^{(3)}$	45	$\lambda_7^{(4)}$	51	$\lambda_5^{(1)}$

4.3 Finding dominant poles

Table 1 and Figure 1(b) show the points obtained for the five point lattice rule explained higher. It is in this order that the interpolation points are chosen in Algorithms 2 and 4. Figure 2 shows the transfer functions for these five interpolation points. It shows how the parameters change the location of the peaks (i.e., the imaginary parts of the dominant poles). The first pole does not change much. For higher frequencies, there clearly is more pronounced parameter dependent behaviour. This is also true for the weighted residue. Figure 1(a) shows the imaginary parts of the first seven dominant poles for five parameter values shown in the lattice in Figure 1(b). This illustrates that, for this problem, it makes sense to compute the eigenvalue functions of γ one by one.

All experiments reported here are executed in Matlab 7.9 on a DELL Latitude 6500 (Intel P8400 2.26 GHz, 2 GB RAM). The Algorithms 2, 3 and 4 are executed with initial value $s = 1$ and $\mathrm{TOL} = 10^{-7}$ is used for accepting the eigenvalues, see Algorithm 1. An alternative choice of initial value for Algorithms 3 and 4 will be discussed further.

When the system matrices K, C, and M are real, and also the excitation b and observation c are real, it is clear that the residues for (λ, x, y) and its complex conjugate are equal. Therefore, when the triplet (λ, x, y) is selected as a dominant pole, we also take $(\bar{\lambda}, \bar{x}, \bar{y})$ as dominant triplet. In practice, however, adding x and $\bar{x}$ to X is equivalent to adding $\mathrm{Re}(x)$ and $\mathrm{Im}(x)$ to X, since both produce the same subspace. This leads to real reduced matrices, which may be a computational and storage advantage.

For this example, we computed the ten dominant poles for the five lattice points from Table 1. The associated eigenvectors span a subspace of dimension 50. When the eigenvectors are split in their real and imaginary parts, the reduced model is real but has dimension 100. The frequency response functions are plotted in the frequency range $[0\,\mathrm{rad/s}, 3000\,\mathrm{rad/s}]$ in Figure 2. In the following, we compare the three Algorithms 2–4.

We first used Algorithm 2 and found ten dominant poles as a function of γ, one by one. Algorithm 2 needs 71 iterations for finding 10 dominant pole functions in 3.59 minutes. Recall that in each iteration, one vector is added to the subspace, so V and W have each 71 columns. We illustrate the convergence behaviour of Algorithm 2 for computing the first six dominant poles in Table 2. The table shows at which iteration an eigenvalue has converged. In the first 15 iterations, the eigenvalues nearest the origin (λ_1) are found for all interpolation points as the first dominant pole. Then, at iteration 21, the second dominant poles for all interpolation points are found. It should be noted that the first two dominant poles correspond to the two eigenvalues nearest the real axis. This is not so for the remaining dominant poles. It can also be seen in Figure 2 that the

behaviour becomes less monotonic due to the switching of the third and fourth eigenvalues. For all γ's except $\gamma^{(1)}$, the third and fourth dominant poles are very close (they cannot be distinguished in Figure 2). As can be seen from Table 2, the algorithm found the sixth dominant for $\gamma^{(2)}$ before the fourth. This illustrates that the dominant pole algorithm is not fully reliable and can miss some eigenvalues.

We now discuss Algorithm 3. First, we want to show that using good initial s indeed reduces the computation time. Let, in Step 3 of Algorithm 3, the initial value of s be chosen equal to 1 for all j instead of the more appropriate choice as described in Algorithm 3. Table 3 illustrates that

Table 3: Number of iterations and CPU time for finding the ten dominant poles for each interpolation point using different choices of s in Algorithm 3.

	$s = 1$		$s = \lambda_1^{(j-1)}$	
	# iteration	time (min)	# iteration	time (min)
$\gamma^{(1)}$	22	0.94	22	0.94
$\gamma^{(2)}$	20	1.00	18	0.92
$\gamma^{(3)}$	14	0.84	13	0.80
$\gamma^{(4)}$	16	1.14	12	0.92
$\gamma^{(5)}$	11	1.00	10	0.96
Total	83	4.92	75	4.45

the better choice indeed reduces the number of iterations slightly. For both choices, the number of iterations for $\gamma^{(1)}$ are the same, since in both cases, $s = 1$. For the other interpolation points, the better selection, from Algorithm 3, needs less iterations. As it is shown in Figure 2, the first dominant poles for each $\gamma^{(j)}$ are close to each other. So it makes sense when we find one of them for $\gamma^{(j)}$ then we can use it as an initial value for the next parameter $\gamma^{(j+1)}$.

We now compare Algorithms 3 and 4 when the better choice of initial s is used. Recall that the motivation to introduce Algorithm 4 is to reduce the growth of the initial bases V and W for larger values of j in Algorithm 3. For $\gamma^{(5)}$, e.g., the initial V has already forty columns in Algorithm 3, where it has only ten columns in Algorithm 4. Smaller spaces may lead to more iterations, since the spaces have less information, but, since they are smaller, they require less work for solving the projected problem. Table 4 shows these points: although Algorithm 4 requires more iterations

Table 4: Number of iterations and CPU time for finding the ten dominant poles for each interpolation point using Algorithms 3 and 4.

	Algorithm 3		Algorithm 4	
	# iteration	time (min)	# iteration	time (min)
$\gamma^{(1)}$	22	0.94	22	0.94
$\gamma^{(2)}$	18	0.92	18	0.92
$\gamma^{(3)}$	13	0.80	13	0.68
$\gamma^{(4)}$	12	0.92	15	0.77
$\gamma^{(5)}$	10	0.96	10	0.52
Total	75	4.45	78	3.83

than Algorithm 3, it is slightly cheaper.

Whereas the previous discussion compared the algorithms to construct the reduced models, let us now have a look at the quality of the reduced model. Note that the presented algorithms produce the same reduced model with fifty poles, i.e., ten poles for five interpolation points. Recall that the dimension of the original system is $14,876$. Figure 3 shows the absolute error for this

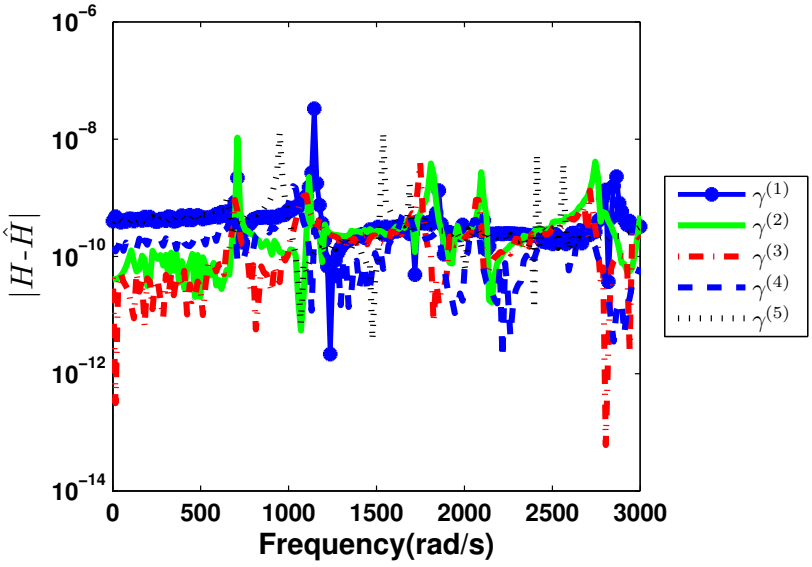


Figure 3: Absolute error for the five lattice interpolation points.

reduced model for each of the five interpolation points, shown in Table 1 and their corresponding exact transfer functions are shown in Figure 2.

We use two criteria to estimate the quality of the reduced model.

- *Global transfer function error:* The global transfer function is defined by

$$\sqrt{\sum \left(\left|H(\omega,\gamma^{(j)})\right|\right)^2} \tag{12}$$

  and the global transfer function error by

$$\sqrt{\sum \left(\left|H(\omega,\gamma^{(j)}) - \widehat{H}(\omega,\gamma^{(j)})\right|\right)^2}. \tag{13}$$

  This error is the mean error all parameter values.

- *Relative parametric error:* The relative parametric error is defined as

$$\text{error}_{\text{rel}} = \frac{\int_{\omega_{\min}}^{\omega_{\max}} |H - \widehat{H}| d\omega}{\int_{\omega_{\min}}^{\omega_{\max}} |H| d\omega} \tag{14}$$

  for any $\gamma \in \Gamma$. The integral is computed by using the trapezoid rule where r equidistant points are selected in the frequency range $[\omega_{\min}, \omega_{\max}]$.

This error shows the error for a given parameter. Figure 4 illustrates the global transfer function (12) and the global transfer function error (13) for the reduced models of all five interpolation points, shown in Table 1 and their corresponding exact global transfer functions are shown in Figure 2. The relative parametric error (14) for the five interpolation points lies between 7.7×10^{-5} and 2.6×10^{-4} where $r = 200$ equidistant points are selected in the frequency range $[0\,\text{rad/s}, 3000\,\text{rad/s}]$.

In order to evaluate the quality of the reduced model for other than the interpolation points, we have selected 100 points from the grid obtained by 10 equidistant values of $\gamma_1 \in [1\times10^{-4}, 9\times10^{-3}]$ and $\gamma_2 \in [-1, 400]$, see Figure 5.

Figure 6 shows the global transfer function of (12) and global transfer function error of (13) for these 100 γ's. The figure shows that a good approximation is obtained in the entire Γ region by

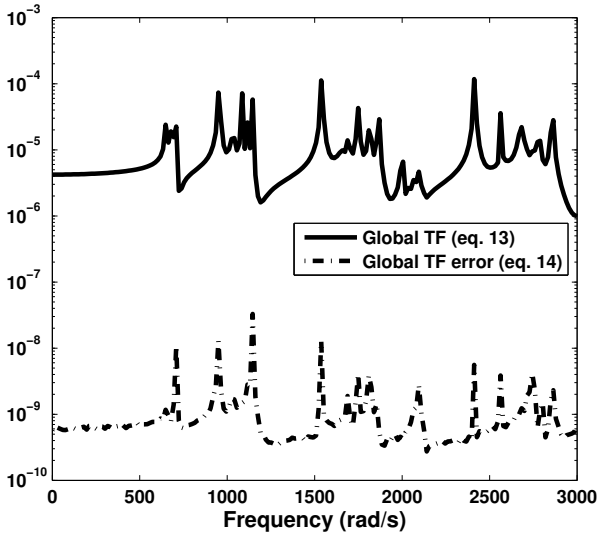


Figure 4: Global transfer function (solid line) and global transfer function error (dashed-dotted line) for five interpolation points.

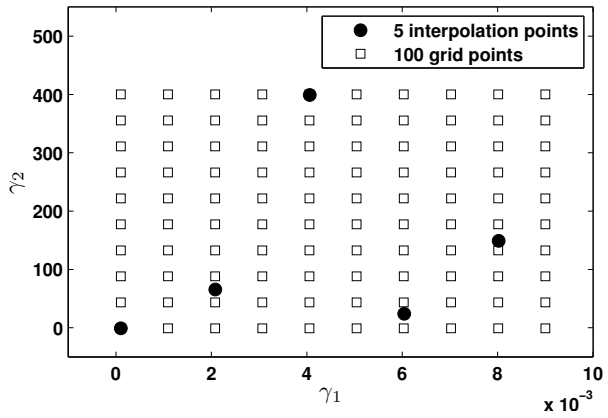


Figure 5: The location of 100 additional points, 10 equidistant points are selected for each parameters $\gamma_1 \in [1 \times 10^{-4}, 9 \times 10^{-3}]$ and $\gamma_2 \in [-1, 400]$ which are shown by squares and the location of 5 interpolation points which are shown by circles.

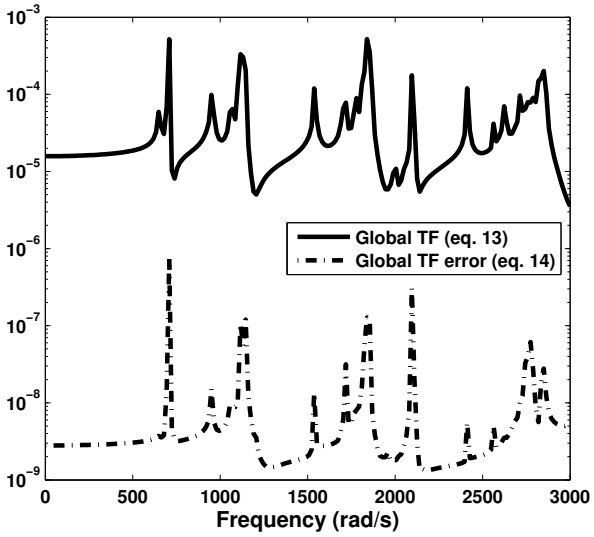


Figure 6: The global transfer function (solid line) and global transfer function error (dashed-dotted line) for each $\gamma^{(j)}$, $j = 1, \ldots, 100$ which are shown in Figure 5.

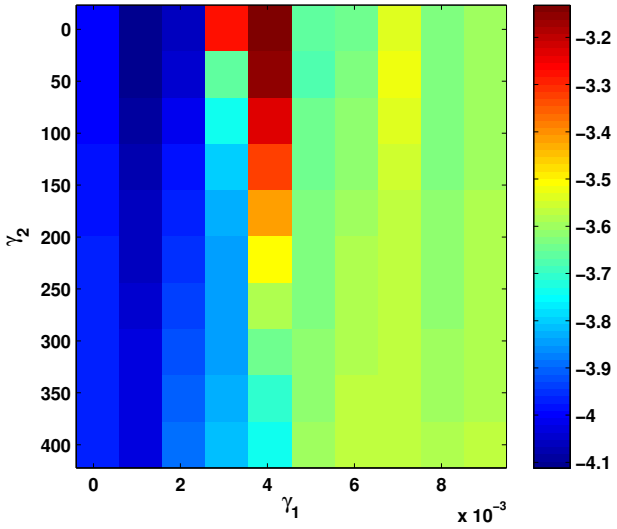


Figure 7: $\log(\text{error}_{\text{rel}})$ for 100 points are shown in Figure 5.

considering only five interpolation points. The relative parametric error($\text{error}_{\text{rel}}$) for these points (are shown in Figure 5) lies between 7.7×10^{-5} and 7.4×10^{-4}. Figure 7 illustrates the $\log(\text{error}_{\text{rel}})$ against the parameters γ_1 and γ_2.

The values we chose for γ_2 in Figure 5 only cover a small range. We compared the accuracy of the reduced model for 200 additional points with larger values of γ_2 shown in Figure 8, chosen as

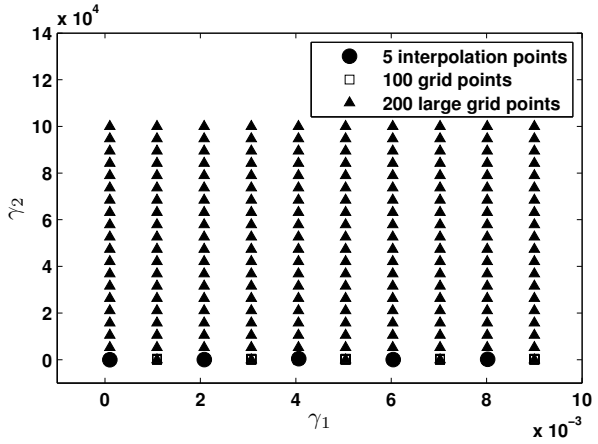


Figure 8: The location of 200 additional points, 10 equidistant values of $\gamma_1 \in [1\times10^{-4}, 9\times10^{-3}]$ and 20 equidistant values of $\gamma_2 \in [1, 1\times10^5]$ which are shown by triangles.

the grid points obtained from 10 equidistant values of $\gamma_1 \in [1\times10^{-4}, 9\times10^{-3}]$ and 20 equidistant values $\gamma_2 \in [1, 1\times10^5]$.

The global transfer function and the global absolute error for the additional 200 points is shown in Figure 9 and it shows that a good approximation is also obtained for a large range of γ_2's. The

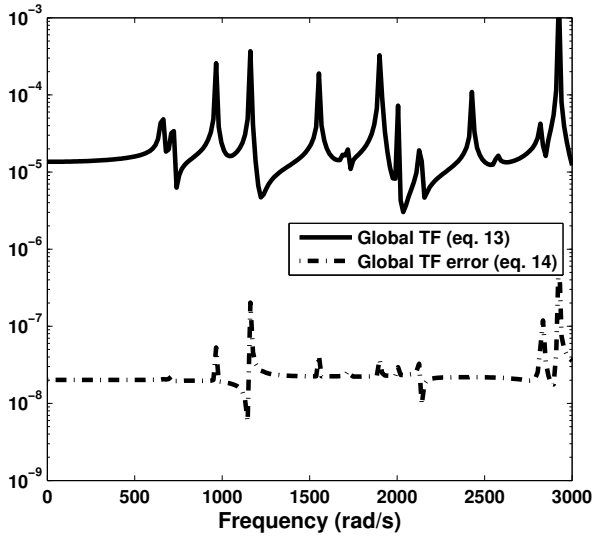


Figure 9: The global transfer function (solid line) and global transfer function error (dashed-dotted line) for each $\gamma^{(j)}$, $j = 1, \ldots, 200$ which are shown in Figure 8.

relative parametric error($\text{error}_{\text{rel}}$) for these points (shown in Figure 8) lies between 9.2×10^{-5} and 1.5×10^{-3}. Figure 10 illustrates the $\log(\text{error}_{\text{rel}})$ against the parameters γ_1 and γ_2.

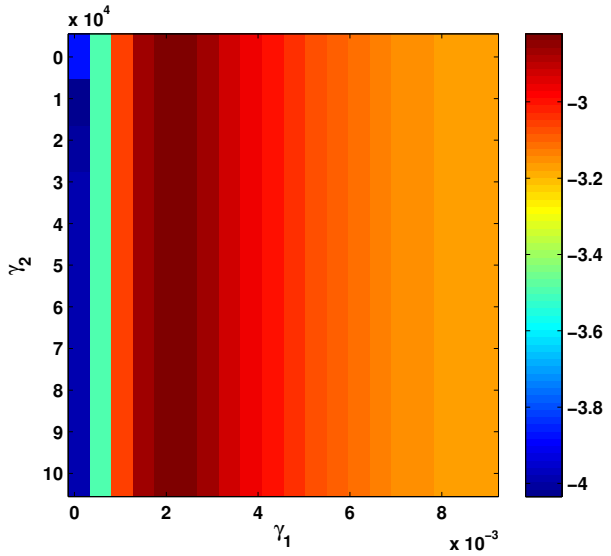


Figure 10: $\log(\text{error}_{\text{rel}})$ for 200 points are shown in Figure 8.

4.4 Modified residue for a frequency range

The standard definition of dominant pole takes the entire frequency range, i.e., from $-\infty$ to $+\infty$ into account. Usually, systems of the form (1) arise from a spatial discretization, where the frequency range is limited to the granularity of the discretization. This suggests that the selection of dominant poles should be limited to poles that make physical sense. For the first example, only frequencies up to $3000\,^{\text{rad}}/_{\text{s}}$ play an important role. In addition, for many problems of acoustics and vibrations, only a frequency range of interest should be considered for a specific model. We consider two cases.

1. Slide window $[0, \omega_{\max}]$. In the case of a low frequency range, e.g., $[0, \omega_{\max}]$, we are not looking for dominant poles λ_i with $|\texttt{imag}(\lambda_i)| \gg \omega_{\max}$ (see Figure 11). We therefore change the definition

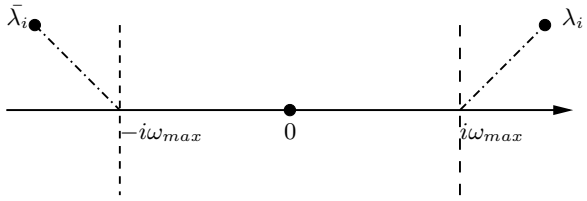


Figure 11: Required range of frequency $[0, \omega_{\max}]$.

of dominance by changing the definition of weighted residue into

$$\rho = \begin{cases} \dfrac{R(\lambda_i)}{\min\{|i\omega_{\max} - \lambda_i|, |i\omega_{\max} + \lambda_i|\}} & , \quad |\text{Im}(\lambda_i)| > \omega_{\min} \\ \dfrac{R(\lambda_i)}{|\text{Re}(\lambda_i)|} & , \quad |\text{Im}(\lambda_i)| \leq \omega_{\min} \end{cases} \tag{15}$$

As a result, when $|\texttt{imag}(\lambda_i)| \gg \omega_{\max}$ then λ_i cannot be dominant even when it lies close to the imaginary axis.

2. Slide window $[\omega_{\min}, \omega_{\max}]$. In this case, see Figure 12, we have to take into account a lower

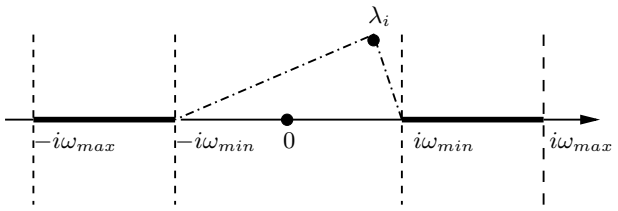


Figure 12: Required range of frequency $[\omega_{\min}, \omega_{\max}]$.

bound and an upper bound to the frequency:

$$\rho = \begin{cases} \dfrac{R(\lambda_i)}{\min\{|i\omega_{\min} - \lambda_i|, |i\omega_{\min} + \lambda_i|\}} & , \quad |\mathrm{Im}(\lambda_i)| < \omega_{\min} \\ \dfrac{R(\lambda_i)}{\min\{|i\omega_{\max} - \lambda_i|, |i\omega_{\max} + \lambda_i|\}} & , \quad |\mathrm{Im}(\lambda_i)| > \omega_{\min} \\ \dfrac{R(\lambda_i)}{|\mathrm{Re}(\lambda_i)|} & , \quad \text{otherwise.} \end{cases} \tag{16}$$

Note that changing the definition of weighted residue to (15) or (16) does not exclude that the dominant pole(s) lie outside the required frequency range.

We changed the frequency range to $[1100, 2100]$ for the same numerical example. We computed three dominant poles using the alternative definition of dominance (16) in Algorithm 2 and 4 with the same five point lattice rule and the initial value is $s_0 = 1100$. As is shown in Figure 13 all dominant poles in the correct range are found. Algorithm 2, for finding three dominant functions (15 dominant poles) in this interval, needs 29 iterations in 0.95 minutes. Table 5 shows the number

Table 5: Number of iterations and CPU times for finding the three dominant poles in frequency range $[1100, 2100]$ for five interpolation points using Algorithms 4.

	Algorithm 4	
	# iteration	time (min)
$\gamma^{(1)}$	8	0.26
$\gamma^{(2)}$	6	0.22
$\gamma^{(3)}$	3	0.11
$\gamma^{(4)}$	4	0.15
$\gamma^{(5)}$	3	0.11
Total	24	0.83

of iterations and CPU times for finding the dominant poles using Algorithm 4.

4.5 Floor damper with two design parameters

In this application, we consider the design of a tuned mass damper whose function is to alleviate the vibration of the floor inside a building located near a highway. Its conceptual model is shown in Figure 14 [28].

The floor is $10m \times 10m \times 0.3m$ in size. Its Young's modulus, Poisson's ratio and density are 30GPa, 0.3 and 2500 kg/m3, respectively. The damper can be modeled as a classical stiffness-damping-mass system with a fixed mass value $m1 = 3750$kg. Our objective is to reduce the system vibrations over the frequency range $[0\,\mathrm{rad/s}, 350\,\mathrm{rad/s}]$ by choosing the stiffness k_1 and the damping factor c_1 of the damper. In this example parameter, γ is (c_1, k_1). The size of system is $n = 29800$ and TOL $= 10^{-7}$ and parameters c_1 and k_1 belong to the interval $[1e3\,\mathrm{Ns/m}, 4e3\,\mathrm{Ns/m}]$

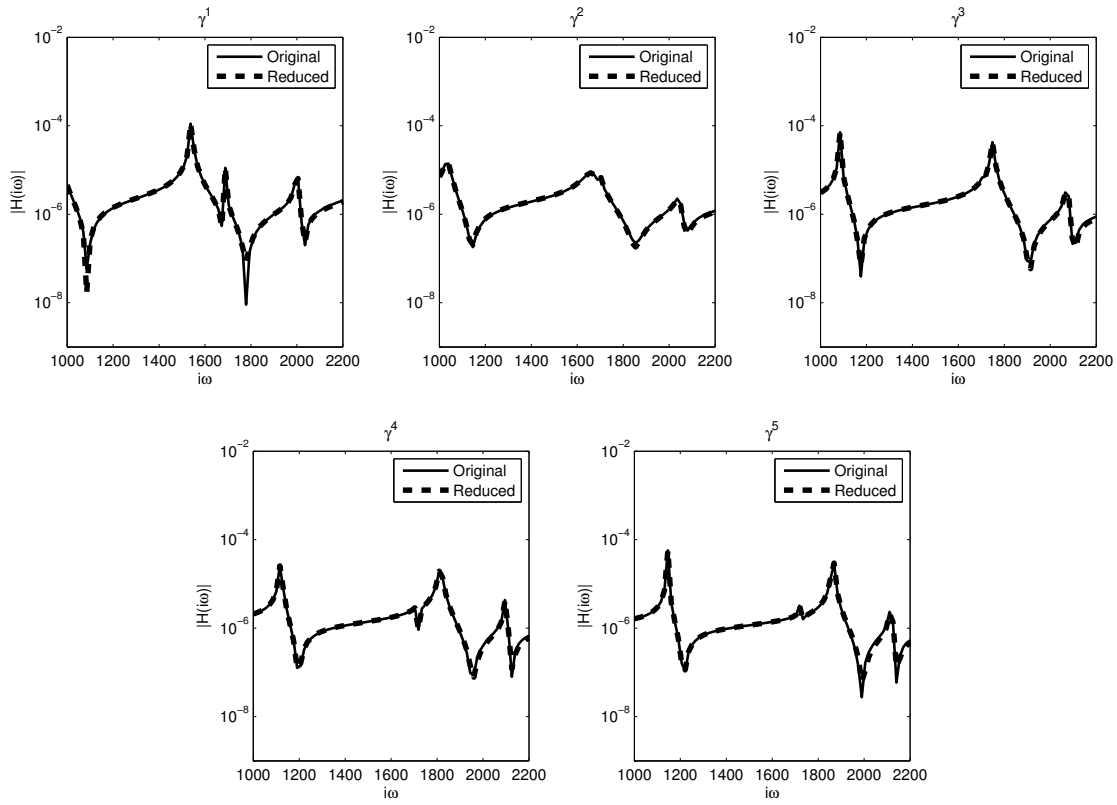


Figure 13: Transfer function of the original large scale and reduced models, dominant poles are found in slide window [1100,2100].

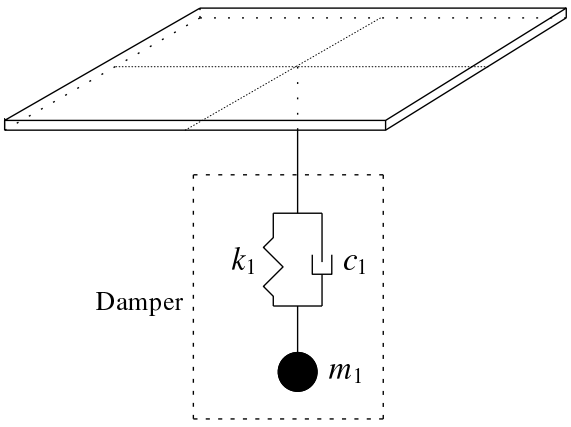


Figure 14: The Conceptual Model of the Floor Damper.

and $[1e6\,{}^{\mathrm{N}}/_{\mathrm{m}}, 1.5e6\,{}^{\mathrm{N}}/_{\mathrm{m}}]$, respectively. The discretized model which is describing the dynamic system is

$$\begin{cases} \left((1+0.02i)K_0 + (k_1 + i\omega c_1)K_1 - \omega^2 M\right) x &= f \\ y &= c^* x. \end{cases}$$

For the selection of the interpolation points in the rectangle $[1e3, 4e3] \times [1e6, 1.5e6]$, we again use a lattice rule. The unit square should first be shifted and scaled to match the desired region in order to be able to use the lattice points. For this example, we chose only three interpolation points. The interpolation points are shown in Table 6.

Table 6: Parameters $\gamma^i = (c_1, k_1)$ are found by lattice points (α_1, α_2) for the second example.

	α_1	α_2	c_1	k_1
$\gamma^{(1)}$	0	0	1000	1e6
$\gamma^{(2)}$	0.33	0.66	2000	1.3e6
$\gamma^{(3)}$	0.66	0.33	3000	1.1e6

Table 7 shows the number of iterations and the CPU times for finding three dominant poles for

Table 7: Number of iterations and CPU time for finding the 3 dominant poles for each interpolation point using Algorithms 4, for the example floor damper with two design parameters.

	Algorithm 4	
	# iteration	time (min)
$\gamma^{(1)}$	6	0.90
$\gamma^{(2)}$	4	0.64
$\gamma^{(3)}$	4	0.64
Total	14	2.18

each interpolation point by applying Algorithm 4. Algorithm 2 used 12 iterations, which required 1.84 minutes CPU time.

We illustrate the quality of the reduction of the method for three different parameters (Table 6) with initial values $s = 1$ and $\mathrm{TOL} = 10^{-7}$. For each parameter $\gamma^{(i)}$, 3 dominant poles are found. So we have the right and left eigenvectors, X, Y, for 9 dominant poles for all $\gamma^{(i)}$.

We now evaluate the reduced model for 81 additional points in Γ obtained by selecting 9 equidistant values for $\gamma_1 \in [1 \times 10^3, 4 \times 10^3]$ and $\gamma_2 \in [1 \times 10^6, 1.5 \times 10^6]$. Figure 15 shows the global transfer function of (12) and global transfer function error of (13) for these 81 points. The relative parametric error (14) for the 81 points lies between 2.9×10^{-4} and 3.8×10^{-4} where $r = 100$ equidistant points are selected in the frequency range $[0\,{}^{\mathrm{rad}}/_{\mathrm{s}}, 350\,{}^{\mathrm{rad}}/_{\mathrm{s}}]$. Figure 16 illustrates the $\log(\mathrm{error}_{\mathrm{rel}})$ against the parameters γ_1 and γ_2. The order of the original system is 29,800 and the frequency range of interest is $[0\,{}^{\mathrm{rad}}/_{\mathrm{s}}, 350\,{}^{\mathrm{rad}}/_{\mathrm{s}}]$.

4.6 Footbridge damper

In this example, we study the footbridge located over the Dijle river in Mechelen, Belgium (see the sketch in Figure 17). It is 31.354 meters in length and the four tuned mass dampers (TMD) are located at main the nodes corresponding to 11.299m, 19.314m, 10.549m and 20.309m, respectively each of which is 40.72kg in weight.

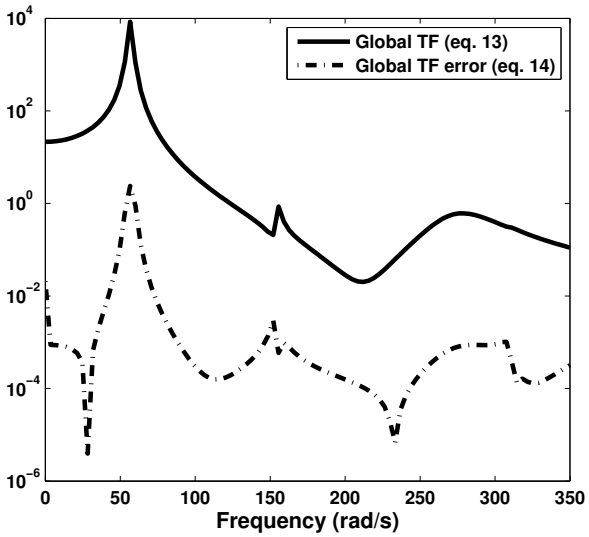


Figure 15: The global transfer function (solid line) and global transfer function error (dashed-dotted line) for 81 parameter points.

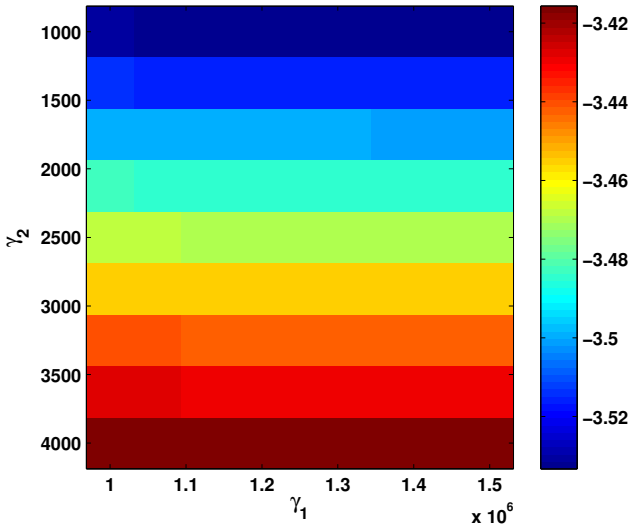


Figure 16: $\log(\text{error}_{\text{rel}})$ for 81 points.

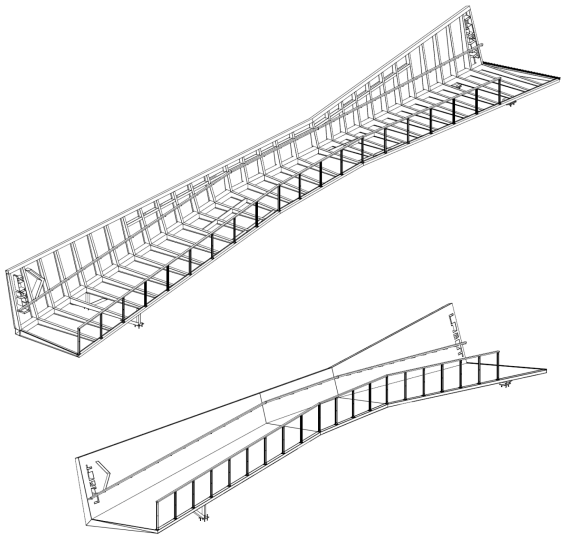


Figure 17: The conceptual model of the footbridge.

The discretized model which is describing the footbridge dynamic system is

$$\begin{cases} \Big(K_0 + i\omega C_0 + \sum_{i=1}^{4}(k_i + i\omega c_i)K_i - \omega^2 M_0\Big)x = f, \\ \qquad\qquad\qquad\qquad\qquad\qquad y = \ell^* x, \end{cases} \tag{17}$$

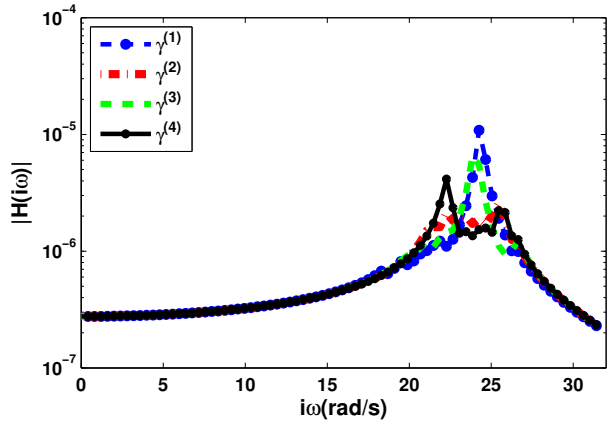
where K_0 and M_0 are obtained from a finite element model with 25962 degrees of freedom (DOFs), $C_0 = 0.1003M_0 + 0.0001591K_0$, K_i is a matrix with four non-zero entries that represents the interaction between the i-th TMD and the footbridge, the input vector f represents a unit excitation at the center span, and the output vector ℓ picks out the vibration at the center span. The frequency range of interest is $[\omega_L, \omega_H] = [0\,\mathrm{rad/s}, 10\pi\,\mathrm{rad/s}]$, [26].

The stiffness k_i and damping c_i coefficients are the 8 parameters of the system (17), so $\gamma^{(j)} = (c_1, c_2, c_3, c_4, k_1, k_2, k_3, k_4)$. Similar to the two previous examples, we use the lattice rule [7] with four interpolation points, which in this case are the points shown in Table 8. The last column

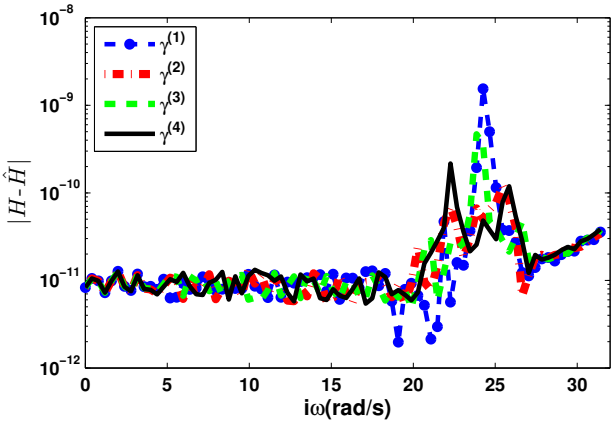
Table 8: The interpolation points for the example of footbridge damper are found by lattice rule parameter $\gamma^i = (c_1, c_2, c_3, c_4, k_1, k_2, k_3, k_4)$.

	$\gamma^{(1)}$	$\gamma^{(2)}$	$\gamma^{(3)}$	$\gamma^{(4)}$	desired interval
c_1	40	50	45	55	$[40, 60]$
c_2	27	37	32	42	$[27, 47]$
c_3	55	45	50	40	$[35, 55]$
c_4	23	33	38	28	$[23, 43]$
k_1	20000	25000	27500	22500	$[20000, 30000]$
k_2	16000	21000	18500	23500	$[16000, 26000]$
k_3	18000	23000	20500	25500	$[18000, 28000]$
k_4	14000	19000	16500	21500	$[14000, 24000]$

of the table gives the range of the parameter corresponding to that row. In this example we are looking for three dominant poles for each $\gamma^{(i)}$, in the frequency range of interest, using (15). Figure 18(a) presents the exact transfer function corresponding to the interpolation points in



(a) Exact transfer functions



(b) absolute errors

Figure 18: Exact transfer function for 4 lattice interpolation points, and absolute error $|H - \widehat{H}|$.

Table 8 and Figure 18(b) shows the absolute error of each $\gamma^{(j)}$. The relative parametric error for 4 lattice interpolation points lies between 2.7×10^{-5} and 5.04×10^{-5} where $r = 80$ equidistant points are selected in the frequency range $[0\,{}^{\mathrm{rad}}/_{\mathrm{s}}, 10\pi\,{}^{\mathrm{rad}}/_{\mathrm{s}}]$.

We have also selected four additional points different from the four interpolation points. Figure 19 shows the global transfer function and global transfer function error for these four points, as

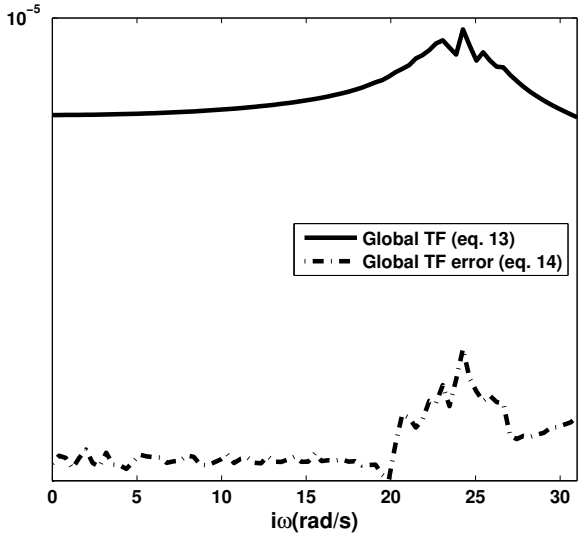


Figure 19: Global transfer function (solid line) and global transfer function error (dashed-dotted line) for four additional lattice points.

a function of ω. The relative parametric error for 4 additional points lies between 2.7×10^{-5} and 3.4×10^{-5} where $r = 80$ equidistant points are selected in the frequency range $[0\,{}^{\mathrm{rad}}/_{\mathrm{s}}, 10\pi\,{}^{\mathrm{rad}}/_{\mathrm{s}}]$. We used Algorithm 2 for finding 3 dominant poles for the four interpolation points. The order of the reduced model is thus $k = 12$. The 12 dominant poles are found in 18 iterations which required a CPU time of 1.15 minutes. Table 9 shows the number of iterations and the CPU times

Table 9: Number of iterations and CPU times for finding the three dominant poles for each interpolation point using Algorithms 4, for the example of footbridge damper.

	Algorithm 4	
	# iteration	time (min)
$\gamma^{(1)}$	8	0.45
$\gamma^{(2)}$	4	0.25
$\gamma^{(3)}$	7	0.43
$\gamma^{(4)}$	3	0.19
Total	22	1.32

for finding the three dominant poles for each interpolation point.

5 Conclusions

We have proposed a dominant pole algorithm for parametric model order reduction. For the applications we solved, a few dominant eigenvalues are sufficient for a good approximation, as

well as a small number of interpolation points in the parameter space. We have also shown that it is advantageous to reuse eigenvectors and eigenvalues for previously computed parameters as starting guesses for the coming parameters.

Acknowledgements

This paper presents research results of the Belgian Network DYSCO, funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State Science Policy Office. The scientific responsibility rests with its authors. The research is also partially funded by the Research Council KU Leuven grants PFV/10/002 and OT/10/038. We also thank Geert Lombaert for providing us with the footbridge example.

References

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

[2] Z. Bai and R. W. Freund. A partial Padé-via-Lanczos method for reduced-order modeling. *Linear Algebra Appl.*, 332–334:139–164, 2001.

[3] M. Barrault, Y. Maday, N. C. Nguyen, and A. T. Patera. An empirical interpolation method: Application to efficient reduced-basis discretization of partial differential equations. *Comptes Rendus Mathematique*, 339:667–672, 2004.

[4] U. Baur, C. Beattie, P. Benner, and Serkan Gugercin. Interpolatory projection methods for parameterized model reduction. *SIAM J. Sci. Comput.*, 33(5):2489–2518, October 2011.

[5] C.A. Beattie and S. Gugercin. Interpolation theory for structure-preserving model reduction. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 4204 –4208, dec. 2008.

[6] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[7] R. Cools, F.Y. Kuo, and D. Nuyens. Constructing embedded lattice rules for multivariate integration. *SIAM Journal on Scientific Computing*, 28:2162–2188, 2006.

[8] R. F. Curtain and H. Zwart. *An introduction to infinite-dimensional linear systems theory*. Springer-Verlag, NY, 1995.

[9] E. D. Denman, J. Leyva-Ramos, and G.J. Jeon. The algebraic theory of latent projectors in lambda matrices. *Applied Mathematics and Computation*, 9:173–300, 1981.

[10] P. Feldman and R. W. Freund. Efficient linear circuit analysis by Padé approximation via the Lanczos process. *IEEE Trans. Computer-Aided Design*, CAD-14:639–649, 1995.

[11] Y. Li, Z. Bai, and Y. Su. A two-directional Arnoldi process and its application to parametric model order reduction. *Journal of computational and applied mathematics*, 226(1):10–21, 2009.

[12] Y. T. Li, Z. Bai, Y. Su, and X. Zeng. Model order reduction of parameterized interconnect networks via a two-directional Arnoldi process. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(9):1571–1582, 2008.

[13] Y.T. Li, Z. Bai, Y. Su, and X. Zeng. Parameterized model order reduction via a two-directional Arnoldi process. In *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, pages 868–873, 2007.

[14] N. Martins, L. T. G. Lima, and H. J. C. P. Pinto. Computing dominant poles of power system transfer functions. *IEEE Trans. Power Syst*, 11:162–170, 1996.

[15] K. Meerbergen. Locking and restarting quadratic eigenvalue solvers. *SIAM J. Sci. Comput.*, 22(5):1814–1839, 2001.

[16] S.-I. Niculescu. *Delay effects on stability. A robust control approach.* Springer-Verlag London, 2001.

[17] A. Odabasioglu, M. Celik, and L. Paganini. Prima: Passive reduced-order interconnect macromodeling algorithm. *IEEE TCAD of Integ. Circuits and Systems*, 17(8):645–654, 1998.

[18] J. Rommes and N. Martins. Efficient computation of multivariable transfer function dominant poles using subspace acceleration. *IEEE Trans. Power Syst.*, 21:1471–1483, 2006.

[19] J. Rommes and N. Martins. Efficient computation of transfer function dominant poles using subspace acceleration. *IEEE Trans. Power Syst.*, 21:1218–1226, 2006.

[20] J. Rommes and N. Martins. Computing transfer function dominant poles of large-scale second-order dynamical systems. *SIAM J. Sci. Comput.*, 30(4):2137–2157, 2008.

[21] J. Rommes and G. L. G. Sleijpen. Convergence of the dominant pole algorithm and Rayleigh quotient iteration. *SIAM Journal on Matrix Analysis and Applications*, 1:346–363, 2008.

[22] Y. Saad. *Numerical methods for large eigenvalue problems.* Manchester University Press, 1992.

[23] M. Saadvandi, K. Meerbergen, and E. Jarlebring. On dominant poles and model reduction of second order time-delay systems. *Appl. Numer. Math.*, 62(1):21–34, 2012.

[24] J. Smith, J. Hauer, and D. Trudnowski. Transfer function identification in power system applications. *IEEE Trans. Power Systems*, 8(3):1282–1290, 1993.

[25] A. Stathopulos, Y. Saad, and K. Wu. Dynamic thick restarting of the Davidson and implicitly restarted Arnoldi methods. *SIAM Journal on Scientific Computing*, 19(1):227–245, 1998.

[26] R. Van Beeumen, K. Van Nimmen, G. Lombaert, and K. Meerbergen. Model reduction for dynamical systems with quadratic output. *International Journal for Numerical Methods in Engineering*, 91(3):229–248, June 2012.

[27] J.H. Wilkinson. *The algebraic eigenvalue problem.* Clarendon Press, Oxford, UK, 1965.

[28] Y. Yue and K. Meerbergen. Using Krylov-Padé model order reduction for accelerating design optimization of structures and vibrations in the frequency domain. *International Journal for Numerical Methods in Engineering*, 90(10):1207–1232, 2012.