On dominant poles and model reduction of second order time-delay systems

Maryam Saadvandi
Karl Meerbergen
Elias Jarlebring

Report TW 584, January 2011

On dominant poles and model reduction of second order time-delay systems

Maryam Saadvandi
Karl Meerbergen
Elias Jarlebring

Report TW 584, January 2011

Department of Computer Science, K.U.Leuven

Abstract

The method known as the *dominant pole algorithm* (DPA) 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. In this paper, we show how this approach can be adapted to a class of second order delay systems, which are large scale nonlinear problems whose transfer functions have an infinite number of simple poles. Deflation is a very important ingredient for this type of methods. Because of the nonlinearity, many deflation approaches for linear systems are not applicable. We therefore propose an alternative technique that essentially removes computed poles from the system's input and output vectors. In general, this technique changes the residues, and hence, modifies the order of dominance of the poles, but we prove that, under certain conditions, the residues stay near the original residues. The new algorithm is illustrated by a numerical example.

Keywords : second order time-delay system, dominant poles, residue, eigenvalue problem, model order reduction.
MSC : Primary : 65F15, 65F50, 65F30

On Dominant Poles and Model Reduction of Second Order Time-Delay Systems

Maryam Saadvandi, Karl Meerbergen and Elias Jarlebring

Department of Computer Science, KULeuven

Abstract

The method known as the *dominant pole algorithm* (DPA) 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. In this paper, we show how this approach can be adapted to a class of second order delay systems, which are large scale nonlinear problems whose transfer functions have an infinite number of simple poles. Deflation is a very important ingredient for this type of methods. Because of the nonlinearity, many deflation approaches for linear systems are not applicable. We therefore propose an alternative technique that essentially removes computed poles from the system's input and output vectors. In general, this technique changes the residues, and hence, modifies the order of dominance of the poles, but we prove that, under certain conditions, the residues stay near the original residues. The new algorithm is illustrated by a numerical example.

Key words: second order time-delay system, dominant poles, residue, eigenvalue problem, model order reduction.

1 Introduction

The analysis of vibrations often leads to large scale systems of ordinary differential equations. In this paper, we consider a second order system with delays,

$$\left\{ \begin{array}{rcl} M\ddot{\underline{\mathbf{x}}}(t) + C\dot{\underline{\mathbf{x}}}(t) + K\underline{\mathbf{x}}(t) & = & \mathbf{f}\underline{u}(t) - G\dot{\underline{\mathbf{x}}}(t-\tau) - F\underline{\mathbf{x}}(t-\tau) \\ \underline{y}(t) & = & \mathbf{d}^*\underline{\mathbf{x}}(t) \end{array} \right. \tag{1.1}$$

where $M, C, K \in \mathbb{R}^{n\times n}$ are mass, damping and stiffness matrices, respectively and $G, F \in \mathbb{R}^{n\times n}$ can be interpreted as matrices stemming from a delayed control of the system. The vector $\mathbf{f} \in \mathbb{R}^n$ is an external force, $\mathbf{d} \in \mathbb{R}^n$ is the output vector and $\underline{\mathbf{x}}(t) \in \mathbb{C}^n$ is the state vector and $\underline{u}(t)$ is the input and $\underline{y}(t)$ is the output; $\tau > 0$ is the delay.

Many properties of vibrations can be studied in the frequency domain, here given by the Fourier transform of the state equation (1.1),

$$\left\{ \begin{array}{rcl} -\omega^2 M\mathbf{x}(\omega) + i\omega C\mathbf{x}(\omega) + K\mathbf{x}(\omega) & = & \mathbf{f}u(\omega) - i\omega G e^{-i\tau\omega}\mathbf{x}(\omega) - F e^{-i\tau\omega}\mathbf{x}(\omega) \\ y(\omega) & = & \mathbf{d}^*\mathbf{x}(\omega) \end{array} \right. \tag{1.2}$$

where we take $u \equiv 1$. A frequency domain representation is particularly useful when only a particular frequency range is of interest. The relevant dynamical behaviour of a system can indeed often be extracted from a given frequency range.

By eliminating the state $\mathbf{x}(\omega)$ in (1.2) we have that the input and output in the frequency domain are related by

$$y(\omega) = H(i\omega)u(\omega),$$

where $H(i\omega)$ is called the *frequency response function* (FRF) and where $H : \mathbb{C} \to \mathbb{C}$

$$H(s) := \mathbf{d}^*(s^2M + sC + K + sGe^{-\tau s} + Fe^{-\tau s})^{-1}\mathbf{f} \tag{1.3}$$

is called the transfer function and can be obtained by applying the Laplace transform to (1.1) under the condition $\underline{\mathbf{x}}(0) = \mathbf{0}$.

As usual, we will define the *poles of the system* as the poles of H and denote them by λ_i, $i \in \mathbb{N}$. Note that a time-delay system generically has a countably infinite number poles since it has a countably infinite number of eigenvalues [18, Chapter 1].

The analysis of vibrations originating from finite element discretizations normally leads to second order systems with large sparse matrices. In such situations we often need to evaluate (1.3) for a large number of values of the parameter ω. For each ω, a large sparse linear system needs to be solved. In this context, the concept of *model order reduction* (MOR) involves approximating (1.2) in some way such that the simulation is computationally cheaper but the relevant dynamics of the system is maintained.

Model order reduction methods for standard linear time-invariant systems, where H is a rational function, have been extensively studied in the literature and the existing approaches have advantages and disadvantages. There are, for instance, moment-matching approaches based on Krylov methods which are usually cheap to compute, but do not always produce the most accurate or smallest reduced model [1]. Balanced truncation is a technique which is optimal (in the sense of Hankel norm error) [1, 16, 20, 21] and is based on classifying states in terms of controllability and observability. Another class of approaches is based on a *modal expansion* and will be the type of approach in this paper. See [24] and references therein for similar approaches. The essential idea is to consider expansions of the type

$$H(s) \approx \tilde{H}(s) := \sum_{i=1}^{m} \frac{R_i}{s - \lambda_i}, \tag{1.4}$$

and approximate H by using only the terms which are (in some sense) dominant. The approximation is sometimes (e.g. [24]) called the *modal equivalent* (approximation) and it can be used for, e.g., identification [28]. This approach is also referred to as modal truncation or modal superposition [3]. The subtle difference lies in that the dominant pole algorithm only computes the poles that are dominant in the frequency response, whereas the modal truncation method uses all poles corresponding to a frequency range.

In the presence of delays, the situation is more complicated and the model reduction approaches for systems without delays mentioned above are normally not directly applicable. However, some model reduction methods are available also for systems with delays. An extension of the one sided moment matching Arnoldi method was recently proposed for time-delay systems [17]. There are also methods with interpretations as rational approximations [15]. See the references in [17] for more literature on MOR for time-delay systems. Note that the dominant pole algorithm has, to our knowledge, not been extended and adapted for time-delay systems or more general nonlinear systems.

In order to carry out model reduction based on dominant poles and to characterize the relevant terms in an expansion similar to (1.4), we introduce the concept of dominance, justify an expansion and derive formulae for this expansion in Section 2. We will adapt the method known as the dominant pole algorithm (DPA) to compute the dominant poles, since it is known to favor poles which are dominant [26]. The dominant pole algorithm (DPA) was introduced for first order systems in [24], generalized to multiple-input-multiple-output (MIMO) systems in [23] and to second order systems in [25]. The derivation presented in these works, which are based on Newton's method, carry over naturally to our context and we present the derivation in Section 3.

DPA can be improved in several ways. In the original version of the dominant pole algorithm only one dominant pole could be found. Rommes and Martins [23–25] introduced a subspace method with deflation in order to compute more than one pole. Deflation is important since it removes converged poles from the system, so that they are not recomputed in the following iterations of DPA, at least not with finite dimensions. For linear systems, this corresponds well to the deflation or locking of eigenvalues that are used in methods for large scale eigenvalue problems. The deflation in [23–25] does not carry over to our context since they are based on a transformation to first-order form which is not available for time-delay systems. For this reason, we will propose a subspace method with new type of deflation (in Section 3.2). It is based on modifying **f** and **d** such it is less likely that poles which are already computed (called deflated poles) will be recomputed. As usual (in e.g. [25]) the reduced model is constructed by projecting the original system onto the subspaces spanned by the right and left eigenvectors, computed by the dominant pole algorithm.

In this context we also wish to point out that there exist approaches for general nonlinear dynamical systems. For example, the trajectory piecewise linear (TPWL) approach [19, 22, 29] linearizes the nonlinear system around a finite number of suitably selected states and approximates the nonlinear system by a piecewise linearization that is obtain from combining the (reduced) linearized systems via a weighting procedure. Proper orthogonal decomposition (POD) is a method that derives a reduced model for (a possibly nonlinear) system by taking snapshots in time [4,6,11]. POD does not need any information about the particular structure of the problem. As it is based on the snapshots only, it may be applied in a straightforward fashion to arbitrary nonlinear problems.

The paper is organized as follows. In Section 2, we define the concept of poles, residues and dominant poles. We introduce the definition of a system with an infinite number of simple poles. In Section 3 we present the DPA for computing dominant poles which typically consists two steps: modal expansion step (subsection 3.1) and projection step (subsection 3.2). Section 4 describes the deflation technique. Numerical examples illustrating the deflation are given in Section 5.

2 Residue and ordering poles by dominance

The model reduction as well as the dominant pole algorithm we will present in later sections are based on the fact that some poles of the system are more important than other. In this section we will introduce an ordering concept of the poles, which is expressed in terms of coefficients in a rational expansion of the inverse of an associated matrix functions. This expansion is called the modal expansion. We first introduce the modal expansion for (1.1) by making an analogy with matrix polynomials. Consider a matrix polynomial

$$P(\lambda) = A_0 + A_1\lambda + \cdots + A_m\lambda^m, \;\; A_0, \ldots, A_m \in \mathbb{C}^{n\times n},$$

and A_m is non-singular. The problem of finding the triple $(\mathbf{x}_i, \mathbf{y}_i, \lambda_i)$ such that

$$P(\lambda_i)\mathbf{x}_i = 0, \quad \mathbf{y}_i^* P(\lambda_i) = 0 \quad \text{and} \quad \mathbf{x}_i \neq 0, \mathbf{y}_i \neq 0,$$

is called the polynomial eigenvalue problem (PEP). If all eigenvalues are simple, then the inverse of the matrix polynomial $P(\lambda)$ can be written as a sum of rank-one matrices. More precisely,

$$P(\lambda)^{-1} = \sum_{i=1}^{nm} \frac{\mathbf{x}_i\mathbf{y}_i^*}{\mathbf{y}_i^* P'(\lambda_i)\mathbf{x}_i} \frac{1}{\lambda - \lambda_i}. \tag{2.5}$$

This is shown in, e.g. [8]. In order to simplify the notation we will now introduce

$$A(\lambda) := \lambda^2 M + \lambda C + K + \lambda G e^{-\lambda\tau} + F e^{-\lambda\tau}.$$

Hence,

$$A'(\lambda) = 2\lambda M + C + G e^{-\lambda\tau} - \lambda\tau G e^{-\lambda\tau} - \tau F e^{-\lambda\tau}.$$

We need an expansion similar to (2.5) for $A(\lambda)$. The derivation of the expansion (2.5) in [8] is based on rewriting the PEP as a generalized eigenvalue problem, i.e., it is based on a transformation to first-order form. The approach to rewrite the system to first order (as in [8]) is not applicable in our context since time-delay systems can not be transformed to (finite-dimensional) first-order systems. We are not aware of results similar to (2.5) for arbitrary nonlinearities. It does not exist in the standard literature on time-delay systems [2, 10, 18] and the derivations for PEPs (e.g. [8, 9, 14]) appear all to be based on linearization. For our setting, we can however derive a result of the corresponding expansion which is not based on rewriting the system to (finite-dimensional) first-order form.

Theorem 1 (Modal expansion) *Suppose the time-delay system* (1.1) *only has simple eigenvalues and suppose M is non-singular. Let $\mathbf{x}_i$ and $\mathbf{y}_i$ are right and left eigenvectors corresponding to eigenvalue λ_i, respectively. Then, for any $s \in \mathbb{C}$ such that s is not an eigenvalue, the transfer function can be expanded as,*

$$H(s) = \mathbf{d}^* A(s)^{-1}\mathbf{f} = \sum_{i=1}^{\infty} \frac{R_i}{s-\lambda_i}, \tag{2.6}$$

where $R_i \in \mathbb{C}$ is called the residue *and given by*

$$R_i = \frac{(\mathbf{d}^*\mathbf{x}_i)(\mathbf{y}_i^*\mathbf{f})}{\mathbf{y}_i^* A'(\lambda_i)\mathbf{x}_i}. \tag{2.7}$$

Proof. First note that the inverse of A can be expanded as,

$$A(\lambda)^{-1} = \sum_{i=1}^{\infty} \frac{P_i}{\lambda-\lambda_i}. \tag{2.8}$$

This follows from [7, Lemma 4.3.10] and the fact that the system (1.1) can be written as a first-order time-delay system using a companion linearization process (under the assumption that M is non-singular). The companion linearization is straightforward, but also given Section 3.2.

The rest of the proof consists of determining a formula for P_i from which (2.7) follows directly. We multiply (2.8) from the right by $(\lambda-\lambda_j)A(\lambda)$,

$$\begin{aligned} I(\lambda-\lambda_j) =& \sum_{i=1}^{\infty} \frac{\lambda-\lambda_j}{\lambda-\lambda_i} P_i A(\lambda) \\ =& P_1 A(\lambda)\frac{\lambda-\lambda_j}{\lambda-\lambda_1} + \cdots + P_{j-1}A(\lambda)\frac{\lambda-\lambda_j}{\lambda-\lambda_{j-1}} + P_j A(\lambda) + \\ & P_{j+1}A(\lambda)\frac{\lambda-\lambda_j}{\lambda-\lambda_{j+1}} + \cdots + P_m A(\lambda)\frac{\lambda-\lambda_j}{\lambda-\lambda_m} + \cdots . \end{aligned} \tag{2.9}$$

Since all eigenvalues are distinct and the equality must hold for any λ, it must hold for $\lambda = \lambda_j$. All terms but one vanish and we have that

$$P_j A(\lambda_j) = 0. \tag{2.10}$$

Analogously, by multiplying (2.8) from the left by $(\lambda-\lambda_j)A(\lambda)$, we also have that

$$A(\lambda_j)P_j = 0. \tag{2.11}$$

Now note that $A(\lambda_j)$ has a one-dimensional nullspace (since λ_j is simple) given by the eigenvectors. From (2.10) and (2.11) it follows that P_j has the structure

$$P_j = \gamma_j \mathbf{x}_j \mathbf{y}_j^*, \tag{2.12}$$

where it remains to determine the scalar γ_j. The derivative of (2.9) with respect to λ is

$$I = P_1 A'(\lambda)\frac{\lambda-\lambda_j}{\lambda-\lambda_1} + P_1 A(\lambda)\frac{\lambda_j-\lambda_1}{(\lambda-\lambda_1)^2} + \cdots + P_j A'(\lambda) + \\ \cdots + P_m A'(\lambda)\frac{\lambda-\lambda_j}{\lambda-\lambda_m} + P_m A(\lambda)\frac{\lambda_j-\lambda_m}{(\lambda-\lambda_m)^2} + \cdots .$$

With $\lambda = \lambda_j$ we find

$$I = P_1 A(\lambda_j)\frac{1}{\lambda_j-\lambda_1} + \cdots + P_j A'(\lambda_j) + \cdots + P_m A(\lambda_j)\frac{1}{\lambda_j-\lambda_m} + \cdots .$$

Multiplication from the right and the left by $\mathbf{x}_j$ and $\mathbf{y}_j^*$, respectively, and substitution of P_j from (2.12), yields

$$\gamma_j = \frac{1}{\mathbf{y}_j^* A'(\lambda_j)\mathbf{x}_j}. \tag{2.13}$$

By combining (2.12) and (2.13) we have derived the explicit expression for P_i,

$$P_i = \frac{\mathbf{x}_i \mathbf{y}_i^*}{\mathbf{y}_i^* A'(\lambda_i)\mathbf{x}_i}.$$

The residue expansion (2.6) follows directly from insertion into the formula for P_i. □

We will now introduce the concept of dominance. There are different ways to order the poles. A common approach is to use the modulus of the residue and weight it such that eigenvalues close to the imaginary axis have higher dominance [23–26].

Definition 1 *The* weighted residue*, denoted by ρ_i, is defined as*

$$\rho_i = \frac{|R_i|}{|\mathrm{Re}(\lambda_i)|}, \tag{2.14}$$

where R_i is the associated residue of eigenvalue λ_i.

Definition 2 *The pole λ_i of $H(s)$ with corresponding weighted residue ρ_i is called the* dominant pole *if and only if*

$$\rho_i > \rho_j$$

for all $j \neq i$.

Let us now assume that the terms in (2.6) are ordered following decreasing weighted residues, i.e.,

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

This type of ordering is the basis for the methods known as modal truncation, where we approximate $H(s)$ by

$$H(s) \approx \tilde{H}(s) = \sum_{j=1}^{p} \frac{R_j}{s-\lambda_j}. \tag{2.15}$$

The terms of $\tilde{H}(s)$ are thus ordered following decreasing modulus. This approach is successful we have a good approximation even when $p \ll n$. We will see in the next section, how the left and right eigenvectors associated with the poles in (2.15) will be used to build the reduced model.

3 The dominant pole algorithm for time-delay systems

We have now defined (in Definition 2) the dominant poles as the poles with largest weighted residues. The transfer function of a large-scale second order time-delay system usually has a small number of dominant poles compared to the number of variables. In this section we want to make a reduced model by projecting using the matrices X and $Y \in \mathbb{R}^{n\times p}$, whose columns are respectively the right and left dominant eigenvectors. Then the reduced model becomes

$$\begin{cases} s^2\widehat{M}\hat{\mathbf{x}}(s) + s\widehat{C}\hat{\mathbf{x}}(s) + \widehat{K}\hat{\mathbf{x}}(s) + s\widehat{G}e^{-s\tau}\hat{\mathbf{x}}(s) + \widehat{F}e^{-s\tau}\hat{\mathbf{x}}(s) &= \hat{\mathbf{f}}u(s) \\ \hat{y}(s) &= \hat{\mathbf{d}}^*\hat{\mathbf{x}}(s) \end{cases} \tag{3.16}$$

where $\widehat{M} = Y^*MX,\ \widehat{C} = Y^*CX,\ \widehat{K} = Y^*KX,\ \widehat{G} = Y^*GX,\ \widehat{F} = Y^*FX \in \mathbb{R}^{p\times p}$, $\hat{\mathbf{f}} = Y^*\mathbf{f}$ and $\hat{\mathbf{d}} = X^*\mathbf{d} \in \mathbb{R}^{p\times 1}$. An interesting and often useful property of reductions of this type is that the structure is preserved, i.e., (1.2) and (3.16) are of the same type. The transfer function of reduced system (3.16) is

$$\widehat{H}(s) = \hat{\mathbf{d}}^*\widehat{A}(s)\hat{\mathbf{f}}$$

where $\widehat{A}(s) = s^2\widehat{M} + s\widehat{C} + \widehat{K} + s\widehat{G}e^{-\tau s} + \widehat{F}e^{-\tau s}$. We ultimately wish to construct a reduced model (3.16) where $p \ll n$ and $\widehat{H}(s)$ is a 'good' approximation to $H(s)$.

The matrices X and Y are computed by using the dominant pole algorithm (DPA). In subsection 3.1 we extend the idea of DPA to nonlinear systems. In subsection 3.2, we introduce subspace projection for computation of the full set of the dominant poles and the modal bases X and Y.

3.1 Simple dominant pole algorithm

Let the function $Q : \mathbb{C} \to \mathbb{C}$ be defined by

$$Q(s) := \frac{1}{H(s)}.$$

If $s_* \in \mathbb{C}$ is a pole of the system, then $Q(s_*) = 0$, i.e., the poles of $H(s)$ are the roots of $Q(s)$. For finding the roots of $Q(s)$, we use Newton's method. The derivative of $Q(s)$ is

$$Q'(s) = -\frac{H'(s)}{H^2(s)},$$

where $H'(s)$ is given by

$$H'(s) = -\mathbf{d}^*A(s)^{-1}A'(s)A(s)^{-1}\mathbf{f}.$$

In the same spirit as [5], the Newton scheme with initial pole estimate s_k is

$$\begin{aligned} s_{k+1} &= s_k - \frac{Q(s_k)}{Q'(s_k)} \\ &= s_k + \frac{1}{H(s_k)}\frac{H(s_k)^2}{H'(s_k)} \\ &= s_k - \frac{\mathbf{d}^*\mathbf{v}_k}{\mathbf{w}_k^*A'(s_k)\mathbf{v}_k}, \end{aligned} \tag{3.17}$$

where $\mathbf{v}_k = A(s)^{-1}\mathbf{f}$ and $\mathbf{w}_k = A(s)^{-*}\mathbf{d}$. The Newton scheme (3.17) is presented in Algorithm 3.1, which we call DPATD.

Algorithm 3.1 (DPATD)

```
INPUT: System $(M, C, K, G, F, \mathbf{f}, \mathbf{d}, \tau)$, initial value $s$, TOL$\ll 1$
OUTPUT: Dominant pole $s$, $\mathbf{x}, \mathbf{y}$ right and left corresponding eigenvectors
1: repeat
2:    Solve $\mathbf{v} \in \mathbb{C}$ from $A(s)\mathbf{v} = \mathbf{f}$
3:    Solve $\mathbf{w} \in \mathbb{C}$ from $A(s)^*\mathbf{w} = \mathbf{c}$
4:    Compute the new pole
          $$s = s - \frac{\mathbf{d}^*\mathbf{v}}{\mathbf{w}^*A'(s)\mathbf{v}}$$
5:    Let $\mathbf{x} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$, $\mathbf{y} = \frac{\mathbf{w}}{\|\mathbf{w}\|}$
6: until
          $\mathtt{max}(\|A(s)\mathbf{x}\|, \|A(s)^*\mathbf{y}\|) < \text{TOL}$
```

3.2 Subspace projection and selection of dominant poles

The DPATD algorithm can compute only one dominant pole. In addition, its behavior is not very easy to predict, in the sense that the Newton process may converge to any pole. In [26], it is shown that the convergence is relatively reliable in the sense that the dominant pole is often found. It appears to be more reliable than the Rayleigh quotient iteration method, which is a Newton method for the eigenvalue problem. In order to increase the reliability even more, subspace projection was used in [24]. In addition subspace projection speeds up the method and allows us to compute more than one dominant pole. The idea of subspace projection originates from the Jacobi-Davidson eigenvalue solver [27], which is a combination of the Newton process and subspace projection. In the context of the computation of dominant poles, two subspaces are stored: one for the search space of the right eigenvectors and one for the left eigenvectors. We denote them by V and W, respectively. The overview of the dominant pole algorithm with subspace projection for second order time-delay system (SPDPATD) is presented in Algorithm 3.2. We now discuss the different steps of the algorithm.

Algorithm 3.2 (SPDATD)

```
INPUT: System $(M, C, K, G, F, \mathbf{f}, \mathbf{d}, \tau)$, initial value $s$ in the upper half plane, TOL, $k_{\min}$, $k_{\max}$,
    wanted number of poles $p_{\text{wanted}}$.
OUTPUT: Dominant poles $\Lambda = \mathtt{diag}(\lambda_1, \cdots, \lambda_p)$, $X$, $Y$ associated eigenvectors.
1: Set $\Lambda = X = Y = V = W = [\,]$.
2: while size of $\Lambda$ is less than $p_{\text{wanted}}$ do
3:    Solve $\mathbf{v}$ from $A(s)\mathbf{v} = \mathbf{f}$ .
4:    Solve $\mathbf{w}$ from $A(s)^*\mathbf{w} = \mathbf{d}$.
5:    Let $V = \mathtt{orthogonal}(V, \mathbf{v})$ and $W = \mathtt{orthogonal}(W, \mathbf{w})$.
6:    $[\widetilde{\Lambda}, \widetilde{X}, \widetilde{Y}]$ = Solve eigenvalue problem (3.2).
7:    $[\widetilde{\Lambda}, \widetilde{X}, \widetilde{Y}]$ = sort the $[\widetilde{\Lambda}, \widetilde{X}, \widetilde{Y}]$ in decreasing $|R_i|/|\text{Re}(\lambda_i)|$ order.
8:    Compute dominant Ritz triplet $(\lambda, \mathbf{x}, \mathbf{y})$
      $\lambda = \widetilde{\Lambda}(1)$,   $\mathbf{x} = V\widetilde{X}(:,1)$,   $\mathbf{y} = W\widetilde{Y}(:,1)$,   $\mathbf{x} = \mathbf{x}/\|\mathbf{x}\|$,   $\mathbf{y} = \mathbf{y}/\|\mathbf{y}\|$.
9:    if $\mathtt{max}(\|A(\lambda)\mathbf{x}\|, \|A(\lambda)^*\mathbf{y}\|) < \text{TOL}$ then
10:      Set $\Lambda = [\Lambda, \lambda, \bar{\lambda}]$,   $X = [X\ \text{Re}(\mathbf{x})\ \text{Im}(\mathbf{x})]$,   $Y = [Y\ \text{Re}(\mathbf{y})\ \text{Im}(\mathbf{y})]$
11:      Remove the first column from $\widetilde{X}$ and $\widetilde{Y}$.
12:   end if
13:   if the number of columns of $V$ and $W$ is $k_{\max}$, then
14:      Let $\widetilde{X}_{k_{\min}} = [\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_{k_{\min}}]$ and $\widetilde{Y}_{k_{\min}} = [\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{k_{\min}}]$.
15:      Orthogonalize the columns of $\widetilde{X}_{k_{\min}}$ and $\widetilde{Y}_{k_{\min}}$.
16:      Let $V = V\widetilde{X}_{k_{\min}}$ and $W = W\widetilde{Y}_{k_{\min}}$.
```

```
17:     end if
18:     Choose $s$ as the most dominant unconverged pole in the upper half plane.
19: end while
```

The algorithm computes the poles one by one. The initial value of s is given as input to the algorithm. For choosing the initial value, we let $\tau = 0$ and solve the quadratic eigenvalue problem $(\lambda^2 M + \lambda C + K + \lambda G + F)\mathbf{x} = 0$ and choose the eigenvalue nearest the origin. We now discuss the different steps of the algorithm.

In lines 3 and 4, we solve $\mathbf{v}$ and $\mathbf{w}$ from two linear systems. In our experiments, we used the matlab backslash operator, which is a direct linear system solver. In production code, the matrix factorization for $A(s)$ could be reused for computing $\mathbf{w}$, which basically reduces the computational cost by two. Alternatively, an iterative linear system solver can be employed.

In line 5, the vector $\mathbf{v}$ is added to the matrix V, whose columns store the iteration vectors. Similarly $\mathbf{w}$ is stored in W. Gram-Schmidt orthogonalization is used in order to obtain orthogonal columns in V and W respectively.

In line 6, we solve a small scale but nonlinear eigenvalue problem

$$(\tilde{\lambda}_i^2 \widetilde{M} + \tilde{\lambda}_i \widetilde{C} + \widetilde{K} + \tilde{\lambda}_i \widetilde{G} e^{-\tilde{\lambda}_i \tau} + \widetilde{F} e^{-\tilde{\lambda}_i \tau})\tilde{\mathbf{x}}_i = 0, \qquad \tilde{\mathbf{x}}_i \neq 0 \quad (i \in \mathbb{N}),$$
$$(\tilde{\lambda}_i^2 \widetilde{M} + \tilde{\lambda}_i \widetilde{C} + \widetilde{K} + \tilde{\lambda}_i \widetilde{G} e^{-\tilde{\lambda}_i \tau} + \widetilde{F} e^{-\tilde{\lambda}_i \tau})^* \tilde{\mathbf{y}}_i = 0, \qquad \tilde{\mathbf{y}}_i \neq 0 \quad (i \in \mathbb{N}),$$

where $\widetilde{M} = W^* M V$, $\widetilde{C} = W^* C V$, $\widetilde{K} = W^* K V$, $\widetilde{G} = W^* G V$ and $\widetilde{F} = W^* F V \in \mathbb{R}^{k \times k}$. We assume $k \ll n$. We therefore reformulate (3.2) as the following eigenvalue problem of double dimension

$$\begin{cases} (\lambda\ E - A_0 - A_1\ e^{-\lambda\tau})\ \varphi &= 0, \\ (\lambda\ E - A_0 - A_1\ e^{-\lambda\tau})^* \psi &= 0, \end{cases}$$

where

$$E = \begin{bmatrix} I & 0 \\ 0 & \widetilde{M} \end{bmatrix}, \quad A_0 = \begin{bmatrix} 0 & I \\ -\widetilde{K} & -\widetilde{C} \end{bmatrix}, \quad A_1 = \begin{bmatrix} 0 & 0 \\ -\widetilde{F} & -\widetilde{G} \end{bmatrix}.$$

This problem can be solved by the Arnoldi method recently introduced in [13]. The vectors $\widetilde{\mathbf{x}}_i$ and $\widetilde{\mathbf{y}}_i$ are simply obtained by an inverse iteration step:

$$\begin{aligned} \widetilde{\mathbf{x}}_i &= (\widetilde{\lambda}_i^2 \widetilde{M} + \tilde{\lambda}_i \widetilde{C} + \widetilde{K} + \tilde{\lambda}_i \widetilde{G})^{-1} \widetilde{\mathbf{f}} \\ \widetilde{\mathbf{y}}_i &= (\widetilde{\lambda}_i^2 \widetilde{M} + \tilde{\lambda}_i \widetilde{C} + \widetilde{K} + \tilde{\lambda}_i \widetilde{G})^{-*} \widetilde{\mathbf{d}} \end{aligned}$$

Note that any right-hand side of inverse iteration can be used. We use the eigentriplet $(\tilde{\lambda}_i, \tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i)$ to construct an approximate eigentriplet $(\lambda_i, \mathbf{x}_i, \mathbf{y}_i)$ of the original problem

$$\lambda_i = \tilde{\lambda}_i, \quad \mathbf{x}_i = V\tilde{\mathbf{x}}_i, \quad \mathbf{y}_i = W\tilde{\mathbf{y}}_i.$$

In line 7, the Ritz values are sorted in decreasing residue.

In line 8, we select the current most dominant pole that was not deflated yet and compute the associated right and left Ritz vectors.

In lines 9-12, we check whether the computed poles are accurate using the residual norms $\|A(\lambda)\mathbf{x}\|$ and $\|\mathbf{y}^* A(\lambda)\|$, which are measures of the backward error. If the eigentriplet is considered accurate, it is added to the list of known poles. Since the matrices in (1.1) are real, the eigenpairs come in complex conjugate pairs, so we add $\mathrm{Re}(\mathbf{x})$ and $\mathrm{Im}(\mathbf{x})$ instead of $\mathbf{x}$ and $\overline{\mathbf{x}}$ to X, and add $\mathrm{Re}(\mathbf{y})$ and $\mathrm{Im}(\mathbf{y})$ to Y. In this way, X and Y are real matrices.

In each iteration, one vector is added to V and W respectively. When many poles are wanted or the algorithm converges slowly, the bases may become very large. To reduce the storage cost, the bases V and W are shrinked when they each have $k_{\max}$ columns [24]. The new size of the bases is $k_{\min}$. The basis is chosen so that it spans the $k_{\min}$ most dominant Ritz vectors that have not yet converged. Therefore, $k_{\min}$ is, by preference, larger than or equal to the remaining number of poles to be computed. See lines 13–17.

In line 18, we select s as the most dominant unconverged pole. We make sure we always select s in the upper half plane to prevent the selection of a pole in the lower half plane that was added to the system in steps 9-13. It is indeed possible that if s is selected in the lower half plane, we select the complex conjugate of the pole that just converged. This would obvously be a bad selection.

After finding k dominant poles, the reduced system is constructed by projecting the original system on X and Y, which contain the dominant right and left eigenvectors, respectively. The structure of the original system is preserved by this projection, i.e., the reduced model is also a time-delay system.

4 Deflation

As stated in the introduction, we need to extend Algorithm 3.2 with some procedure (called *deflation*) in order to avoid that the algorithm converges to the same pole several times. We will later illustrate the importance of this with an example in Section 5. See [23, 25] for deflation for first and second order linear dynamical systems.

The deflation technique we will propose for our nonlinear case is conceptually different from other approaches. We believe this is necessary, since the procedures for first and second order systems do not seem to carry over to this nonlinear (and not polynomial) case. In the procedure we propose, we will have to carry out a deflation procedure in each iteration step (and not only when a pole is found). The procedure is also such that the residues of all poles change. Hence, the order of dominance of the poles may change. We will however show that these changes are reasonably small, at least for nearby poles.

The procedure is based on considering a modified system (which we will call the *deflated system*) of the same type as (1.1),

$$\begin{cases} (s^2M + sC + K + sGe^{-\tau s} + Fe^{-\tau s})\mathbf{x}(s) &= \tilde{\mathbf{f}}u(s) \\ y(s) &= \tilde{\mathbf{d}}^*\mathbf{x}(s), \end{cases} \tag{4.18}$$

where we have changed the right-hand side vectors $\mathbf{f}$ and $\mathbf{d}$. We will later let these right-hand sides depend on the iterate s_k and we will therefore now denote them as functions of s, $\tilde{\mathbf{f}}(s)$ and $\tilde{\mathbf{d}}(s)$. Note that (4.18) and (1.2) generically have the same poles, but with different residues. We denote the residue of λ_j associated with the deflated system (4.18) by $\widetilde{R}_j$

$$\widetilde{R}_j = \frac{\left(\tilde{\mathbf{d}}^*(s)\mathbf{x}_j\right)\left(\mathbf{y}_j^*\tilde{\mathbf{f}}(s)\right)}{\mathbf{y}_j^*A'(\lambda_j)\mathbf{x}_j}. \tag{4.19}$$

The general idea of the deflation technique we propose here is based on the fact that we can influence the residues $\widetilde{R}_j$ throughout the iteration by modifying $\tilde{\mathbf{f}}(s)$ and $\tilde{\mathbf{d}}(s)$. Suppose the algorithm has already found the poles $\lambda_1, \dots, \lambda_m$ (called deflated poles). We would, ideally, like to have the situation where

- the residues of the deflated poles $(\lambda_1, \dots, \lambda_m)$ are zero, i.e.,

$$\widetilde{R}_1 = \cdots = \widetilde{R}_m = 0, \tag{4.20}$$

and

- the residues of the other poles remain unchanged, i.e.,

$$\widetilde{R}_i = R_i \quad \text{for all } i > m. \tag{4.21}$$

In this situation, we would not recompute solutions since the computed poles have zero residues. Moreover, the algorithm would be likely to converge to the *next* not-yet-computed dominant pole. The ideal situation does not appear to be achievable with the freedom aviable in (4.18), due to the nonlinearity of $A(s)$. With the ideal situation in mind, we can however design a reasonable way to choose $\tilde{\mathbf{f}}(s)$ and $\tilde{\mathbf{d}}(s)$ and control the residues, such that (4.20) holds and (4.21) holds approximately.

We propose to select $\tilde{\mathbf{f}}(s)$ and $\tilde{\mathbf{d}}(s)$ as follows. Suppose the algorithm has already converged to m poles $\lambda_1, \ldots, \lambda_m$, which we now wish to deflate. We will start by considering $\tilde{\mathbf{f}}(s)$ and $\tilde{\mathbf{d}}(s)$ of the form

$$\left\{ \begin{array}{rcl} \tilde{\mathbf{f}}(s) & = & \mathbf{f} - A(s)\mathbf{X}_m\mathbf{a}_m \\ \tilde{\mathbf{d}}(s) & = & \mathbf{d} - A(s)^*\mathbf{Y}_m\mathbf{b}_m \end{array} \right. \tag{4.22}$$

where $X_m = [\mathbf{x}_1,\ \mathbf{x}_2,\ \cdots,\ \mathbf{x}_m]$, $Y_m = [\mathbf{y}_1,\ \mathbf{y}_2,\ \cdots,\ \mathbf{y}_m]$ are right and left converged eigenvector matrices and $\mathbf{a}_m$ and $\mathbf{b}_m$ are to be determined. In order to achieve properties similar to the ideal situation described above, it is natural to impose $\mathbf{Y}_m^*\tilde{\mathbf{f}}(s) = 0$ and $\tilde{\mathbf{d}}^*(s)\mathbf{X}_m = 0$, since we want (4.20), i.e., that the residues (4.19) corresponding to the deflated poles vanish. These conditions lead to the following choice of the coefficients $\mathbf{a}_m = [\alpha_1, \ldots, \alpha_m]^T$ and $\mathbf{b}_m = [\beta_1, \ldots, \beta_m]^T$,

$$\left\{ \begin{array}{rcl} \mathbf{a}_m & = & (Y_m^* A(s) X_m)^{-1}(Y_m^*\mathbf{f}) \\ \mathbf{b}_m & = & (Y_m^* A(s) X_m)^{-*}(X_m^*\mathbf{d}). \end{array} \right. \tag{4.23}$$

Note that the generic situation is that $Y_m^* A(s) X_m$ is not singular. This can, of course, in general, not be guaranteed. Suppose that $Y_m^* A(s) X_m \mathbf{z} = 0$ and $\mathbf{t}^* Y_m^* A(s) X_m = 0$ for $\mathbf{z} \neq 0$ and $\mathbf{t} \neq 0$. Then s is an eigenvalue of the non-linear eigenvalue problem $Y_m^* A(s) X_m$. If $A(s)$ would be linear in s, this would be impossible when s is different from $\lambda_1, \ldots, \lambda_m$. For a non-linear problem, it is possible. In our numerical experiments, we did, however, not encounter any difficulties of this kind.

From the reasoning above we have the following conclusion.

Theorem 2 *By choosing $\tilde{\mathbf{f}}(s)$ and $\tilde{\mathbf{d}}(s)$ according to* (4.22) *and* (4.23)*, the residue of the deflated poles vanish for almost all s, i.e.,*

$$\widetilde{R}_1 = \cdots = \widetilde{R}_m = 0.$$

Recall that in the ideal situation we would also like the residues of not-yet-converged poles to not change much, preferably not at all as in (4.21). We will show that with the choice (4.22) and (4.23) the change of residue is small for the eigenvalues near s. Since the eigenvalues far away from s are unlikely to be computed (unless perhaps, a far away pole would appear to be very dominant), the change of residue of far away poles is not as important as the residue for nearby poles. This is shown in the following theorem.

Theorem 3 *Suppose λ_j is a simple (not deflated) pole with corresponding eigentriplet $(\lambda_j, \mathbf{x}_j, \mathbf{y}_j)$. By choosing $\tilde{\mathbf{f}}(s)$ and $\tilde{\mathbf{d}}(s)$ according to* (4.22) *and* (4.23)*, the change of the residue R_j when deflating is linear in the difference $s - \lambda_j$, i.e.,*

$$\widetilde{R}_j - R_j = O(s - \lambda_j),$$

where R_j is given by (2.7) *and $\widetilde{R}_j$ is given by* (4.19).

Proof. In this proof, we will need to use the fact that

$$\begin{cases} Y_m^* A(s)\mathbf{x}_j &= (s-\lambda_j) Y_m^* A'(\lambda_j)\mathbf{x}_j + O\left((s-\lambda_j)^2\right) \\ \mathbf{y}_j^* A(s) X_m &= (s-\lambda_j)\mathbf{y}_j^* A'(\lambda_j) X_m + O\left((s-\lambda_j)^2\right). \end{cases} \tag{4.24}$$

This follows directly from the Taylor expansion of $A(s)$ around λ_j,

$$A(s) = A(\lambda_j) + (s-\lambda_j)A'(\lambda_j) + O\left((s-\lambda_j)^2\right),$$

and that $A(\lambda_j)\mathbf{x}_j = 0$ and $\mathbf{y}_j^* A(\lambda_j) = 0$.

We now form the difference $\widetilde{R}_j - R_j$ and note that $\widetilde{R}_j$ and R_j have the same denominator. The numerator of $\widetilde{R}_j - R_j$ is

$$(\tilde{\mathbf{d}}^*(s)\mathbf{x}_j)(\mathbf{y}_j^*\tilde{\mathbf{f}}(s)) - (\mathbf{d}^*\mathbf{x}_j)(\mathbf{y}_j^*\mathbf{f}). \tag{4.25}$$

Substitute the definition of $\tilde{\mathbf{d}}(s)$ and $\tilde{\mathbf{f}}(s)$, i.e., (4.22), in (4.25). We have

$$\begin{aligned} &(\tilde{\mathbf{d}}^*(s)\mathbf{x}_j)(\mathbf{y}_j^*\tilde{\mathbf{f}}(s)) - (\mathbf{d}^*\mathbf{x}_j)(\mathbf{y}_j^*\mathbf{f}) = \\ &-\mathbf{d}^*\mathbf{x}_j\mathbf{y}_j^* A(\lambda_j) X_m \mathbf{a}_m - \mathbf{b}_m Y_m^* A(\lambda_j)\mathbf{x}_j\mathbf{y}_j^*\mathbf{f} + \mathbf{b}_m Y_m^* A(\lambda_j)\mathbf{x}_j\mathbf{y}_j^* A(\lambda_j) X_m \mathbf{a}_m. \end{aligned} \tag{4.26}$$

By using (4.24) in (4.26), it follows that

$$\begin{aligned} &(\tilde{\mathbf{d}}^*(s)\mathbf{x}_j)(\mathbf{y}_j^*\tilde{\mathbf{f}}(s)) - (\mathbf{d}^*\mathbf{x}_j)(\mathbf{y}_j^*\mathbf{f}) = \\ &\qquad - \mathbf{d}^*\mathbf{x}_j(s-\lambda_j)\mathbf{y}_j^* A'(\lambda_j) X_m \mathbf{a}_m - \mathbf{b}_m (s-\lambda_j) Y_m^* A'(\lambda_j)\mathbf{x}_j\mathbf{y}_j^*\mathbf{f} \\ &\qquad\qquad + \mathbf{b}_m (s-\lambda_j)^2\, Y_m^* A'(\lambda_j)\mathbf{x}_j\mathbf{y}_j^* A'(\lambda_j) X_m \mathbf{a}_m + O((s-\lambda_j)^2), \end{aligned}$$

which proves the theorem. □

We are now ready to modify Algorithm 3.2 using deflation. The algorithm is modified in two places. First, we use the modified residues $\widetilde{R}_j$ in line 7 to make sure that deflated eigenvectors do not remain dominant. We noticed, by numerical experiments, that this selection was important for the reliability of the method. However, the modified residues may change the order of dominance of the poles, which may lead to a wrong selection to continue the method. In the following section, we will show a numerical example arising from a second order system with delay for which the impact of the change of residue appears to be minor.

Second, we change $\mathbf{f}$ and $\mathbf{d}$ by the formula (4.22), where X_m (and Y_m) contain the deflated right (and left) eigenvectors, as well as their complex conjugates. With the modified $\mathbf{f}$ and $\mathbf{d}$, the deflated vectors have residue zero and are expected not to appear in the solution of the system. The new vector to expand the subspace is thus

$$\mathbf{v} = A(s)^{-1}(\mathbf{f} - A(s)X_m\mathbf{a}_m) = A(s)^{-1}\mathbf{f} - X_m\mathbf{a}_m. \tag{4.27}$$

Note, that if the deflated vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ are spanned by the columns of V (and similarly, $\mathbf{y}_1, \ldots, \mathbf{y}_m$ are spanned by the columns of W), then after orthogonalization against the columns of V, the term $X_m\mathbf{a}_m$ disappears since it is spanned by the columns of V. That implies that the deflation formulae do not change the subspace. This is not the case when also the complex conjugate vector is used in the deflation, as we do in our algorithm. In this case, there are $\mathbf{x}_j$, $j \in \{1, \ldots, m\}$ that are not in the range of V.

The use of deflation leads to the following algorithm:

Algorithm 4.1 (SPDATD)

```
INPUT: System $(M, C, K, G, F, \mathbf{f}, \mathbf{d}, \tau)$, initial value $s$ in the upper half plane, TOL, $k_{\min}$, $k_{\max}$,
    wanted number of poles $p_{\text{wanted}}$.
OUTPUT: Dominant poles $\Lambda = \texttt{diag}(\lambda_1, \cdots, \lambda_p)$, $X, Y$ associated eigenvectors.
 1: Set $\Lambda = X = Y = V = W = [\,]$, $\tilde{\mathbf{f}} = \mathbf{f}$, $\tilde{\mathbf{d}} = \mathbf{d}$
 2: while size of $\Lambda$ is less than $p_{\text{wanted}}$ do
 3:   Let $\tilde{\mathbf{f}} = \mathbf{f} - X(Y^*A(s)X)^{-1}Y^*\mathbf{f}$ and $\tilde{\mathbf{d}} = \mathbf{d} - Y(Y^*A(s)^*X)^{-*}Y^*\mathbf{d}$.
 4:   Solve $\mathbf{v}$ from $A(s)\mathbf{v} = \tilde{\mathbf{f}}$
 5:   Solve $\mathbf{w}$ from $A(s)^*\mathbf{w} = \tilde{\mathbf{d}}$
 6:   Let $V = \texttt{orthogonal}(V, \mathbf{v})$ and $W = \texttt{orthogonal}(W, \mathbf{w})$
 7:   $[\widetilde{\Lambda}, \widetilde{X}, \widetilde{Y}]$ = Solve eigenvalue problem (3.2)
 8:   $[\widetilde{\Lambda}, \widetilde{X}, \widetilde{Y}]$ = sort the $[\widetilde{\Lambda}, \widetilde{X}, \widetilde{Y}]$ in decreasing $|\widetilde{R}_i|/|\text{Re}(\lambda_i)|$ order
 9:   Compute dominant Ritz triplet $(\lambda, \mathbf{x}, \mathbf{y})$
      $\lambda = \widetilde{\Lambda}(1)$,   $\mathbf{x} = V\widetilde{X}(:,1)$,   $\mathbf{y} = W\widetilde{Y}(:,1)$,   $\mathbf{x} = \mathbf{x}/\|\mathbf{x}\|$,   $\mathbf{y} = \mathbf{y}/\|\mathbf{y}\|$
10:   if $\texttt{max}(\|A(\lambda)\mathbf{x}\|, \|A(\lambda)^*\mathbf{y}\|) <$ TOL then
11:     Set $\Lambda = [\Lambda, \lambda, \overline{\lambda}]$,   $X = [X \ \text{Re}(\mathbf{x}) \ \text{Im}(\mathbf{x})]$,   $Y = [Y \ \text{Re}(\mathbf{y}) \ \text{Im}(\mathbf{y})]$
12:     Remove the first column from $\widetilde{X}$ and $\widetilde{Y}$.
13:   end if
14:   if the number of columns of $V$ and $W$ is $k_{\max}$, then
15:     Let $\widetilde{X}_{k_{\min}} = [\widetilde{\mathbf{x}}_1, \ldots, \widetilde{\mathbf{x}}_{k_{\min}}]$ and $\widetilde{Y}_{k_{\min}} = [\widetilde{\mathbf{y}}_1, \ldots, \widetilde{\mathbf{y}}_{k_{\min}}]$.
16:     Orthogonalize the columns of $\widetilde{X}_{k_{\min}}$ and $\widetilde{Y}_{k_{\min}}$.
17:     Let $V = V\widetilde{X}_{k_{\min}}$ and $W = W\widetilde{Y}_{k_{\min}}$.
18:   end if
19:   Choose $s$ as the most dominant unconverged pole in the upper half plane
20: end while
```

Note that, in practice, we will not add $\mathbf{x}$ and $\overline{\mathbf{x}}$ to X at convergence. We rather add the real and imaginary parts of $\mathbf{x}$ to X. Also, when λ is real, we do not add the complex conjugate vectors. This makes the suspace real and reduces its storage cost by a factor two. It is easy to see that the computation of $\mathbf{a}_m$ and $\mathbf{b}_m$ can still be performed in the same way, i.e., by using (4.23).

5 Numerical example

In this section, we illustrate SPDPATD (Algorithm 4.1) for a second order TDS problem, arising from a coupled mass-spring system with dampers and feedback controls with delay shown in Figure 1. (See [12] for a similar but small scale problem.) It includes n masses, n springs and n dampers. The order of the system is therefore n. A feedback controller with delay is added to the q left-most masses in the figure. The matrices $M, C, K \in \mathbb{R}^{n\times n}$ from (1.1) are

$$M = \begin{bmatrix} m & & & \\ & m & & \\ & & \ddots & \\ & & & m \end{bmatrix}, \quad K = \begin{bmatrix} 2\kappa & -\kappa & & & \\ -\kappa & 2\kappa & -\kappa & & \\ & \ddots & \ddots & \ddots & \\ & & -\kappa & 2\kappa & -\kappa \\ & & & -\kappa & \kappa \end{bmatrix},$$

$$C = \begin{bmatrix} 3c & -2c & & & & \\ -2c & 3c & -c & & & \\ & -c & 3c & -2c & & \\ & & \ddots & \ddots & \ddots & \\ & & & -c & 3c & -2c \\ & & & & -2c & 2c \end{bmatrix},$$

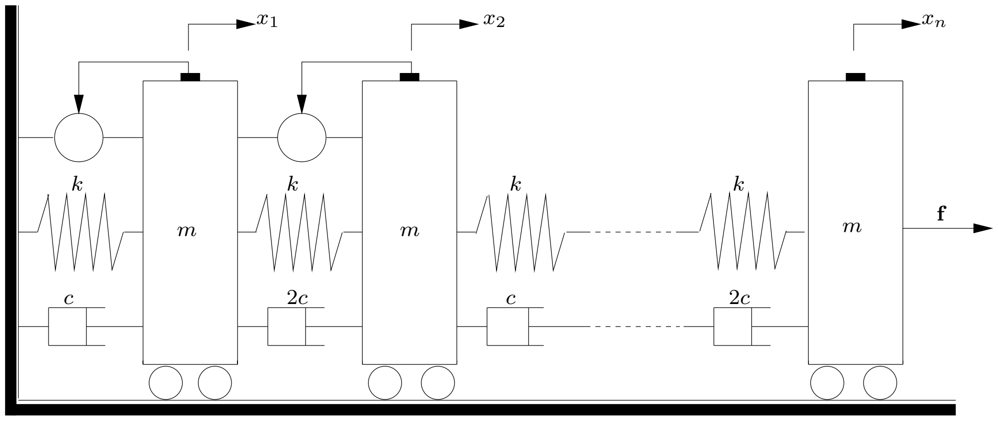


Figure 1: mass-dual system under the state feedback with time-delay

with mass $m = 1$, damping $c = 0.2$ and spring stiffness $\kappa = 1$. The control matrices F and $G \in \mathbb{R}^{n\times n}$ are given by

$$F = \left[\begin{array}{ccccc|c} 0.3 & -0.3 & 0 & & & \\ 0 & 0.3 & -0.3 & & & 0_{q\times(n-q)} \\ & \ddots & \ddots & \ddots & & \\ & & 0 & 0.3 & -0.3 & \\ \hline & & 0_{(n-q)\times q} & & & 0 \end{array}\right],$$

$$G = \left[\begin{array}{ccccc|c} 0.1 & -0.1 & 0 & & & \\ 0 & 0.1 & -0.1 & & & 0_{q\times(n-q)} \\ & \ddots & \ddots & \ddots & & \\ & & 0 & 0.1 & -0.1 & \\ \hline & & 0_{(n-q)\times q} & & & 0 \end{array}\right].$$

The external force $\mathbf{f}$ is a vector with zeros everywhere, except the last entry which is one and we set $\mathbf{d} = \mathbf{f}$. We chose delay $\tau = 2$. In our experiments, we selected $n = 1000$ and $q = 100$. The goal is to compute the transfer function for $s \in i[0, 0.05]$.

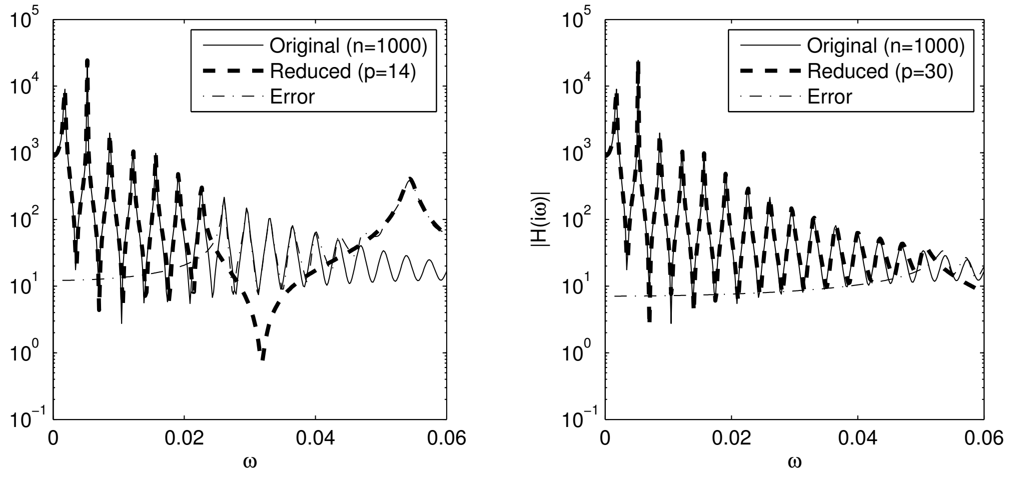
5.1 Illustration of the reduction

We will first demonstrate that the reduced model resulting from computation with Algorithm 4.1, is accurate and has the expected features.

The algorithm is executed with the following parameters. We use that the initial s is chosen from the dominant pole of the problem with $\tau = 0$, which can be computed, e.g., using [25]. In our case, the initial value was $s = 0.000000902811790 + 0.001737926450381i$. The frequency response function (1.3) has one pole near this initial value, and therefore computes the first dominant pole in one iteration only. In our runs, we never shrinked the subspace as indicated in lines 15–19: $k_{\max}$ was always larger than the number of vectors in V and W. For the solution of the small scale eigenvalue problem using the Arnoldi method [13] in line 7 of Algorithm 4.1, we performed as many iterations as required to obtain accurate eigenvalues estimates of the dominant Ritz values.

Figure 2 illustrates the reduction with $p = 14$ and $p = 30$ for TOL $= 10^{-6}$. We notice that the frequency response (or transfer function) is well approximated near the first $p/2$ peaks. The positions of those peaks are close to the imaginary parts of the computed dominant poles. Recall that the poles are complex and the matrices are real. As mentioned before, we also add the complex conjugate of the found dominant poles in order to obtain a real reduced system. We therefore have to compute only $p/2$ (complex) poles.

We notice that with TOL $= 10^{-3}$, some of the low frequency peaks are missing and the positions of the other peaks are not accurate in the frequency response function of the reduced model (as may be expected with such a large value of TOL). We also observed (without presenting the plots) that no higher accuracy was obtained for smaller TOL than 10^{-6}.



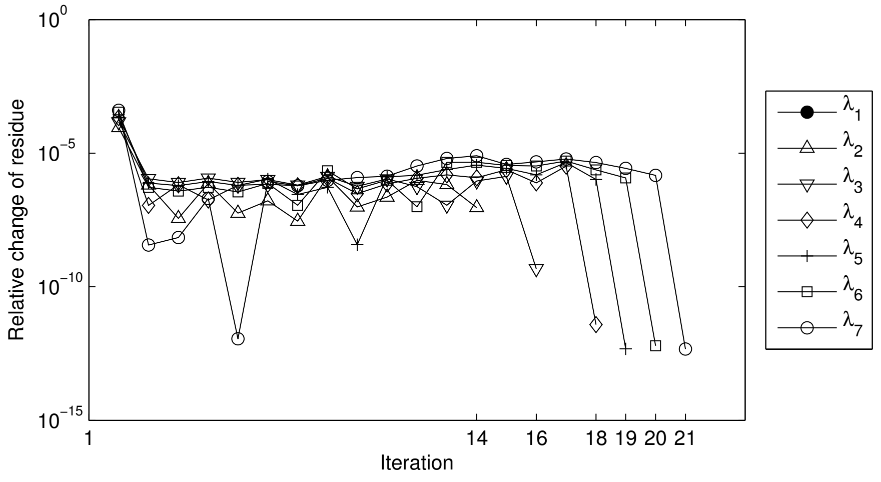
(a) $p = 14$, TOL $= 10^{-6}$ (b) $p = 30$, TOL $= 10^{-6}$

Figure 2: Transfer functions of the original large scale and reduced models, and the error $|\hat{H} - H|$ for different p.

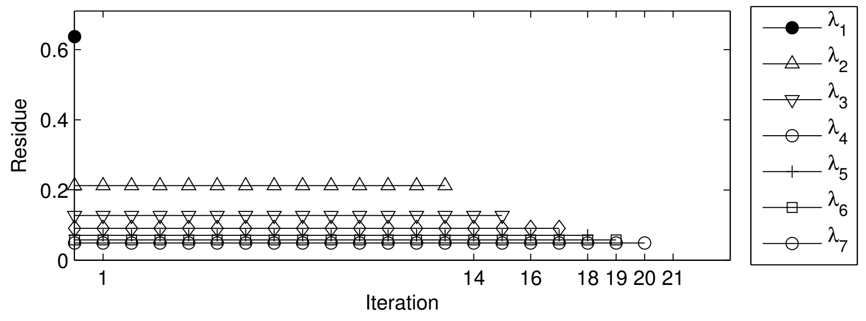
5.2 Illustration of the change or residue

We pointed out in Section 4 that an important property of the proposed deflation strategy is that the residues change and we justify with theory (in Theorem 3) that the change should typically be small. We now wish to illustrate the impact of the change of the residue in Algorithm 4.1 again for the example above. In these simulations we fix TOL $= 10^{-6}$ and $p = 14$.

Figure 3 is a visualization of the residue and change of residue. The figure should be interpreted as follows. In both of the subfigures, the printed iteration numbers (on the horizontal axis) correspond to iterations at which a pole has been flagged converged, i.e., a pole is flagged as converged in iteration 1, 14, 16, 18, 19 and 20. Figure 3(a) shows the relative change in residue, i.e., $|\widetilde{R}_j - R_j|/|R_j|$, for the seven dominant poles as a function of the iteration count. We first observe that the relative change is of order 10^{-6} which is, at least in this situation, sufficiently small to not change the order of dominance for the poles. This can also be observed in Figure 3(b), where we visualize the modified residue as a function of the iteration. It is clear from Figure 3(b), that the order of dominance, determined by (2.14), did not change in the presence of deflation. We also observe Figure 3(b) that (as predicted by Theorem 3) when s converges to a pole, the relative change of residue converges to zero.



(a)



(b)

Figure 3: Figure (a): the lines show the relative change of the residues of the seven dominant poles at each iteration. The poles $\lambda_i (i = 1, \ldots, 7)$ have converged at iterations 1, 14, 16, 18, 19, 20, and 21, respectively. Each line corresponds to a pole. Figure (b) shows the absolute values of the modified weighted residues for each iteration. Each line corresponds to a pole.

5.3 Further insight in the deflation

In order to provide further insight into the impact of the deflation strategy, we will now consider some variants of Algorithm 4.1. In a sense, the deflation has an impact in two places in Algorithm 4.1: the expansion step with deflated right-hand sides (lines 4–5) and the ordering of the poles using modified residues in line 8. We will now treat these components separately and turn off the impact of the deflation of one component at a time. For our purposes it will be sufficient to first consider the following variations on Algorithm 4.1:

V1 This is Algorithm 4.1 without any changes.

V2 The second variation uses the deflated right-hand sides in lines 4 and 5 of Algorithm 4.1, but uses the original residues, i.e., R_j instead of $\widetilde{R}_j$ in line 8.

V3 The third variation uses unchanged right-hand sides $\mathbf{f}$ and $\mathbf{d}$ in lines 4 and 5 of Algorithm 4.1, but uses the modified residues $\widetilde{R}_j$.

Variant	deflated right-hand sides (lines 4–5)	deflated residue (line 8)	$\lambda_{1,2}$	$\lambda_{3,4}$	$\lambda_{5,6}$	$\lambda_{7,8}$	$\lambda_{9,10}$	$\lambda_{11,12}$	$\lambda_{13,14}$
V1	Yes	Yes	1	14	16	18	19	20	21
V2	Yes	No	1	–	–	–	–	–	–
V3	No	Yes	1	4	6	8	9	11	12

Table 1: Convergence of the poles: iteration number at which λ_j converged. A dash indicates, the pole was missed.

Note that the fourth combination (unchanged right-hand side and original residue) completely fails by converging to the first dominant pole all the time and will hence not be reported here. For all runs, we used the same algorithmic parameters: $p = 14$, $k_{\max} = 50$, TOL $= 10^{-6}$ and $s_0 = 9.0281 \times 10^{-7} + 1.7379 \times 10^{-3}i$. The results are summarized in Table 1. The table shows the convergence history for the different approaches, i.e., the iteration number at which the poles were computed. Note that poles appear in pairs, since $\lambda_{2j} = \overline{\lambda_{2j-1}}$, $j = 1, \ldots, 7$. We draw the following conclusion from Table 1. Both V1 and V3 solve the problem, whereas variant V2 fails. This indicates the importance of the deflation in the *selection step* (line 8) in the algorithm. The deflation does apparently not play an important role in the expansion step (lines 4,5) as similar (or even better) properties are observed when using the original (non-deflated) $\mathbf{f}$ and $\mathbf{d}$.

6 Conclusions and outlook

In this paper, we modified the dominant pole algorithm, which was initially designed for linear problems, to solve nonlinear systems whose transfer function has an infinite number of simple poles. We applied the algorithm to a large system with time-delay. The major novelty was the introduction of a deflation strategy. We showed that the deflation strategy works well for the problem that we solved. We illustrated the important properties of the deflation strategy: the residues of the poles change, but converge to the original residues after a number of iterations.

Finally, we wish to comment on generality of the method. Extensions of the algorithm to other non-linear problems are possible. For any non-linear system for which a modal expansion of the form (2.6) exists, and the projected eigenvalue problem (3.2) can be solved, the dominant pole algorithm could be considered as a method for model order reduction.

Acknowledgements

This paper presents research results of the Belgian Network DYSCO (Dynamical Systems, Control, and Optimization), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State Science Policy Office (IAP P6- DYSCO). The research is also partially funded by the Research Council K.U.Leuven grants CoE EF/05/006 (Optimization in Engineering), DBOF/08/051, OT/10/038 (Multi-parameter model order reduction and its applications) and STRT1- 09/33 of the K.U.Leuven Research Council and the project G.0712.11N of the Research Foudation - Flanders (FWO). The scientific responsibility rests with its author(s).

The first author is in debt to Prof. Wim Desmet for his suggestions and nice discussions.

References

[1] A.C. Antoulas, *Approximation of large-scale dynamical systems*, SIAM, Philadelphia, PA, USA, 2005.

[2] R. Bellman and K. L. Cooke, *Differential-Difference Equations*, Academic Press, 1963.

[3] P. Benner, V. Mehrmann, and D.C. Sorensen (eds.), *Dimension reduction of large-scale systems*, Springer-Verlag, Berlin, Heidelberg, 2005.

[4] G. Berkooz, P. Holmes, and J. Lumley, *The proper orthogonal decomposition in the analysis of turbulent flows*, Annual Review of Fluid Mechanics **25** (1993), 539–575.

[5] L. H. Bezerra, *Written discussion to [16]*, IEEE Trans. Power Syst **30** (2008), no. 4, 2137–2157.

[6] C. Chaturantabut and D. C. Sorensen, *Discrete empirical interpolation for nonlinear model reduction*, 2009.

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

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

[9] I. Gohberg and L. Lerer, *Resultants of matrix polynomials*, Bull. Am. Math. Soc. **82** (1976), 565–567.

[10] J. Hale and S. M. Verduyn Lunel, *Introduction to functional differential equations*, Springer-Verlag, 1993.

[11] P. Holmes, J. Lumley, and G. Berkooz, *Turbulence, coherent structures, dynamical systems and symmetry*, Cambrige University Press, Cambrige, UK, 1996.

[12] H. Y. Hu, E. H. Dowell, and L. N. Virgin, *Stability estimation of high dimensional vibrating systems under state delay feedback control*, Journal of Sound and Vibration **214** (1998), no. 3, 497–511.

[13] E. Jarlebring, K. Meerbergen, and W. Michiels, *A Krylov method for the delay eigenvalue problem*, SIAM Journal on Scientific Computing **32** (2010), no. 6, 3278–3300.

[14] P. Lancaster, *Lambda-matrices and vibrating systems*, Mineola, NY: Dover Publications, 2002.

[15] P. Mäkilä and J. Partington, *Laguerre and Kautz shift approximations of delay systems*, Int. J. Control **72** (1999), no. 10, 932–946.

[16] 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** (1996), 162–170.

[17] W. Michiels, Jarlebring. E., and K. Meerbergen, *Krylov based model order reduction of time-delay systems*, Technical Report TW568, Dept. Comp. Sci., KU Leuven, June 2010, Submitted for publication.

[18] W. Michiels and S.-I. Niculescu, *Stability and stabilization of time-delay systems. An eigenvalue based approach*, SIAM, Philadelphia, PA, USA, 2007.

[19] K. Mohaghegh, M. Striebel, J. ter Maten, and R. Pulch, *Nonlinear model order reduction based on trajectory piecewise linear approach: comparing different linear cores*, Scientific Computing in Electrical Engineering SCEE 2008 (J. Roos and L. R. G. Costa, eds.), vol. 14, Springer, 2010, pp. 563–570.

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

[21] A. Odabasioglu, M. Celik, and L. Paganini, *Prima: Passive reduced-order interconnect macromodeling algorithm*, IEEE TCAD of Integ. Circuits and Systems **17** (1998), no. 8, 645–654.

[22] M. J. Rewieński and J. White, *A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices*, IEEE Trans. CAD Int. Circ. Syst. **22** (2003), no. 2, 155–170.

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

[24] ———, *Efficient computation of transfer function dominant poles using subspace acceleration*, IEEE Trans. Power Syst. **21** (2006), 1218–1226.

[25] J. Rommes and N. Martins, *Computing transfer function dominant poles of large-scale second-order dynamical systems*, SIAM J. Sci. Comput. **30** (2008), no. 4, 2137–2157.

[26] 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** (2008), 346–363.

[27] G.L.G. Sleijpen and H.A. van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM Journal on Matrix Analysis and Applications **17** (1996), 401–425.

[28] J. Smith, J. Hauer, and D. Trudnowski, *Transfer function identification in power system applications*, IEEE Trans. Power Systems **8** (1993), no. 3, 1282–1290.

[29] Dmitry Vasilyev, Michal Rewieński, and Jacob White, *A TBR-based Trajectory Piecewise-Linear Algorithm for Generating Accurate Low-order Models for Nonlinear Analog Circuits and MEMS*, Design Automation Conference, June 2003, pp. 490–495.