An inner-outer nonlinear programming approach for constrained quadratic matrix model updating*

M. Andretta † E. G. Birgin‡ M. Raydan§

September 1, 2014

Abstract

The Quadratic Finite Element Model Updating Problem (QFEMUP) concerns with updating a symmetric second-order finite element model so that it remains symmetric and the updated model reproduces a given set of desired eigenvalues and eigenvectors by replacing the corresponding ones from the original model. Taking advantage of the special structure of the constraint set, it is first shown that the QFEMUP can be formulated as a suitable constrained nonlinear programming problem. Using this formulation, a method based on successive optimizations is then proposed and analyzed. To avoid that spurious modes (eigenvectors) appear in the frequency range of interest (eigenvalues) after the model has been updated, additional constraints based on a quadratic Rayleigh quotient are dynamically included in the constraint set. A distinct practical feature of the proposed method is that it is implementable computing only a few eigenvalues and eigenvectors of the associated quadratic matrix pencil. The results of our numerical experiments on illustrative problems show that the algorithm works well in practice.

Keywords: Quadratic matrix model updating, quadratic Rayleigh quotient, nonlinear programming, algorithms.

1 Introduction

The Quadratic Finite Element Model Updating Problem (QFEMUP) concerns with updating a finite-element generated model of a vibrating structure of the form:

$$M\ddot{x}(t) + D\dot{x}(t) + Kx(t) = 0, \tag{1}$$

*This work was supported by PRONEX-CNPq/FAPERJ (E-26/111.449/2010-APQ1), FAPESP (grants 2010/10133-0, 2013/03447-6, 2013/05475-7, and 2013/07375-0), and CNPq (PVE 71/2013, proj. 400926/2013-0).

†Department of Applied Mathematics and Statistics, Institute of Mathematical and Computer Sciences, University of São Paulo, Av. Trabalhador São-carlense, 400, Centro, 13566-590, São Carlos, SP, Brazil. e-mail: andretta@icmc.usp.br

‡Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090, São Paulo, SP, Brazil. e-mail: egbirgin@ime.usp.br

§Departamento de Cómputo Científico y Estadística, Universidad Simón Bolívar, Ap. 89000, Caracas 1080-A, Venezuela. e-mail: mraydan@usb.ve

where M, D, and K are real $n \times n$ matrices known as mass, damping, and stiffness, respectively; and $\dot{x}(t)$ and $\ddot{x}(t)$ denote the first and second derivatives of the time-dependent vector $x(t)$. The eigenvalues of the associated quadratic pencil

$$Q(\lambda) = \lambda^2 M + \lambda D + K \tag{2}$$

are related to natural frequencies and the eigenvectors are the mode shapes of the vibrating system (1) (see, e.g., [17, 18, 31]). The quadratic pencil (2) has $2n$ eigenvalues and $2n$ eigenvectors. The dynamics of the system are modeled by these eigenvalues and eigenvectors. For example, it is well-known that the stability of a vibrating system is determined by the nature of a few dominating natural frequencies. It is also well-known that sometimes the vibrating structures experience dangerous vibrations, called *resonance*, when a natural frequency becomes close or equal to a frequency of an external force, such as earthquake, gusty wind, weights of the human bodies, among others. Failures of many structures like buildings, bridges, airplane wings, and turbines have been attributed to resonance.

Equation (1) is usually obtained by discretization of a distributed parameter system with finite element techniques, and therefore, known as the finite element model. The matrices M, D, and K are often very large and sparse, but have some structures, such as, M is symmetric and positive definite ($M = M^T > 0$) and often diagonal, D and K are symmetric ($D = D^T$ and $K = K^T$), and the stiffness matrix K is often tridiagonal and positive semi-definite.

The QFEMUP consists in updating the quadratic pencil $Q(\lambda)$ to another quadratic pencil

$$\tilde{Q}(\lambda) = \lambda^2 M + \lambda \tilde{D} + \tilde{K} \tag{3}$$

in such a way that a small number $1 \leq p < 2n$ of given measured eigenvalues and eigenvectors from a real-life structure or an experimental structure are reproduced by the updated pencil. Besides the basic requirements of preserving the symmetry and sparsity pattern of the new matrices $\tilde{D}$ and $\tilde{K}$ and reproducing the p measured eigenvalues and eigenvectors, there are certain other engineering issues that must be taken into account while solving the problem in practice. For instance, it is important that the new matrices $\tilde{D}$ and $\tilde{K}$ are as close as possible to the original ones D and K, respectively, which imposes an optimization approach. It is also very important that no spurious modes appear in the frequency range of interest after the model has been updated; see [27]. The so called no spill-over constraint which, assuming that the p eigenpairs to be replaced are known, forces the additional $2n - p$ eigenvalues and corresponding eigenvectors to remain unchanged, clearly guarantees that no spurious modes will appear in the frequency range of interest. Several iterative numerical schemes have been recently proposed to accomplish all the mentioned requirements, including the no spill-over constraint, for several different scenarios; see, e.g., [5, 11, 12, 13, 14, 15, 16, 20, 21, 24, 25, 31, 32, 33] and references in there. In most cases, the no spill-over constraint is accomplished by using some clever linear algebra theoretical results that involve the solution of several large-scale Lyapunov, Sylvester, and block linear systems per iteration.

In this paper, we present a new optimization approach that not only maintains the symmetry, the sparsity structure, and the nearness of the matrices $\tilde{D}$ and $\tilde{K}$, while reproducing the p measured eigenvalues and eigenvectors, but also pays special attention to the fundamental engineering requirement of making sure that no spurious modes appear in the frequency range

of interest. In this work, we accomplish all these requirements without forcing the no spill-over constraint. For that, our new scheme combines an optimization procedure with the dynamical inclusion of additional constraints in an inner-outer iterative scheme. The additional constraints are based on a suitable recent extension of the Rayleigh quotient for quadratic eigenvalue problems [29, 38]. A key practical feature of the proposed scheme is that it is implementable computing only a few additional eigenvalues and eigenvectors of the associated quadratic matrix pencil. Variations of finite element model updating problems, with different levels of difficulty, have been solved in the past using numerical optimization techniques of several types; see, e.g., [1, 5, 6, 7, 11, 19, 33, 36].

The rest of the paper is organized as follows. In Section 2, we formulate the QFEMUP as a constrained optimization problem and describe the variables and the constrains, including the way of forcing the matrices' sparsity structure and symmetry. In Section 3, we describe the suitable use of the Rayleigh quotient for quadratic eigenvalue problems for building the constraints or cuts, to avoid when necessary the presence of spurious modes in the frequency range of interest. We also describe in detail the inner-outer iterative scheme, and discuss its theoretical properties. In Section 4, we show the performance of our scheme on some illustrative examples. Concluding remarks are presented in Section 5.

2 Mathematical programming formulation

Consider the quadratic pencil $Q(\lambda)$ given by (2), where $M, D, K \in \mathbb{R}^{n\times n}$ are given matrices such that M is symmetric positive definite and D and K are symmetric. Let $1 \leq p < 2n$, $\lambda_i \in \mathbb{C}$, and $x_i \in \mathbb{C}^n$ be such that (λ_i, x_i) for $i = 1, \ldots, p$ are the desired eigenpairs. The goal is to find matrices $\tilde{D}, \tilde{K} \in \mathbb{R}^{n\times n}$ such that (λ_i, x_i) are eigenpairs of the updated quadratic eigenpencil $\tilde{Q}(\lambda)$ given by (3), i.e.,

$$(\lambda_i^2 M + \lambda_i \tilde{D} + \tilde{K})x_i = 0,\ i = 1, \ldots, p. \tag{4}$$

Matrices $\tilde{D} = (\tilde{d}_{ij})$ and $\tilde{K} = (\tilde{k}_{ij})$ must be symmetric and must preserve the sparsity pattern of $D = (d_{ij})$ and $K = (k_{ij})$, respectively. In addition, $\tilde{D}$ and $\tilde{K}$ must be as close as possible to D and K, respectively.

Let $\hat{I}_D$ be a given set of pairs of the form (i, j) with $i \leq j$ such that $d_{ij} = 0$ and such that those null elements are required to be preserved in $\tilde{D}$, i.e., $\hat{I}_D$ represents the null elements (sparsity pattern) in the upper triangle of matrix D that must be preserved in $\tilde{D}$. Analogously, let $\hat{I}_K$ be the indices of the null elements of the upper triangle of K to be preserved, and let

$$I_D = \{(i, j) \mid 1 \leq i \leq j \leq n \text{ such that } (i, j) \notin \hat{I}_D\}$$

and

$$I_K = \{(i, j) \mid 1 \leq i \leq j \leq n \text{ such that } (i, j) \notin \hat{I}_K\}.$$

Therefore, elements $\tilde{d}_{ij}$ with $(i, j) \in I_D$ are the unknown elements or variables of the desired matrix $\tilde{D}$. For the remaining elements of $\tilde{D}$ we have that: (a) if $i > j$ and $(j, i) \in I_D$ then $\tilde{d}_{ij} = \tilde{d}_{ji}$ in order to preserve symmetry, and (b) if neither (i, j) nor (j, i) are in I_D then $\tilde{d}_{ij} = \tilde{d}_{ji} = 0$ in order to preserve the desired sparsity pattern. Similarly, the unknowns related to the matrix $\tilde{K}$ are $\tilde{k}_{ij}$ with $(i, j) \in I_K$. It is assumed that a non-negative real number U such that $-U \leq \tilde{d}_{ij} \leq U$ for all $(i, j) \in I_D$ and $-U \leq \tilde{k}_{ij} \leq U$ for all $(i, j) \in I_K$ is known.

Summing up, the mathematical programming formulation of the QFEMUP is given by

$$\begin{array}{ll} \text{Minimize} & \|\tilde{D}-D\|_F^2+\|\tilde{K}-K\|_F^2 \\ \text{subject to} & MX\Lambda^2+\tilde{D}X\Lambda+\tilde{K}X=0, \\ & -U\leq \tilde{d}_{ij}\leq U \ \text{ for all } \ (i,j)\in I_D, \\ & -U\leq \tilde{k}_{ij}\leq U \ \text{ for all } \ (i,j)\in I_K, \end{array} \tag{5}$$

where $\Lambda=\mathrm{diag}(\lambda_1,\ldots,\lambda_p)\in\mathbb{C}^{p\times p}$ and $X\in\mathbb{C}^{n\times p}$ has columns $x_1,\ldots,x_p$.

Note that in (5) the variables are $\tilde{d}_{ij}$ with $(i,j)\in I_D$ and $\tilde{k}_{ij}$ with $(i,j)\in I_K$. Hence the number of variables is $|I_D|+|I_K|$, and the sparsity pattern and symmetry of D and K are preserved because

$$\tilde{d}_{ij}=\begin{cases} 0, & \text{if } i\leq j \text{ and } (i,j)\notin I_D, \\ \tilde{d}_{ji}, & \text{if } i>j, \end{cases} \quad \text{and} \quad \tilde{k}_{ij}=\begin{cases} 0, & \text{if } i\leq j \text{ and } (i,j)\notin I_K, \\ \tilde{k}_{ji}, & \text{if } i>j. \end{cases}$$

This means that the symmetry and the sparsity pattern of $\tilde{D}$ and $\tilde{K}$ are guaranteed by a clever choice of the variables, and that those requirements do not need to be considered as explicit constraints, thus reducing the number of constraints as well. Moreover, note that in the objective function in (5) we have

$$\|\tilde{D}-D\|_F^2=\sum_{(i,i)\in I_D}(\tilde{d}_{ii}-d_{ii})^2+2\sum_{\substack{(i,j)\in I_D\\ i\neq j}}(\tilde{d}_{ij}-d_{ij})^2,$$

since the remaining terms are null, and, similarly,

$$\|\tilde{K}-K\|_F^2=\sum_{(i,i)\in I_K}(\tilde{k}_{ii}-k_{ii})^2+2\sum_{\substack{(i,j)\in I_K\\ i\neq j}}(\tilde{k}_{ij}-k_{ij})^2.$$

Therefore, since the number of variables, the number of constraints, and the computational complexity of evaluating the objective function and the constraints in (5) are of the order of the number of non-null entries in M, D, and K, the model (5) is suitable for solving potentially large-sized instances of the QFEMUP.

The mathematical programming formulation (5) is equivalent (in the sense of providing the same solution) to the approach based on alternating projection methods introduced in [36], the novelty of the former being the possibility of dealing in an efficient way with the requirement of avoiding spurious modes in the frequency range of interest after the model has been updated, as will be described in Section 3.

A remark on the way we tackled the requirement "having the new matrices $\tilde{D}$ and $\tilde{K}$ as close as possible to the original ones D and K" is in order. The objective function in (5) minimizes the Frobenius norm of the difference between $(\tilde{D},\tilde{K})$ and (D,K). The squaring of the Frobenius norm has no effect in this case other than providing differentiability of the objective function. However, having $(\tilde{D},\tilde{K})$ as close as possible to (D,K) is not exactly the same as having $\tilde{D}$ as close as possible to D and *at the same time* $\tilde{K}$ as close as possible to K. This bi-objective

problem (see, for example, [35]) would be much more difficult to be solved (since computing the Pareto frontier [35] would be needed). An alternative would be to minimize the sum of those distances, i.e., $\|\tilde{D} - D\| + \|\tilde{K} - K\|$, where $\|\cdot\|$ is an arbitrary norm. Note that squaring the norms would transform the problem into a different problem and not squaring the norms would make, for example, the case in which the Frobenius norm is considered, a non-differentiable problem. Another alternative would be to minimize the largest between $\|\tilde{D} - D\|$ and $\|\tilde{K} - K\|$ for any arbitrary norm, i.e.,

$$\begin{array}{lrcl} \text{Minimize} & z & & \\ \text{subject to} & \|\tilde{D} - D\| & \leq & z, \\ & \|\tilde{K} - K\| & \leq & z, \\ & MX\Lambda^2 + \tilde{D}X\Lambda + \tilde{K}X & = & 0, \\ & -U \leq \tilde{d}_{ij} \leq U \text{ for all } (i,j) & \in & I_D, \\ & -U \leq \tilde{k}_{ij} \leq U \text{ for all } (i,j) & \in & I_K. \end{array} \tag{6}$$

If the Frobenius norm is considered in (6), squaring both sides in the first two inequalities does not alter the problem and it becomes continuous and differentiable. All these alternatives correspond to different interpretations of the nearness requirement and they probably have different solutions. In the present work we consider the objective function in (5), which is the usual interpretation of the nearness requirement considered in the literature (see, for example, [27] and the references therein).

Finally, a remark on the hyperplane constraint in (5) is also in order. Note that $\lambda_i \in \mathbb{C}$ and $x_i \in \mathbb{C}^n$ for $i = 1, \ldots, p$. Therefore, to deal with regular nonlinear programming solvers, that handle problems in the real (non-complex) space, it would be adequate to re-write the constraint as:

$$\begin{aligned} \Re(MX\Lambda^2 + \tilde{D}X\Lambda + \tilde{K}X) &= 0, \\ \Im(MX\Lambda^2 + \tilde{D}X\Lambda + \tilde{K}X) &= 0, \end{aligned}$$

where $\Re(c)$ and $\Im(c)$ represent the real part a and the imaginary part b of a complex number $c = a + bi$, arriving to the formulation

$$\begin{array}{lrcl} \text{Minimize} & \|\tilde{D} - D\|_F^2 + \|\tilde{K} - K\|_F^2 & & \\ \text{subject to} & \Re(MX\Lambda^2 + \tilde{D}X\Lambda + \tilde{K}X) & = & 0, \\ & \Im(MX\Lambda^2 + \tilde{D}X\Lambda + \tilde{K}X) & = & 0, \\ & -U \leq \tilde{d}_{ij} \leq U \text{ for all } (i,j) & \in & I_D, \\ & -U \leq \tilde{k}_{ij} \leq U \text{ for all } (i,j) & \in & I_K. \end{array} \tag{7}$$

Note that (7) is a continuous and differentiable nonlinear programming problem for which any off-the-shelf nonlinear programming method may be applied.

3 Nonlinear cuts and algorithmic framework

In this section, we consider the extra requirement of avoiding that spurious eigenvalues appear in the updated quadratic eigenpencil (3). For simplicity of exposition, we will focus on the frequent situation in which $\hat{\lambda} \in \mathbb{R}$ is given (the reader can think of a "small" and negative value for $\hat{\lambda}$ but any value is possible) and there is a constraint that says that all eigenvalues of the updated eigenpencil (3) must have their real part less than or equal to $\hat{\lambda}$.

Let us assume that $(\mu_1, y_1), \ldots, (\mu_{2n}, y_{2n})$ are the (unknown) eigenpairs of the original quadratic eigenpencil (2) and that $1 \leq p < 2n$ and $(\lambda_1, x_1), \ldots, (\lambda_p, x_p)$ are the desired eigenpairs. In addition, let us assume that the p eigenpairs that must be substituted *are known* and that (without loss of generality) those eigenpairs are the first p eigenpairs of (2), i.e., $(\mu_1, y_1), \ldots, (\mu_p, y_p)$. In what follows we will also assume that

(a) the p desired eigenpairs are such that $\Re(\lambda_i) \leq \hat{\lambda}$ for $i = 1, \ldots, p$,

(b) the p eigenpairs that will be substituted are such that $\Re(\mu_i) > \hat{\lambda}$ for $i = 1, \ldots, p$,

(c) the remaining $2n - p$ eigenpairs are such that $\Re(\mu_i) \leq \hat{\lambda}$ for $i = p + 1, \ldots, 2n$.

The so called no spill-over constraint forces the $2n - p$ eigenpairs $(\mu_{p+1}, y_{p+1}), \ldots, (\mu_{2n}, y_{2n})$ to remain unchanged and, therefore, the updated quadratic eigenpencil (3) will have all its eigenvalues with their real part less than or equal to $\hat{\lambda}$ as desired. The methodologies, already developed, that preserve the *unknown* $2n - p$ eigenpairs $(\mu_{p+1}, y_{p+1}), \ldots, (\mu_{2n}, y_{2n})$ of the original quadratic eigenpencil (2) as eigenpairs of the updated quadratic eigenpencil (3) certainly avoids the advent of spurious eigenvalues in the updated quadratic eigenpencil, at the price of solving several large-scale Lyapunov, Sylvester, and block linear systems per iteration.

In the present work, we address the extra requirement of avoiding that spurious eigenvalues appear in the updated quadratic eigenpencil (3) using a different approach. Solving the mathematical programming problem (7), we find matrices $\tilde{D}$ and $\tilde{K}$ such that the updated quadratic eigenpencil (3) has the desired eigenpairs $(\lambda_1, x_1), \ldots, (\lambda_p, x_p)$ for $i = 1, \ldots, p$, i.e., such that (4) holds. To achieve this goal, none of the eigenpairs $(\mu_1, y_1), \ldots, (\mu_{2n}, y_{2n})$ of the original quadratic eigenpencil (2) are assumed to be known. It is the nearness requirement, expressed in the minimization of the distance between $(\tilde{D}, \tilde{K})$ and (D, K), that helps to safeguard as much as possible the eigenpairs of the original quadratic eigenpencil. Then, an eigenvalue with largest real part of the updated quadratic eigenpencil (3) is computed. If its real part is larger than $\hat{\lambda}$, then a (normalized) associated eigenvector $\tilde{x}$ is computed; a constraint, that depends on $\tilde{x}$, is added to the mathematical programming model (7) with the attempt of avoiding the detected spurious eigenvalue; and a new nonlinear programming problem is solved. This iterative process is repeated until the updated quadratic eigenpencil has no spurious eigenvalues. The nature of the constraint or cut that is added to the mathematical programming model is discussed below. The idea of generating additional constraints or cuts from standard (non-quadratic) eigenvectors was used to solve Lyapunov equations by Geromel [28] and to solve constrained least-squares matrix problems by Hu [30].

Assume that an eigenvalue $\tau \in \mathbb{C}$ with largest real part of the updated eigenpencil (3) has been computed and that $\Re(\tau) > \hat{\lambda}$. Then, a normalized associated eigenvector $\tilde{x} \in \mathbb{C}^n$ is computed. From the Galerkin condition (see [29, 38]), it follows that τ must be one of the two

solutions of the quadratic equation

$$q(\theta) = \theta^2(\tilde{x}^* M \tilde{x}) + \theta(\tilde{x}^* \tilde{D} \tilde{x}) + (\tilde{x}^* \tilde{K} \tilde{x}),$$

that are given by

$$\theta_1 = \frac{-(\tilde{x}^* \tilde{D} \tilde{x}) + \sqrt{(\tilde{x}^* \tilde{D} \tilde{x})^2 - 4(\tilde{x}^* M \tilde{x})(\tilde{x}^* \tilde{K} \tilde{x})}}{2(\tilde{x}^* M \tilde{x})}$$

and

$$\theta_2 = \frac{-(\tilde{x}^* \tilde{D} \tilde{x}) - \sqrt{(\tilde{x}^* \tilde{D} \tilde{x})^2 - 4(\tilde{x}^* M \tilde{x})(\tilde{x}^* \tilde{K} \tilde{x})}}{2(\tilde{x}^* M \tilde{x})}.$$

If $\tau = \theta_1$ then the new constraint is given by

$$\Re \left(\frac{-(\tilde{x}^* \tilde{D} \tilde{x}) + \sqrt{(\tilde{x}^* \tilde{D} \tilde{x})^2 - 4(\tilde{x}^* M \tilde{x})(\tilde{x}^* \tilde{K} \tilde{x})}}{2(\tilde{x}^* M \tilde{x})} \right) \leq \hat{\lambda} - \varepsilon, \tag{8}$$

where $\varepsilon > 0$ is a given small tolerance, which is subtracted from $\hat{\lambda}$ to slightly overstate what we want to achieve. Otherwise, we have that $\tau = \theta_2$ and then the new constraint is given by

$$\Re \left(\frac{-(\tilde{x}^* \tilde{D} \tilde{x}) - \sqrt{(\tilde{x}^* \tilde{D} \tilde{x})^2 - 4(\tilde{x}^* M \tilde{x})(\tilde{x}^* \tilde{K} \tilde{x})}}{2(\tilde{x}^* M \tilde{x})} \right) \leq \hat{\lambda} - \varepsilon. \tag{9}$$

Recall that when solving problem (7) with the additional constraint (8) or the additional constraint (9), we have that the vector $\tilde{x}$ is given, as well as the matrix M, and so the variables are the elements $\tilde{d}_{ij}$ of $\tilde{D}$ with $(i,j) \in I_D$ and the elements $\tilde{k}_{ij}$ of $\tilde{K}$ with $(i,j) \in I_K$. The complete iterative procedure is described below.

Algorithm 3.1. Let $M, D, K \in \mathbb{R}^{n \times n}$ be given matrices such that M is positive definite and $D = (d_{ij})$ and $K = (k_{ij})$ are symmetric. Let $1 \leq p < 2n$ and let $(\lambda_i, x_i) \in \mathbb{C} \times C^n$ for $i = 1, \ldots, p$ be the desired quadratic eigenpairs. Let $\hat{\lambda} \in \mathbb{R}$ be a parameter that describes the forbidden region for the quadratic eigenvalues of the updated quadratic eigenpencil. Let $U > 0$ be a given large real number and let $\varepsilon > 0$ be a given small tolerance. Set $\kappa \leftarrow 0$.

Step 1. *Set the sparsity patterns*

Compute

$$I_D = \{(i,j) \mid 1 \leq i \leq j \leq n \text{ such that } d_{ij} \neq 0\}$$

and

$$I_K = \{(i,j) \mid 1 \leq i \leq j \leq n \text{ such that } k_{ij} \neq 0\}.$$

Step 2. *Optimization step*

By solving the nonlinear programming problem with $|I_D| + |I_K|$ variables given by

$$\begin{array}{lrcl} \text{Minimize} & \|\tilde{D} - D\|_F^2 + \|\tilde{K} - K\|_F^2 & & \\ \text{subject to} & \Re(MX\Lambda^2 + \tilde{D}X\Lambda + \tilde{K}X) & = & 0, \\ & \Im(MX\Lambda^2 + \tilde{D}X\Lambda + \tilde{K}X) & = & 0, \\ & -U \le \tilde{d}_{ij} \le U \text{ for all } (i,j) & \in & I_D, \\ & -U \le \tilde{k}_{ij} \le U \text{ for all } (i,j) & \in & I_K, \end{array} \tag{10}$$

plus

$$\Re\left(\frac{-(u_j^*\tilde{D}u_j) + s_j\sqrt{(u_j^*\tilde{D}u_j)^2 - 4(u_j^*Mu_j)(u_j^*\tilde{K}u_j)}}{2(u_j^*Mu_j)}\right) \le \hat{\lambda} - \varepsilon,\ j = 0, \ldots, \kappa - 1, \tag{11}$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_p) \in \mathbb{C}^{p\times p}$ and $X \in \mathbb{C}^{n\times p}$ has columns $x_1, \ldots, x_p$, find matrices $\tilde{D}_\kappa$ and $\tilde{K}_\kappa$ such that

$$(\lambda_i^2 M + \lambda_i \tilde{D}_\kappa + \tilde{K}_\kappa)x_i = 0,\ i = 1, \ldots, p.$$

Step 3. *Check for spurious quadratic eigenvalues*

Step 3.1. Compute a quadratic eigenvalue τ with largest real part of the updated quadratic eigenpencil

$$\tilde{Q}(\lambda) = \lambda^2 M + \lambda\tilde{D}_\kappa + \tilde{K}_\kappa.$$

Step 3.2. If $\Re(\tau) \le \hat{\lambda}$ then stop returning $\tilde{D}_\kappa$ and $\tilde{K}_\kappa$.

Step 4. *Add a new cut and iterate*

Step 4.1. Compute a quadratic eigenvector u_κ associated with τ (such that $\|u_\kappa\|_2 = 1$).

Step 4.2. If

$$\tau = \frac{-(u_\kappa^*\tilde{D}_\kappa u_\kappa) + \sqrt{(u_\kappa^*\tilde{D}_\kappa u_\kappa)^2 - 4(u_\kappa^*Mu_\kappa)(u_\kappa^*\tilde{K}_\kappa u_\kappa)}}{2(u_\kappa^*Mu_\kappa)}$$

then set $s_\kappa = 1$. Otherwise, if

$$\tau = \frac{-(u_\kappa^*\tilde{D}_\kappa u_\kappa) - \sqrt{(u_\kappa{}^*\tilde{D}_\kappa u_\kappa)^2 - 4(u_\kappa^*Mu_\kappa)(u_\kappa^*\tilde{K}_\kappa u_\kappa)}}{2(u_\kappa^*Mu_\kappa)}$$

then set $s_\kappa = -1$.

Step 4.3. Set $\kappa \leftarrow \kappa + 1$ and go to Step 2.

Remark. Notice that, for any iteration κ, the feasible region of the nonlinear programming problem solved at Step 2 is a closed subset of the closed and bounded set $B_D \times B_K$, where

$$B_D = \{D \in \mathbb{R}^{n\times n} |\ D = D^T, -U \leq d_{ij} \leq U \ \text{ if } \ (i,j) \in I_D, \ \text{and } d_{ij} = 0 \ \text{ if } \ (i,j) \in \hat{I}_D\}$$

and

$$B_K = \{K \in \mathbb{R}^{n\times n} |\ K = K^T, -U \leq k_{ij} \leq U \ \text{ if } \ (i,j) \in I_K, \ \text{and } k_{ij} = 0 \ \text{ if } \ (i,j) \in \hat{I}_K\}.$$

Notice also that at each iteration $\kappa \geq 1$ all the cuts in (11) from previous iterations are kept as constraints for the current nonlinear programming problem. Therefore, the feasible region is monotonically reduced when κ increases. Our next theorem establishes that Algorithm 3.1 terminates in a finite number of iterations.

Theorem 3.1 *Let $\hat{\lambda} \in \mathbb{R}$ and $\varepsilon > 0$ be given. If Algorithm 3.1 is applied to solve the optimization problem (7) with the additional constraint that all eigenvalues of the updated eigenpencil (3) must have their real part less than or equal to $\hat{\lambda}$, then it terminates in a finite number of iterations.*

Proof. Let us consider the functions $\theta_1 : B_D \times B_K \times S_u \to \mathbb{R}$ and $\theta_2 : B_D \times B_K \times S_u \to \mathbb{R}$, where $S_u = \{u \in \mathbb{C}^n |\ \|u\|_2 = 1\}$ is the unit sphere in $\mathbb{C}^n$, that define the possible cuts in (11)

$$\theta_1(\tilde{D}, \tilde{K}, u) = \Re\left(\frac{-(u^*\tilde{D}u) + \sqrt{(u^*\tilde{D}u)^2 - 4(u^*Mu)(u^*\tilde{K}u)}}{2(u^*Mu)}\right)$$

and

$$\theta_2(\tilde{D}, \tilde{K}, u) = \Re\left(\frac{-(u^*\tilde{D}u) - \sqrt{(u^*\tilde{D}u)^2 - 4(u^*Mu)(u^*\tilde{K}u)}}{2(u^*Mu)}\right).$$

Since M is symmetric and positive definite, and $\|u\|_2 = 1$, then $u^*Mu \geq \lambda_{\min}(M) > 0$, where $\lambda_{\min}(M) > 0$ is the smallest eigenvalue of M. Hence, in θ_1 and θ_2 the denominator is uniformly bounded away from zero, and as a consequence both functions are continuous on $B_D \times B_K \times S_u$. Moreover, since $B_D \times B_K \times S_u$ is a compact set, then the functions θ_1 and θ_2 are uniformly continuous on $B_D \times B_K \times S_u$. Therefore, for the given $\varepsilon > 0$, there exists $\delta_1 > 0$ such that

$$\|[\tilde{D}' : \tilde{K}' : u'] - [\tilde{D} : \tilde{K} : u]\|_F < \delta_1 \tag{12}$$

implies that

$$|\theta_1(\tilde{D}', \tilde{K}', u') - \theta_1(\tilde{D}, \tilde{K}, u)| < \varepsilon, \tag{13}$$

for all $(\tilde{D}', \tilde{K}', u')$ and $(\tilde{D}, \tilde{K}, u)$ in $B_D \times B_K \times S_u$. Similarly, for the given $\varepsilon > 0$ there exists $\delta_2 > 0$ such that

$$\|[\tilde{D}' : \tilde{K}' : u'] - [\tilde{D} : \tilde{K} : u]\|_F < \delta_2 \tag{14}$$

implies that

$$|\theta_2(\tilde{D}', \tilde{K}', u') - \theta_2(\tilde{D}, \tilde{K}, u)| < \varepsilon, \tag{15}$$

for all $(\tilde{D}', \tilde{K}', u')$ and $(\tilde{D}, \tilde{K}, u)$ in $B_D \times B_K \times S_u$. In here, $[\tilde{D} : \tilde{K} : u]$ denotes a block matrix which is built stacking the matrices $\tilde{D}$, $\tilde{K}$, and the vector u, i.e., it is a matrix with $2n + 1$ columns and n rows.

For each iteration κ, if the algorithm does not stop at Step 3.2, it sets $s_\kappa = 1$ or $s_\kappa = -1$. It means that exactly one of the two functions θ_1 or θ_2 is chosen to build a new cut at Step 4. Hence, Algorithm 3.1 generates two subsequences of iterations identified with $J_{\theta_1} \subset \mathbb{N}$ and $J_{\theta_2} \subset \mathbb{N}$, which are the sets of indices κ associated with the iterations for which θ_1 (or equivalently $s_\kappa = 1$) is chosen, and the indices for which θ_2 (or equivalently $s_\kappa = -1$) is chosen, respectively. Hence, once κ iterations have been performed (numbered from 0 to $\kappa - 1$), it follows that $J_{\theta_1} \cap J_{\theta_2} = \emptyset$ and $J_{\theta_1} \cup J_{\theta_2} = \{0, 1, \ldots, \kappa - 1\}$.

Let us suppose, by way of contradiction, that Algorithm 3.1 iterates infinitely many times using the function θ_1 to build the cuts, i.e., that J_{θ_1} is an infinite set of indices. In that case, Algorithm 3.1 generates a sequence $\{u_\kappa\}$ in S_u for $\kappa \in J_{\theta_1}$. Since S_u is compact then there exists an accumulation point of that sequence in S_u. Hence, for $\delta_1 > 0$ there exist u_{i_1} and u_{i_2} with $i_1 < i_2$ and $i_1, i_2 \in J_{\theta_1}$, satisfying

$$\|[\tilde{D}_{i_2} : \tilde{K}_{i_2} : u_{i_1}] - [\tilde{D}_{i_2} : \tilde{K}_{i_2} : u_{i_2}]\|_F = \|u_{i_1} - u_{i_2}\|_2 < \delta_1. \tag{16}$$

Now, since $i_1 < i_2$, matrices $\tilde{D}_{i_2}$ and $\tilde{K}_{i_2}$ computed at iteration i_2 satisfy the constraint in (11) with $j = i_1$ given by

$$\theta_1(\tilde{D}_{i_2}, \tilde{K}_{i_2}, u_{i_1}) \leq \hat{\lambda} - \varepsilon. \tag{17}$$

However, since Algorithm 3.1 does not stop at iteration i_2 (Step 3.2) and $i_2 \in J_{\theta_1}$, this means that u_{i_2} is such that

$$\theta_1(\tilde{D}_{i_2}, \tilde{K}_{i_2}, u_{i_2}) > \hat{\lambda}. \tag{18}$$

Clearly, (16), (17), and (18) contradict (12) and (13), the uniform continuity of θ_1 on $B_D \times B_K \times S_u$. Therefore the number of indices in the set J_{θ_1} is finite, say N_1.

Let us now suppose, by way of contradiction, that Algorithm 3.1 iterates infinitely many times using the function θ_2 to build the cuts, i.e., that J_{θ_2} is an infinite set of indices. Repeating the same sequence of arguments as before but now using $\delta_2 > 0$ and θ_2 instead of $\delta_1 > 0$ and θ_1, and contradicting (14) and (15), i.e., the uniform continuity of θ_2 on $B_D \times B_K \times S_u$, we conclude that the number of indices in the set J_{θ_2} is also finite, say N_2. Thus, Algorithm 3.1 terminates in a finite number of iterations $\widehat{N} = N_1 + N_2$, and the result is established. □

Notice that when Algorithm 3.1 terminates, at Step 3.2, matrices $\tilde{D}_{\widehat{N}}$ and $\tilde{K}_{\widehat{N}}$ satisfy all the constraints in (10). Moreover, the real part of all the eigenvalues λ of the quadratic eigenpencil $\lambda^2 M + \lambda \tilde{D}_{\widehat{N}} + \tilde{K}_{\widehat{N}}$ are less than or equal to $\hat{\lambda}$. Therefore, Algorithm 3.1 terminates at a feasible solution of (10) that also satisfies the extra requirement of not having spurious eigenvalues and that it is optimal for problem (10,11).

4 Numerical experiments

To give further insight into the new approach of dynamically adding quadratic Rayleigh quotient cuts to the nonlinear programming setting, for solving the QFEMUP, we present the results of

some numerical experiments. We implemented Algorithm 3.1 in Fortran 90. All tests were conducted on a computer with 4 Intel Core i7-3417U 1.9GHz processors and 4GB of RAM memory, running GNU/Linux operating system (Ubuntu 4.8.2-19ubuntu1, kernel 3.13.0-32). Codes were compiled by the GFortran Fortran compiler of GCC (version 4.8.2) with the -O3 optimization directive enabled.

At Step 2 of Algorithm 3.1, the optimization subproblems given by (10,11) were solved using the nonlinear programming solver Algencan [2, 3, 10]. Algencan version 3.0.0, available for download at the TANGO Project web page (`http://www.ime.usp.br/~egbirgin/tango/`) was considered. All parameters were used with their default values, while feasibility and optimality tolerances were both set to 10^{-4}. Algencan is an augmented Lagrangian method for nonlinear programming that solves the bound-constrained augmented Lagrangian subproblems using Gencan [8, 9, 4], an active-set method for bound-constrained minimization. The initial guess for the (iterative process of solving the) nonlinear programming subproblems is always given by the original matrices D and K.

At Step 3, a quadratic eigenvalue τ with largest real part and, when required, an associated eigenvector, are computed using subroutines `dnaupd` and `dneupd` from Arpack [34]. Arpack subroutines, based on implicitly restarted Arnoldi methods, are applied to compute an eigenvalue with largest real part of the linearization given by applying the substitution $v = \lambda x$ in $(\lambda^2 M + \lambda\tilde{D} + \tilde{K})x = 0$, that yields the generalized eigenvalue problem

$$\begin{pmatrix} 0 & I \\ -\tilde{K} & -\tilde{D} \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} - \lambda \begin{pmatrix} I & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} = 0. \tag{19}$$

Other linearizations are possible and the most adequate choice depends on the nonsingularity of M and $\tilde{K}$ and, in the large-scale case, on the sparsity structure of the matrices. See [38, pp. 252–253] for details.

The performance of Algorithm 3.1 will be illustrated by analyzing its behavior on three small and one medium-sized randomly generated numerical test examples. In all examples, we set $U = 10^{20}$ and $\varepsilon = 2 \times 10^{-4}$. The three small examples were solved in a fraction of a second, and so elapsed CPU times required to solve them are not reported.

4.1 Example 1.

In the first example, we considered the quadratic eigenpencil (2) with matrices

$$M = \begin{pmatrix} 0.7110 & 0.0212 & -0.5813 \\ 0.0212 & 0.8509 & 0.4498 \\ -0.5813 & 0.4498 & 1.7045 \end{pmatrix}, \quad D = \begin{pmatrix} 0.1167 & 0.3240 & 0.0237 \\ 0.3240 & 0.2774 & 0.6079 \\ 0.0237 & 0.6079 & 2.0967 \end{pmatrix},$$

and

$$K = \begin{pmatrix} 0.3521 & 0.0222 & 0.2350 \\ 0.0222 & -0.0007 & 0.0544 \\ 0.2350 & 0.0544 & 1.0708 \end{pmatrix}.$$

The objective is to find symmetric matrices $\tilde{D}$ and $\tilde{K} \in \mathbb{R}^{3\times 3}$ such that the modified eigenpencil (3) has $\lambda_1 = -0.1$ as quadratic eigenvalue associated with the quadratic eigenvector

$x_1 = (0.09, -1.00, 0.07)^T$. In addition, we would like that all quadratic eigenvalues of the modified eigenpencil have its real part not larger that $\hat{\lambda} = -0.1$. Matrices $\tilde{D}$ and $\tilde{K}$ should be as near as possible to D and K, respectively, and, since the given matrices are dense, there is no sparsity pattern to be preserved.

We now describe the application of Algorithm 3.1 to this example. At Step 1, we have that $I_D = I_K = \{(1,1),(1,2),(1,3),(2,2),(2,3),(3,3)\}$, meaning that the nonlinear programming subproblems at Step 2 will have $n = 12$ variables. This reflects the fact that matrices $\tilde{D}$ and $\tilde{K}$ are dense (because matrices D and K are also dense) and that, since $\tilde{D}$ and $\tilde{K}$ are symmetric, the unknowns correspond only to their upper triangle's elements. By solving the nonlinear programming subproblem (10) at Step 2 (note that there are no constraints of type (11) in this first iteration), we obtain

$$\tilde{D}_0 = \begin{pmatrix} 0.1177 & 0.3185 & 0.0248 \\ 0.3185 & 0.2745 & 0.5998 \\ 0.0248 & 0.5998 & 2.0978 \end{pmatrix} \text{ and } \tilde{K}_0 = \begin{pmatrix} 0.3420 & 0.0770 & 0.2237 \\ 0.0770 & 0.0286 & 0.1355 \\ 0.2237 & 0.1355 & 1.0593 \end{pmatrix}. \tag{20}$$

Matrices $\tilde{D}_0$ and $\tilde{K}_0$ in (20) are such that $\|D - \tilde{D}_0\|_F^2 = 0.0002$ and $\|K - \tilde{K}_0\|_F^2 = 0.0205$. At Step 3, we compute a quadratic eigenvalue τ of the modified eigenpencil with largest real part and we obtain that $\tau = -0.0712 \not\leq -0.1 = \hat{\lambda}$. Therefore, at Step 4, we compute a normalized eigenvector u_0 associated with τ given by $u_0 = (0.1106, -0.9909, 0.0762)^T$. In addition, by the test given at Step 4.2, we set $s_0 = 1$. The values u_0 and s_0 define a cut of type (11). In the next iteration ($\kappa = 1$), by solving subproblem (10,11), we obtain matrices

$$\tilde{D}_1 = \begin{pmatrix} 0.1181 & 0.3151 & 0.0251 \\ 0.3151 & 0.3053 & 0.5974 \\ 0.0251 & 0.5974 & 2.0980 \end{pmatrix} \text{ and } \tilde{K}_1 = \begin{pmatrix} 0.3420 & 0.0767 & 0.2237 \\ 0.0767 & 0.0317 & 0.1353 \\ 0.2237 & 0.1353 & 1.0593 \end{pmatrix}. \tag{21}$$

Matrices $\tilde{D}_1$ and $\tilde{K}_1$ in (21) are such that $\|D - \tilde{D}_1\|_F^2 = 0.0012$ and $\|K - \tilde{K}_1\|_F^2 = 0.0206$. These distances are only slightly larger than the distances obtained in the first iteration of the algorithm. This time, when computing a quadratic eigenvalue τ with largest real part, we verify that $\tau = -0.1 = \hat{\lambda}$. This means that we have obtained the desired modified eigenpencil with no spurious quadratic eigenvalues.

4.2 Example 2.

In the second example, we considered the quadratic eigenpencil (2) with matrices

$$M = \begin{pmatrix} 1.6312 & -0.2473 & -1.0380 & 0.4628 \\ -0.2473 & 0.9275 & -0.0052 & 0.2589 \\ -1.0380 & -0.0052 & 2.1554 & 0.1102 \\ 0.4628 & 0.2589 & 0.1102 & 0.8301 \end{pmatrix}, D = \begin{pmatrix} 1.4794 & -1.1102 & 0 & -0.2222 \\ -1.1102 & 0.3455 & 0.1237 & 0 \\ 0 & 0.1237 & 2.4643 & -0.1004 \\ -0.2222 & 0 & -0.1004 & 1.0838 \end{pmatrix},$$

and

$$K = \begin{pmatrix} 0.5875 & -0.1668 & 0 & 0 \\ -0.1668 & 0.1831 & 0.0456 & 0 \\ 0 & 0.0456 & 1.0749 & 0.3803 \\ 0 & 0 & 0.3803 & 0.5624 \end{pmatrix}.$$

The highlight in this 4×4 example is that D has an arbitrary sparsity pattern to be inherited by $\tilde{D}$, while K is tridiagonal, property that must also be inherited by $\tilde{K}$. In this case, we have

$$I_D = \{(1,1),(1,2),(1,4),(2,2),(2,3),(3,3),(3,4),(4,4)\},$$

$$I_K = \{(1,1),(1,2),(2,2),(2,3),(3,3),(3,4),(4,4)\},$$

and the nonlinear programming subproblems at Step 2 have $n = 15$ variables. The objective is to find matrices $\tilde{D}$ and $\tilde{K}$ as near as possible to D and K, respectively, and such that the updated eigenpencil (3) has the two desired eigenpairs (λ_1, x_1) and (λ_2, x_2), where $\lambda_1 = -0.1 + 0.3398i$, $x_1 = (0.5 + 0.04i, 0.8, -0.04 + 0.1i, 0.04 - 0.1i)^T$, and (λ_2, x_2) corresponds to the conjugate eigenpair. In addition, we would also like the updated eigenpencil to have all its quadratic eigenvalues with their real part not larger than $\hat{\lambda} = -0.1$.

By solving the first nonlinear programming subproblem at Step 2 (with $\kappa = 0$, i.e., with no cut constraints), we obtain matrices

$$\tilde{D}_0 = \begin{pmatrix} 1.5950 & -0.9061 & 0 & -0.1048 \\ -0.9061 & 0.7112 & -0.0282 & 0 \\ 0 & -0.0282 & 2.4953 & -0.2540 \\ -0.1048 & 0 & -0.2540 & 1.3599 \end{pmatrix} \tag{22}$$

and

$$\tilde{K}_0 = \begin{pmatrix} 0.5325 & -0.1962 & 0 & 0 \\ -0.1962 & 0.2082 & 0.0309 & 0 \\ 0 & 0.0309 & 0.9775 & 0.4915 \\ 0 & 0 & 0.4915 & 0.4374 \end{pmatrix}. \tag{23}$$

Matrices $\tilde{D}_0$ and $\tilde{K}_0$ in (22,23) are such that $\|D - \tilde{D}_0\|_F^2 = 0.4284$ and $\|K - \tilde{K}_0\|_F^2 = 0.0557$ (the sum being equal to 0.4841). When computing an eigenvalue τ with largest real part we find that $\tau = -0.0798 \not\leq -0.1 = \hat{\lambda}$. Therefore, we compute an associated normalized eigenvector given by $u_0 = (0.0008, 0.1063, -0.5433, 0.8328)^T$, and, by the test at Step 4.2, set $s_0 = 1$. Then, we solve a new nonlinear programming subproblem at Step 2 (this time with a single cut of type (11) given by u_0 and s_0) and find matrices

$$\tilde{D}_1 = \begin{pmatrix} 1.5936 & -0.9073 & 0 & -0.0807 \\ -0.9073 & 0.7109 & -0.0392 & 0 \\ 0 & -0.0392 & 2.4946 & -0.2534 \\ -0.0807 & 0 & -0.2534 & 1.3588 \end{pmatrix} \tag{24}$$

and

$$\tilde{K}_1 = \begin{pmatrix} 0.5318 & -0.1968 & 0 & 0 \\ -0.1968 & 0.2081 & 0.0312 & 0 \\ 0 & 0.0312 & 0.9915 & 0.4758 \\ 0 & 0 & 0.4758 & 0.4612 \end{pmatrix}. \tag{25}$$

This time, the quadratic eigenvalue τ with largest real part corresponds to $\tau = -0.1 \pm 0.3398i$ and, therefore, since $\Re(\tau) = -0.1 = \hat{\lambda}$, we are done. Matrices $\tilde{D}_1$ and $\tilde{K}_1$ in (24) and (25)

are such that $\|D - \tilde{D}_1\|_F^2 = 0.4454$ and $\|K - \tilde{K}_1\|_F^2 = 0.0414$. Since the sum of both squared distances is equal to 0.4868, it means that we have obtained the desired updated eigenpencil, with no spurious eigenvalues, at the price of a "very small" increase ($\approx 0.5\%$) in the objective function value. Observe that, as required, $\tilde{D}_1$ and $\tilde{K}_1$ in (24,25) are symmetric and preserve the desired sparsity pattern.

4.3 Example 3.

In the third example, we considered the quadratic eigenpencil (2) given by matrices

$$M = \begin{pmatrix} 1.9979 & 0.3890 & -0.3500 & 0.5459 \\ 0.3890 & 1.5993 & 0.2906 & -0.8680 \\ -0.3500 & 0.2906 & 1.1656 & -0.5510 \\ 0.5459 & -0.8680 & -0.5510 & 1.8281 \end{pmatrix}, D = \begin{pmatrix} 0.9727 & 0.7667 & -0.1444 & 0.3118 \\ 0.7667 & 0.0000 & 0.1213 & -0.0389 \\ -0.1444 & 0.1213 & 0.7190 & 0.3321 \\ 0.3118 & -0.0389 & 0.3321 & 1.3145 \end{pmatrix},$$

and

$$K = \begin{pmatrix} 0.4018 & 0.4055 & 0.1019 & 0.3685 \\ 0.4055 & 0.5521 & 0.2048 & 0.0112 \\ 0.1019 & 0.2048 & 0.2443 & 0.0941 \\ 0.3685 & 0.0112 & 0.0941 & 0.8133 \end{pmatrix}.$$

The desired quadratic eigenpair is given by $\lambda_1 = -0.1$ and $x_1 = (0.6, -0.6, 0.4, -0.5)^T$. In addition, we would like the modified eigenpencil (3) to have all its eigenvalues with their real part not larger than $\hat{\lambda} = -0.1$.

Since matrices D and K are dense, we have $I_D = I_K = \{(i,j) \mid 1 \leq i \leq j \leq 4\}$ and, thus, the nonlinear programming subproblems at Step 2 have $n = 20$ variables. When solving the first nonlinear programming subproblem with no cuts, we obtained matrices

$$\tilde{D}_0 = \begin{pmatrix} 0.9583 & 0.7707 & -0.1496 & 0.3114 \\ 0.7707 & 0.0063 & 0.1196 & -0.0299 \\ -0.1496 & 0.1196 & 0.7185 & 0.3282 \\ 0.3114 & -0.0299 & 0.3282 & 1.3251 \end{pmatrix} \tag{26}$$

and

$$\tilde{K}_0 = \begin{pmatrix} 0.5460 & 0.3650 & 0.1540 & 0.3721 \\ 0.3650 & 0.4888 & 0.2218 & -0.0788 \\ 0.1540 & 0.2218 & 0.2497 & 0.1332 \\ 0.3721 & -0.0788 & 0.1332 & 0.7072 \end{pmatrix}. \tag{27}$$

These matrices are such that $\|D - \tilde{D}_0\|_F^2 = 0.0006$ and $\|K - \tilde{K}_0\|_F^2 = 0.0646$. When computing a quadratic eigenvalue τ with largest real part of the associated modified eigenpencil, we observed that $\tau = 0.626 \not\leq -0.1 = \hat{\lambda}$. Therefore, we computed an associated normalized eigenvector $u_0 = (-0.5199, 0.715, -0.3242, 0.3368)^T$, and, by the test at Step 4.2, set $s_0 = 1$. When solving the second nonlinear programming subproblem, this time with the addition of the cut given by

u_0 and s_0, we obtained matrices

$$\tilde{D}_1 = \begin{pmatrix} 1.0981 & 0.5784 & -0.0624 & 0.2208 \\ 0.5784 & 0.2710 & -0.0003 & 0.0946 \\ -0.0624 & -0.0003 & 0.7729 & 0.2717 \\ 0.2208 & 0.0946 & 0.2717 & 1.3839 \end{pmatrix} \tag{28}$$

and

$$\tilde{K}_1 = \begin{pmatrix} 0.5600 & 0.3455 & 0.1627 & 0.3632 \\ 0.3455 & 0.5139 & 0.2095 & -0.0653 \\ 0.1627 & 0.2095 & 0.2550 & 0.1278 \\ 0.3632 & -0.0653 & 0.1278 & 0.7123 \end{pmatrix}. \tag{29}$$

Matrices in (28,29) are such that $\|D - \tilde{D}_1\|_F^2 = 0.2703$ and $\|K - \tilde{K}_1\|_F^2 = 0.0655$. When computing a quadratic eigenvalue $\tau = -0.047 \pm 0.4178i$ with largest real part, we observed that $\Re(\tau) = -0.047 \not\leq -0.1 = \hat{\lambda}$. A normalized eigenvector associated with the eigenvalue $-0.047+0.4178i$ is given by $u_1 = (-0.1964-0.5258i, -0.153+0.7568i, -0.1265-0.0316i, 0.2661-0.0305i)^T$. By the test at Step 4.2, we set $s_1 = -1$. This means that the next nonlinear programming subproblem to be solved had two cut constraints. By solving it, we obtained matrices

$$\tilde{D}_2 = \begin{pmatrix} 1.1721 & 0.5134 & -0.1034 & 0.2673 \\ 0.5134 & 0.4118 & 0.0858 & -0.0337 \\ -0.1034 & 0.0858 & 0.7428 & 0.2943 \\ 0.2673 & -0.0337 & 0.2943 & 1.3818 \end{pmatrix} \tag{30}$$

and

$$\tilde{K}_2 = \begin{pmatrix} 0.5662 & 0.3403 & 0.1600 & 0.3661 \\ 0.3403 & 0.5181 & 0.2128 & -0.0690 \\ 0.1600 & 0.2128 & 0.2487 & 0.1354 \\ 0.3661 & -0.0690 & 0.1354 & 0.7032 \end{pmatrix} \tag{31}$$

such that $\|D - \tilde{D}_2\|_F^2 = 0.3555$ and $\|K - \tilde{K}_2\|_F^2 = 0.072$. When computing an eigenvalue τ with largest real part associated with the modified eigenpencil given by matrices in (30,31), we obtained that $\tau = -0.1 \leq \hat{\lambda}$ and, therefore, the algorithm stopped.

It is worth noticing that the final sum of the squared distances is 0.4275, which is relatively larger than the sum of the squared distances obtained after the first iteration of the algorithm (where the nonlinear programming subproblem did not include any cut), whose value is 0.0652. This is because the matrices $\tilde{D}_0$ and $\tilde{K}_0$ in (26,27) obtained in the first iteration have, as well as the original matrices, a quadratic eigenvalue $\lambda \approx 0.6$ that is relatively far from the desired upper limit $\hat{\lambda} = -0.1$. This justifies the "relatively large modification" of the matrices, reflected in the relatively large final squared distances, when compared to the distances obtained after the first iteration of the algorithm that did not include any cut.

4.4 Example 4.

In the fourth example, we considered the quadratic eigenpencil (2) given by sparse matrices $M, D, K \in \mathbb{R}^{100\times 100}$, such that M is diagonal and positive definite and D and K are both

tridiagonal and symmetric. Matrices M, D, and K can be found in the Appendix. The two desired quadratic eigenvalues are given by $\lambda_1 = -0.3 + 0.4713i$ and its conjugate $\lambda_2 = -0.3 - 0.4713i$. The desired eigenvectors x_1 and x_2 are such that x_1 has its non-null elements given by

$$\begin{array}{rcr} [x_1]_1 & = & 0.0010 + 0.0001i \\ [x_1]_2 & = & -0.0010 + 0.0001i \\ [x_1]_{16} & = & -0.0020 + 0.0000i \\ [x_1]_{17} & = & -0.0020 + 0.0020i \\ [x_1]_{18} & = & -0.0100 + 0.0050i \\ [x_1]_{19} & = & 0.8000 + 0.0000i \\ [x_1]_{20} & = & -0.0050 - 0.0020i \\ [x_1]_{100} & = & 0.0010 + 0.0010i \end{array}$$

and x_2 is the conjugate of x_1. In addition, we would like the modified eigenpencil (3) to have all its eigenvalues with their real part not larger than $\hat{\lambda} = -0.3$. Matrices D and K are tridiagonal, so we have $I_D = I_K = \{(i,i) \mid 1 \leq i \leq 100\} \cup \{(i,i+1) \mid 1 \leq i \leq 99\}$ and, thus, the nonlinear programming subproblems at Step 2 have $n = 398$ variables.

Algorithm 3.1 required four iterations to solve this problem. The computed eigenvalue with largest real part at each iteration, as well as the squared distances of the obtained matrices, are reported in Table 1. Intermediate matrices ($\tilde{D}_0$, $\tilde{D}_1$, $\tilde{D}_2$, $\tilde{K}_0$, $\tilde{K}_1$, and $\tilde{K}_2$) and final matrices $\tilde{D}_3$ and $\tilde{K}_3$ are reported in the Appendix. In the considered computational environment, the elapsed CPU time required to solve this problem was 6.96 seconds.

κ	$\|D - \tilde{D}_\kappa\|_F^2$	$\|K - \tilde{K}_\kappa\|_F^2$	τ
0	0.5305	0.1308	$-0.2275 \pm 0.4486i$
1	0.5876	0.1367	$-0.2910 \pm 0.4472i$
2	0.5886	0.1369	$-0.2989 \pm 0.4406i$
3	0.5891	0.1370	$-0.3000 \pm 0.4713i$

Table 1: Details of the progress of Algorithm 3.1 when applied to Example 4.

5 Final remarks

We introduced a new optimization approach for solving the QFEMUP problem, that combines the use of standard nonlinear programming solvers with the dynamical inclusion of additional constraints to avoid that spurious modes appear in the frequency range of interest. These additional constraints or cuts are based on an extension of the Rayleigh quotient for quadratic eigenvalue problems. For this combination of ideas, we have presented an iterative algorithm for which we have established finite termination. A key feature, observed on some preliminary numerical experiments, is that our new machinery finds an optimal and feasible solution computing only a few eigenvalues and eigenvectors of the associated quadratic matrix pencil. For our experiments we have combined the packages Algencan for solving the constrained optimization problems, and the package Arpack for the eigenvalue-eigenvector calculations. Both packages

are easy to obtain and easy to use, however, it is worth mentioning that any other packages can be used as inner-solvers in our iterative scheme.

In our approach, the structure of the feasible region played a key role concerning practical and also theoretical issues. However, the objective function of the sequence of nonlinear programming problems was not referred in the analysis of the algorithm. Hence, our iterative combined scheme can be extended to solve some other related model updating and eigenvalue assignment problems; see e.g., [5, 11, 19, 25, 33]. Moreover, the symmetry of the involved matrices M, D, and K, does not play any special role in our optimization machinery, besides halving the required storage, and so our approach can also be adapted to solve some other problems for which these structural constraints are not required.

6 Appendix

Tables 2–6 describe the matrices related to the numerical example 4.

i	$[M]_{ii}$	i	$[M]_{ii}$	i	$[M]_{ii}$	i	$[M]_{ii}$	i	$[M]_{ii}$	i	$[M]_{ii}$
1	1.5053	18	0.4996	35	1.5666	52	0.2083	69	1.2030	86	1.4889
2	0.1538	19	0.2994	36	0.9184	53	0.4445	70	1.0488	87	0.5119
3	1.6977	20	1.8723	37	1.4335	54	1.0295	71	0.5911	88	0.1202
4	0.5716	21	1.4258	38	1.9216	55	1.5657	72	0.2764	89	0.8877
5	1.4613	22	1.3173	39	0.1249	56	0.0473	73	0.0717	90	0.8505
6	0.4390	23	1.2847	40	1.6696	57	1.7994	74	0.9783	91	0.5080
7	0.5855	24	0.8649	41	0.0548	58	1.9011	75	0.0839	92	0.7225
8	1.4777	25	1.9855	42	1.0039	59	0.3625	76	0.3570	93	0.1266
9	1.6715	26	1.2321	43	0.3965	60	1.5685	77	1.7329	94	1.8337
10	1.8623	27	0.3111	44	1.1356	61	0.5913	78	0.5870	95	1.8101
11	1.5375	28	1.1334	45	0.2888	62	0.4942	79	1.2112	96	0.3831
12	1.8198	29	1.0207	46	0.9464	63	0.9773	80	1.0747	97	1.8462
13	0.6485	30	1.9437	47	1.6218	64	1.0405	81	1.6449	98	1.3943
14	0.5811	31	0.7576	48	1.7579	65	1.1056	82	0.8369	99	0.8907
15	1.3107	32	0.9548	49	1.6852	66	0.6284	83	1.2493	100	0.3181
16	1.6592	33	0.7280	50	1.8897	67	0.2455	84	0.3631		
17	1.4239	34	0.4705	51	1.1201	68	0.2085	85	0.8329		

Table 2: Indices and elements of matrix M for example 4.

i	j	$[D]_{ij}$	$[\tilde{D}_0]_{ij}$	$[\tilde{D}_1]_{ij}$	$[\tilde{D}_2]_{ij}$	$[\tilde{D}_3]_{ij}$	i	j	$[D]_{ij}$	$[\tilde{D}_0]_{ij}$	$[\tilde{D}_1]_{ij}$	$[\tilde{D}_2]_{ij}$	$[\tilde{D}_3]_{ij}$
1	1	0.6249	0.6938	0.8678	0.8766	0.8785	41	40	0.0075	0.0075	0.0075	0.0075	0.0075
2	1	-0.0117	-0.0359	0.0442	0.0266	0.0231	41	41	0.0641	0.0641	0.0641	0.0641	0.0641
2	2	0.2183	0.1694	0.2116	0.2174	0.2169	42	41	-0.0177	-0.0177	-0.0177	-0.0177	-0.0177
3	2	-0.0746	-0.0625	-0.0617	-0.0609	-0.0608	42	42	1.3064	1.3064	1.3064	1.3064	1.3064
3	3	2.1765	2.1765	2.1765	2.1766	2.1766	43	42	-0.0266	-0.0266	-0.0266	-0.0266	-0.0266
4	3	-0.0352	-0.0352	-0.0352	-0.0352	-0.0352	43	43	0.5155	0.5155	0.5155	0.5155	0.5155
4	4	0.7032	0.7032	0.7032	0.7032	0.7032	44	43	0.0212	0.0212	0.0212	0.0212	0.0212
5	4	0.0031	0.0031	0.0031	0.0031	0.0031	44	44	1.4344	1.4344	1.4344	1.4344	1.4344
5	5	1.8362	1.8362	1.8362	1.8362	1.8362	45	44	-0.0021	-0.0021	-0.0021	-0.0021	-0.0021
6	5	0.0032	0.0032	0.0032	0.0032	0.0032	45	45	0.3655	0.3655	0.3655	0.3655	0.3655
6	6	0.5306	0.5306	0.5306	0.5306	0.5306	46	45	0.0052	0.0052	0.0052	0.0052	0.0052
7	6	0.0458	0.0458	0.0458	0.0458	0.0458	46	46	1.1749	1.1749	1.1749	1.1749	1.1749
7	7	0.7282	0.7282	0.7282	0.7282	0.7282	47	46	-0.0363	-0.0363	-0.0363	-0.0363	-0.0363
8	7	0.0368	0.0368	0.0368	0.0368	0.0368	47	47	2.0016	2.0016	2.0016	2.0016	2.0016
8	8	1.7787	1.7787	1.7787	1.7787	1.7787	48	47	0.0281	0.0281	0.0281	0.0281	0.0281
9	8	-0.1126	-0.1126	-0.1126	-0.1126	-0.1126	48	48	2.1579	2.1579	2.1579	2.1579	2.1579
9	9	2.0339	2.0339	2.0339	2.0339	2.0339	49	48	-0.0331	-0.0331	-0.0331	-0.0331	-0.0331
10	9	-0.1332	-0.1332	-0.1332	-0.1332	-0.1332	49	49	2.1106	2.1106	2.1106	2.1106	2.1106
10	10	2.2911	2.2911	2.2911	2.2911	2.2911	50	49	0.0479	0.0479	0.0479	0.0479	0.0479
11	10	-0.0345	-0.0345	-0.0345	-0.0345	-0.0345	50	50	2.4156	2.4156	2.4156	2.4156	2.4156
11	11	2.0015	2.0015	2.0015	2.0015	2.0015	51	50	-0.0204	-0.0204	-0.0204	-0.0204	-0.0204
12	11	0.0482	0.0482	0.0482	0.0482	0.0482	51	51	1.3701	1.3701	1.3701	1.3701	1.3701
12	12	2.3092	2.3092	2.3092	2.3092	2.3092	52	51	-0.0315	-0.0315	-0.0315	-0.0315	-0.0315
13	12	0.0268	0.0268	0.0268	0.0268	0.0268	52	52	0.2578	0.2578	0.2578	0.2578	0.2578
13	13	0.7812	0.7812	0.7812	0.7812	0.7812	53	52	-0.0031	-0.0031	-0.0031	-0.0031	-0.0031
14	13	-0.0074	-0.0074	-0.0074	-0.0074	-0.0074	53	53	0.5275	0.5275	0.5275	0.5275	0.5275
14	14	0.7136	0.7136	0.7136	0.7136	0.7136	54	53	0.0449	0.0449	0.0449	0.0449	0.0449
15	14	0.0111	0.0111	0.0111	0.0111	0.0111	54	54	1.2412	1.2412	1.2412	1.2412	1.2412
15	15	1.5925	1.5925	1.5925	1.5925	1.5925	55	54	0.0477	0.0477	0.0477	0.0477	0.0477
16	15	0.0602	0.0303	0.0303	0.0303	0.0303	55	55	1.9213	1.9213	1.9213	1.9213	1.9213
16	16	1.9953	1.7690	1.7690	1.7690	1.7691	56	55	-0.0049	-0.0049	-0.0049	-0.0049	-0.0049
17	16	-0.0603	-0.3164	-0.3164	-0.3164	-0.3164	56	56	0.0665	0.0665	0.0665	0.0665	0.0665
17	17	1.6940	1.6151	1.6151	1.6151	1.6151	57	56	-0.0454	-0.0454	-0.0454	-0.0454	-0.0454
18	17	-0.0276	-0.1439	-0.1439	-0.1439	-0.1439	57	57	2.1968	2.1968	2.1968	2.1968	2.1968
18	18	0.6022	0.6022	0.6022	0.6022	0.6022	58	57	0.0909	0.0909	0.0909	0.0909	0.0909
19	18	0.0063	0.0034	0.0034	0.0034	0.0034	58	58	2.3015	2.3015	2.3015	2.3015	2.3015
19	19	-0.3571	0.1797	0.1797	0.1797	0.1797	59	58	-0.0632	-0.0632	-0.0632	-0.0632	-0.0632
20	19	0.0127	0.0089	0.0089	0.0089	0.0089	59	59	0.4271	0.4271	0.4271	0.4271	0.4271
20	20	2.3512	2.3513	2.3513	2.3513	2.3513	60	59	0.0235	0.0235	0.0235	0.0235	0.0235
21	20	-0.0559	-0.0027	-0.0027	-0.0027	-0.0027	60	60	1.9162	1.9162	1.9162	1.9162	1.9162
21	21	1.7239	1.7239	1.7239	1.7239	1.7239	61	60	0.0460	0.0460	0.0460	0.0460	0.0460
22	21	-0.0371	-0.0371	-0.0371	-0.0371	-0.0371	61	61	0.7717	0.7717	0.7717	0.7717	0.7717
22	22	1.6275	1.6275	1.6275	1.6275	1.6275	62	61	0.0158	0.0158	0.0158	0.0158	0.0158
23	22	-0.0471	-0.0471	-0.0471	-0.0471	-0.0471	62	62	0.5896	0.5896	0.5896	0.5896	0.5896
23	23	1.5498	1.5498	1.5498	1.5498	1.5498	63	62	-0.0122	-0.0122	-0.0122	-0.0122	-0.0122
24	23	0.0159	0.0159	0.0159	0.0159	0.0159	63	63	1.2641	1.2641	1.2641	1.2641	1.2641
24	24	1.0451	1.0451	1.0451	1.0451	1.0451	64	63	0.0657	0.0657	0.0657	0.0657	0.0657
25	24	-0.1689	-0.1689	-0.1689	-0.1689	-0.1689	64	64	1.2396	1.2396	1.2396	1.2396	1.2396
25	25	2.5574	2.5574	2.5574	2.5574	2.5574	65	64	0.0415	0.0415	0.0415	0.0415	0.0415
26	25	0.0541	0.0541	0.0541	0.0541	0.0541	65	65	1.2803	1.2803	1.2803	1.2803	1.2803
26	26	1.5564	1.5564	1.5564	1.5564	1.5564	66	65	0.0088	0.0088	0.0088	0.0088	0.0088
27	26	0.0366	0.0366	0.0366	0.0366	0.0366	66	66	0.7706	0.7706	0.7706	0.7706	0.7706
27	27	0.3810	0.3810	0.3810	0.3810	0.3810	67	66	0.0260	0.0260	0.0260	0.0260	0.0260
28	27	-0.0691	-0.0691	-0.0691	-0.0691	-0.0691	67	67	0.3015	0.3015	0.3015	0.3015	0.3015
28	28	1.3412	1.3412	1.3412	1.3412	1.3412	68	67	0.0311	0.0311	0.0311	0.0311	0.0311
29	28	0.0160	0.0160	0.0160	0.0160	0.0160	68	68	0.2662	0.2662	0.2662	0.2662	0.2662
29	29	1.2401	1.2401	1.2401	1.2401	1.2401	69	68	0.0501	0.0501	0.0501	0.0501	0.0501
30	29	-0.0470	-0.0470	-0.0470	-0.0470	-0.0470	69	69	1.4331	1.4331	1.4331	1.4331	1.4331
30	30	2.4459	2.4459	2.4459	2.4459	2.4459	70	69	-0.0314	-0.0314	-0.0314	-0.0314	-0.0314
31	30	0.0066	0.0066	0.0066	0.0066	0.0066	70	70	1.2423	1.2423	1.2423	1.2423	1.2423
31	31	0.9368	0.9368	0.9368	0.9368	0.9368	71	70	-0.0447	-0.0447	-0.0447	-0.0447	-0.0447
32	31	-0.0400	-0.0400	-0.0400	-0.0400	-0.0400	71	71	0.6949	0.6949	0.6949	0.6949	0.6949
32	32	1.1424	1.1424	1.1424	1.1424	1.1424	72	71	-0.0152	-0.0152	-0.0152	-0.0152	-0.0152
33	32	-0.0076	-0.0076	-0.0076	-0.0076	-0.0076	72	72	0.3444	0.3444	0.3444	0.3444	0.3444
33	33	0.8920	0.8920	0.8920	0.8920	0.8920	73	72	0.0016	0.0016	0.0016	0.0016	0.0016
34	33	0.0082	0.0082	0.0082	0.0082	0.0082	73	73	0.0907	0.0907	0.0907	0.0907	0.0907
34	34	0.6148	0.6148	0.6148	0.6148	0.6148	74	73	0.0009	0.0009	0.0009	0.0009	0.0009
35	34	-0.0676	-0.0676	-0.0676	-0.0676	-0.0676	74	74	1.2681	1.2681	1.2681	1.2681	1.2681
35	35	1.8878	1.8878	1.8878	1.8878	1.8878	75	74	-0.0404	-0.0404	-0.0404	-0.0404	-0.0404
36	35	-0.0176	-0.0176	-0.0176	-0.0176	-0.0176	75	75	0.1070	0.1070	0.1070	0.1070	0.1070
36	36	1.1609	1.1609	1.1609	1.1609	1.1609	76	75	0.0063	0.0063	0.0063	0.0063	0.0063
37	36	-0.0513	-0.0513	-0.0513	-0.0513	-0.0513	76	76	0.4486	0.4486	0.4486	0.4486	0.4486
37	37	1.7837	1.7837	1.7837	1.7837	1.7837	77	76	0.0944	0.0944	0.0944	0.0944	0.0944
38	37	0.0471	0.0471	0.0471	0.0471	0.0471	77	77	2.1622	2.1622	2.1622	2.1622	2.1622
38	38	2.4379	2.4379	2.4379	2.4379	2.4379	78	77	-0.0339	-0.0339	-0.0339	-0.0339	-0.0339
39	38	-0.0112	-0.0112	-0.0112	-0.0112	-0.0112	78	78	0.7619	0.7619	0.7619	0.7619	0.7619
39	39	0.1619	0.1619	0.1619	0.1619	0.1619	79	78	0.0300	0.0300	0.0300	0.0300	0.0300
40	39	0.0661	0.0661	0.0661	0.0661	0.0661	79	79	1.4654	1.4654	1.4654	1.4654	1.4654
40	40	1.9771	1.9771	1.9771	1.9771	1.9771	80	79	0.0260	0.0260	0.0260	0.0260	0.0260

Table 3: Indices and elements of matrices D, $\tilde{D}_0$, $\tilde{D}_1$, $\tilde{D}_2$, $\tilde{D}_3$ for example 4.

i	j	$[D]_{ij}$	$[\tilde{D_0}]_{ij}$	$[\tilde{D_1}]_{ij}$	$[\tilde{D_2}]_{ij}$	$[\tilde{D_3}]_{ij}$	i	j	$[D]_{ij}$	$[\tilde{D_0}]_{ij}$	$[\tilde{D_1}]_{ij}$	$[\tilde{D_2}]_{ij}$	$[\tilde{D_3}]_{ij}$
80	80	1.3267	1.3267	1.3267	1.3267	1.3267	91	90	-0.0026	-0.0026	-0.0026	-0.0026	-0.0026
81	80	0.0515	0.0515	0.0515	0.0515	0.0515	91	91	0.6285	0.6285	0.6285	0.6285	0.6285
81	81	2.0186	2.0186	2.0186	2.0186	2.0186	92	91	0.0168	0.0168	0.0168	0.0168	0.0168
82	81	-0.0352	-0.0352	-0.0352	-0.0352	-0.0352	92	92	0.8958	0.8958	0.8958	0.8958	0.8958
82	82	1.0750	1.0750	1.0750	1.0750	1.0750	93	92	-0.0059	-0.0059	-0.0059	-0.0059	-0.0059
83	82	0.0339	0.0339	0.0339	0.0339	0.0339	93	93	0.1448	0.1448	0.1448	0.1448	0.1448
83	83	1.5337	1.5337	1.5337	1.5337	1.5337	94	93	-0.0197	-0.0197	-0.0197	-0.0197	-0.0197
84	83	0.0278	0.0278	0.0278	0.0278	0.0278	94	94	2.2673	2.2673	2.2673	2.2673	2.2673
84	84	0.4384	0.4384	0.4384	0.4384	0.4384	95	94	-0.0110	-0.0110	-0.0110	-0.0110	-0.0110
85	84	-0.0167	-0.0167	-0.0167	-0.0167	-0.0167	95	95	2.2483	2.2483	2.2483	2.2483	2.2483
85	85	1.0386	1.0386	1.0386	1.0386	1.0386	96	95	0.0169	0.0169	0.0169	0.0169	0.0169
86	85	-0.0531	-0.0531	-0.0531	-0.0531	-0.0531	96	96	0.4819	0.4819	0.4819	0.4819	0.4819
86	86	1.7935	1.7935	1.7935	1.7935	1.7935	97	96	0.0678	0.0678	0.0678	0.0678	0.0678
87	86	0.0464	0.0464	0.0464	0.0464	0.0464	97	97	2.2184	2.2184	2.2184	2.2184	2.2184
87	87	0.6006	0.6006	0.6006	0.6006	0.6006	98	97	-0.0253	-0.0253	-0.0253	-0.0253	-0.0253
88	87	-0.0064	-0.0064	-0.0064	-0.0064	-0.0064	98	98	1.6361	1.6361	1.6361	1.6361	1.6361
88	88	0.1532	0.1532	0.1532	0.1532	0.1532	99	98	0.0078	0.0078	0.0078	0.0078	0.0078
89	88	-0.0093	-0.0093	-0.0093	-0.0093	-0.0093	99	99	1.0793	1.0793	1.0793	1.0793	1.0793
89	89	1.0679	1.0679	1.0679	1.0679	1.0679	100	99	0.0001	-0.0006	-0.0006	-0.0006	-0.0006
90	89	0.0289	0.0289	0.0289	0.0289	0.0289	100	100	0.3972	0.2940	0.2940	0.2940	0.2940
90	90	0.9835	0.9835	0.9835	0.9835	0.9835							

Table 4: Indices and elements of matrices D, $\tilde{D_0}$, $\tilde{D_1}$, $\tilde{D_2}$, $\tilde{D_3}$ for example 4 (cont.).

i	j	$[K]_{ij}$	$[\tilde{K}_0]_{ij}$	$[\tilde{K}_1]_{ij}$	$[\tilde{K}_2]_{ij}$	$[\tilde{K}_3]_{ij}$	i	j	$[K]_{ij}$	$[\tilde{K}_0]_{ij}$	$[\tilde{K}_1]_{ij}$	$[\tilde{K}_2]_{ij}$	$[\tilde{K}_3]_{ij}$
1	1	0.2980	0.3794	0.4017	0.4040	0.4045	41	40	-0.0131	-0.0131	-0.0131	-0.0131	-0.0131
2	1	-0.0107	-0.0064	-0.0184	-0.0221	-0.0228	41	41	0.0273	0.0273	0.0273	0.0273	0.0273
2	2	0.1122	0.0863	0.0636	0.0679	0.0685	42	41	-0.0325	-0.0325	-0.0325	-0.0325	-0.0325
3	2	-0.0394	-0.0278	-0.0277	-0.0276	-0.0275	42	42	0.6446	0.6446	0.6446	0.6446	0.6446
3	3	1.0301	1.0301	1.0301	1.0301	1.0301	43	42	-0.0166	-0.0166	-0.0166	-0.0166	-0.0166
4	3	-0.0218	-0.0218	-0.0218	-0.0218	-0.0218	43	43	0.2622	0.2622	0.2622	0.2622	0.2622
4	4	0.3388	0.3388	0.3388	0.3388	0.3388	44	43	0.0282	0.0282	0.0282	0.0282	0.0282
5	4	0.0064	0.0064	0.0064	0.0064	0.0064	44	44	0.7142	0.7142	0.7142	0.7142	0.7142
5	5	0.8838	0.8838	0.8838	0.8838	0.8838	45	44	0.0065	0.0065	0.0065	0.0065	0.0065
6	5	0.0104	0.0104	0.0104	0.0104	0.0104	45	45	0.1770	0.1770	0.1770	0.1770	0.1770
6	6	0.2431	0.2431	0.2431	0.2431	0.2431	46	45	-0.0285	-0.0285	-0.0285	-0.0285	-0.0285
7	6	0.0548	0.0548	0.0548	0.0548	0.0548	46	46	0.5735	0.5735	0.5735	0.5735	0.5735
7	7	0.3357	0.3357	0.3357	0.3357	0.3357	47	46	-0.0149	-0.0149	-0.0149	-0.0149	-0.0149
8	7	0.0416	0.0416	0.0416	0.0416	0.0416	47	47	0.9513	0.9513	0.9513	0.9513	0.9513
8	8	0.8690	0.8690	0.8690	0.8690	0.8690	48	47	-0.0559	-0.0559	-0.0559	-0.0559	-0.0559
9	8	-0.0468	-0.0468	-0.0468	-0.0468	-0.0468	48	48	1.0210	1.0210	1.0210	1.0210	1.0210
9	9	0.9956	0.9956	0.9956	0.9956	0.9956	49	48	0.0117	0.0117	0.0117	0.0117	0.0117
10	9	-0.0545	-0.0545	-0.0545	-0.0545	-0.0545	49	49	1.0097	1.0097	1.0097	1.0097	1.0097
10	10	1.1117	1.1117	1.1117	1.1117	1.1117	50	49	0.0006	0.0006	0.0006	0.0006	0.0006
11	10	0.0019	0.0019	0.0019	0.0019	0.0019	50	50	1.1858	1.1858	1.1858	1.1858	1.1858
11	11	0.9711	0.9711	0.9711	0.9711	0.9711	51	50	0.0118	0.0118	0.0118	0.0118	0.0118
12	11	0.0213	0.0213	0.0213	0.0213	0.0213	51	51	0.6299	0.6299	0.6299	0.6299	0.6299
12	12	1.0589	1.0589	1.0589	1.0589	1.0589	52	51	-0.0183	-0.0183	-0.0183	-0.0183	-0.0183
13	12	0.0137	0.0137	0.0137	0.0137	0.0137	52	52	0.1195	0.1195	0.1195	0.1195	0.1195
13	13	0.3661	0.3661	0.3661	0.3661	0.3661	53	52	-0.0113	-0.0113	-0.0113	-0.0113	-0.0113
14	13	0.0021	0.0021	0.0021	0.0021	0.0021	53	53	0.2642	0.2642	0.2642	0.2642	0.2642
14	14	0.3436	0.3436	0.3436	0.3436	0.3436	54	53	0.0265	0.0265	0.0265	0.0265	0.0265
15	14	0.0430	0.0430	0.0430	0.0430	0.0430	54	54	0.5898	0.5898	0.5898	0.5898	0.5898
15	15	0.7593	0.7593	0.7593	0.7593	0.7593	55	54	0.0236	0.0236	0.0236	0.0236	0.0236
16	15	0.0288	0.0118	0.0118	0.0118	0.0118	55	55	0.9430	0.9430	0.9430	0.9430	0.9430
16	16	0.9102	0.7172	0.7172	0.7172	0.7172	56	55	-0.0052	-0.0052	-0.0052	-0.0052	-0.0052
17	16	-0.0649	0.0979	0.0979	0.0979	0.0979	56	56	0.0359	0.0359	0.0359	0.0359	0.0359
17	17	0.7940	0.7934	0.7934	0.7934	0.7934	57	56	-0.0039	-0.0039	-0.0039	-0.0039	-0.0039
18	17	-0.0375	-0.1435	-0.1435	-0.1435	-0.1435	57	57	1.0276	1.0276	1.0276	1.0276	1.0276
18	18	0.2864	0.2866	0.2866	0.2866	0.2866	58	57	0.1115	0.1115	0.1115	0.1115	0.1115
19	18	0.0057	0.0020	0.0020	0.0020	0.0020	58	58	1.0951	1.0951	1.0951	1.0951	1.0951
19	19	0.1731	0.0935	0.0935	0.0935	0.0935	59	58	-0.0412	-0.0412	-0.0412	-0.0412	-0.0412
20	19	-0.0058	0.0026	0.0026	0.0026	0.0026	59	59	0.1886	0.1886	0.1886	0.1886	0.1886
20	20	1.1777	1.1776	1.1776	1.1776	1.1776	60	59	0.0209	0.0209	0.0209	0.0209	0.0209
21	20	-0.0158	-0.0010	-0.0010	-0.0010	-0.0010	60	60	0.8946	0.8946	0.8946	0.8946	0.8946
21	21	0.7943	0.7943	0.7943	0.7943	0.7943	61	60	0.0328	0.0328	0.0328	0.0328	0.0328
22	21	-0.0009	-0.0009	-0.0009	-0.0009	-0.0009	61	61	0.3736	0.3736	0.3736	0.3736	0.3736
22	22	0.7647	0.7647	0.7647	0.7647	0.7647	62	61	0.0094	0.0094	0.0094	0.0094	0.0094
23	22	-0.0057	-0.0057	-0.0057	-0.0057	-0.0057	62	62	0.2816	0.2816	0.2816	0.2816	0.2816
23	23	0.7363	0.7363	0.7363	0.7363	0.7363	63	62	-0.0140	-0.0140	-0.0140	-0.0140	-0.0140
24	23	0.0047	0.0047	0.0047	0.0047	0.0047	63	63	0.6170	0.6170	0.6170	0.6170	0.6170
24	24	0.5057	0.5057	0.5057	0.5057	0.5057	64	63	0.0370	0.0370	0.0370	0.0370	0.0370
25	24	-0.0652	-0.0652	-0.0652	-0.0652	-0.0652	64	64	0.5822	0.5822	0.5822	0.5822	0.5822
25	25	1.2316	1.2316	1.2316	1.2316	1.2316	65	64	0.0605	0.0605	0.0605	0.0605	0.0605
26	25	0.0744	0.0744	0.0744	0.0744	0.0744	65	65	0.6281	0.6281	0.6281	0.6281	0.6281
26	26	0.6940	0.6940	0.6940	0.6940	0.6940	66	65	0.0173	0.0173	0.0173	0.0173	0.0173
27	26	0.0067	0.0067	0.0067	0.0067	0.0067	66	66	0.3706	0.3706	0.3706	0.3706	0.3706
27	27	0.1870	0.1870	0.1870	0.1870	0.1870	67	66	0.0209	0.0209	0.0209	0.0209	0.0209
28	27	-0.0352	-0.0352	-0.0352	-0.0352	-0.0352	67	67	0.1492	0.1492	0.1492	0.1492	0.1492
28	28	0.6323	0.6323	0.6323	0.6323	0.6323	68	67	0.0251	0.0251	0.0251	0.0251	0.0251
29	28	-0.0104	-0.0104	-0.0104	-0.0104	-0.0104	68	68	0.1386	0.1386	0.1386	0.1386	0.1386
29	29	0.5899	0.5899	0.5899	0.5899	0.5899	69	68	0.0446	0.0446	0.0446	0.0446	0.0446
30	29	-0.0952	-0.0952	-0.0952	-0.0952	-0.0952	69	69	0.7056	0.7056	0.7056	0.7056	0.7056
30	30	1.2424	1.2424	1.2424	1.2424	1.2424	70	69	-0.0590	-0.0590	-0.0590	-0.0590	-0.0590
31	30	0.0314	0.0314	0.0314	0.0314	0.0314	70	70	0.6026	0.6026	0.6026	0.6026	0.6026
31	31	0.4440	0.4440	0.4440	0.4440	0.4440	71	70	-0.0032	-0.0032	-0.0032	-0.0032	-0.0032
32	31	-0.0402	-0.0402	-0.0402	-0.0402	-0.0402	71	71	0.3224	0.3224	0.3224	0.3224	0.3224
32	32	0.5256	0.5256	0.5256	0.5256	0.5256	72	71	-0.0249	-0.0249	-0.0249	-0.0249	-0.0249
33	32	0.0002	0.0002	0.0002	0.0002	0.0002	72	72	0.1528	0.1528	0.1528	0.1528	0.1528
33	33	0.4183	0.4183	0.4183	0.4183	0.4183	73	72	0.0075	0.0075	0.0075	0.0075	0.0075
34	33	0.0127	0.0127	0.0127	0.0127	0.0127	73	73	0.0435	0.0435	0.0435	0.0435	0.0435
34	34	0.3118	0.3118	0.3118	0.3118	0.3118	74	73	0.0045	0.0045	0.0045	0.0045	0.0045
35	34	-0.0492	-0.0492	-0.0492	-0.0492	-0.0492	74	74	0.6108	0.6108	0.6108	0.6108	0.6108
35	35	0.9529	0.9529	0.9529	0.9529	0.9529	75	74	-0.0136	-0.0136	-0.0136	-0.0136	-0.0136
36	35	-0.0197	-0.0197	-0.0197	-0.0197	-0.0197	75	75	0.0552	0.0552	0.0552	0.0552	0.0552
36	36	0.5571	0.5571	0.5571	0.5571	0.5571	76	75	-0.0001	-0.0001	-0.0001	-0.0001	-0.0001
37	36	-0.0192	-0.0192	-0.0192	-0.0192	-0.0192	76	76	0.2254	0.2254	0.2254	0.2254	0.2254
37	37	0.8393	0.8393	0.8393	0.8393	0.8393	77	76	0.0219	0.0219	0.0219	0.0219	0.0219
38	37	0.0677	0.0677	0.0677	0.0677	0.0677	77	77	1.0077	1.0077	1.0077	1.0077	1.0077
38	38	1.2079	1.2079	1.2079	1.2079	1.2079	78	77	-0.0653	-0.0653	-0.0653	-0.0653	-0.0653
39	38	-0.0390	-0.0390	-0.0390	-0.0390	-0.0390	78	78	0.3813	0.3813	0.3813	0.3813	0.3813
39	39	0.0787	0.0787	0.0787	0.0787	0.0787	79	78	0.0338	0.0338	0.0338	0.0338	0.0338
40	39	0.0668	0.0668	0.0668	0.0668	0.0668	79	79	0.6700	0.6700	0.6700	0.6700	0.6700
40	40	0.9517	0.9517	0.9517	0.9517	0.9517	80	79	0.0037	0.0037	0.0037	0.0037	0.0037

Table 5: Indices and elements of matrices K, $\tilde{K}_0$, $\tilde{K}_1$, $\tilde{K}_2$, $\tilde{K}_3$ for example 4.

i	j	$[K]_{ij}$	$[\tilde{K_0}]_{ij}$	$[\tilde{K_1}]_{ij}$	$[\tilde{K_2}]_{ij}$	$[\tilde{K_3}]_{ij}$	i	j	$[K]_{ij}$	$[\tilde{K_0}]_{ij}$	$[\tilde{K_1}]_{ij}$	$[\tilde{K_2}]_{ij}$	$[\tilde{K_3}]_{ij}$
80	80	0.6383	0.6383	0.6383	0.6383	0.6383	91	90	0.0088	0.0088	0.0088	0.0088	0.0088
81	80	0.0502	0.0502	0.0502	0.0502	0.0502	91	91	0.3181	0.3181	0.3181	0.3181	0.3181
81	81	0.9791	0.9791	0.9791	0.9791	0.9791	92	91	0.0300	0.0300	0.0300	0.0300	0.0300
82	81	-0.0360	-0.0360	-0.0360	-0.0360	-0.0360	92	92	0.4434	0.4434	0.4434	0.4434	0.4434
82	82	0.5431	0.5431	0.5431	0.5431	0.5431	93	92	-0.0073	-0.0073	-0.0073	-0.0073	-0.0073
83	82	0.0479	0.0479	0.0479	0.0479	0.0479	93	93	0.0653	0.0653	0.0653	0.0653	0.0653
83	83	0.7244	0.7244	0.7244	0.7244	0.7244	94	93	-0.0365	-0.0365	-0.0365	-0.0365	-0.0365
84	83	0.0377	0.0377	0.0377	0.0377	0.0377	94	94	1.1224	1.1224	1.1224	1.1224	1.1224
84	84	0.2167	0.2167	0.2167	0.2167	0.2167	95	94	-0.0261	-0.0261	-0.0261	-0.0261	-0.0261
85	84	-0.0033	-0.0033	-0.0033	-0.0033	-0.0033	95	95	1.0743	1.0743	1.0743	1.0743	1.0743
85	85	0.4981	0.4981	0.4981	0.4981	0.4981	96	95	0.0040	0.0040	0.0040	0.0040	0.0040
86	85	-0.0502	-0.0502	-0.0502	-0.0502	-0.0502	96	96	0.2336	0.2336	0.2336	0.2336	0.2336
86	86	0.8774	0.8774	0.8774	0.8774	0.8774	97	96	0.0659	0.0659	0.0659	0.0659	0.0659
87	86	0.0462	0.0462	0.0462	0.0462	0.0462	97	97	1.0218	1.0218	1.0218	1.0218	1.0218
87	87	0.2655	0.2655	0.2655	0.2655	0.2655	98	97	-0.0390	-0.0390	-0.0390	-0.0390	-0.0390
88	87	-0.0020	-0.0020	-0.0020	-0.0020	-0.0020	98	98	0.7724	0.7724	0.7724	0.7724	0.7724
88	88	0.0770	0.0770	0.0770	0.0770	0.0770	99	98	0.0666	0.0666	0.0666	0.0666	0.0666
89	88	0.0097	0.0097	0.0097	0.0097	0.0097	99	99	0.5172	0.5172	0.5172	0.5172	0.5172
89	89	0.5222	0.5222	0.5222	0.5222	0.5222	100	99	-0.0045	-0.0011	-0.0011	-0.0011	-0.0011
90	89	0.0295	0.0295	0.0295	0.0295	0.0295	100	100	0.1932	0.1392	0.1392	0.1392	0.1392
90	90	0.4324	0.4324	0.4324	0.4324	0.4324							

Table 6: Indices and elements of matrices K, $\tilde{K_0}$, $\tilde{K_1}$, $\tilde{K_2}$, $\tilde{K_3}$ for example 4 (cont.).

References

[1] M. O. Abdalla, K. M. Grigoriadis, and D. C. Zimmerman, Enhanced structural damage detection using alternating projection methods, *AIAA Journal* 36, pp. 1305–1311, 1998.

[2] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt, On Augmented Lagrangian methods with general lower-level constraints, *SIAM Journal on Optimization* 18, pp. 1286–1309, 2007.

[3] R. Andreani, E. G. Birgin, J. M. Martínez, and M. L. Schuverdt, Augmented Lagrangian methods under the Constant Positive Linear Dependence constraint qualification, *Mathematical Programming* 111, pp. 5–32, 2008.

[4] M. Andretta, E. G. Birgin and J. M. Martínez, Practical active-set Euclidean trust-region method with spectral projected gradients for bound-constrained minimization, *Optimization* 54, pp. 305–325, 2005.

[5] Z.-J. Bai, B. N. Datta, and J. Wang, Robust and minimum norm partial quadratic eigenvalue assignment in vibrating systems: A new optimization approach, *Mechanical Systems and Signal Processing* 24, pp. 766–783, 2010.

[6] M. Baruch, Optimization procedure to correct stiffness and flexibility matrices using vibration data, *AIAA Journal* 16, pp. 1208–1210, 1978.

[7] C. A. Beattie and S. W. Smith, Optimal matrix approximants in structural identification, *Journal of Optimization Theory and Applications* 74, pp. 23–56, 1992.

[8] E. G. Birgin and J. M. Martínez, A box-constrained optimization algorithm with negative curvature directions and spectral projected gradients, *Computing [Suppl]* 15, pp. 49–60, 2001.

[9] E. G. Birgin and J. M. Martínez, Large-scale active-set box-constrained optimization method with spectral projected gradients, *Computational Optimization and Applications* 23, pp. 101–125, 2002.

[10] E. G. Birgin and J. M. Martínez, *Practical Augmented Lagrangian Methods for Constrained Optimization*, Society for Industrial and Applied Mathematics, Philadelphia, 2014.

[11] S. Brahma and B. N. Datta, An optimization approach for minimum norm and robust partial quadratic eigenvalue assignment problems for vibrating structures, *Journal of Sound and Vibration* 324, pp. 471–489, 2009.

[12] J. B. Carvalho, B. N. Datta, A. Gupta, and M. Lagadapati, A direct method for model updating with incomplete measured data and without spurious modes, *Mechanical Systems and Signal Processing* 21, pp. 2715–2731, 2007.

[13] J. B. Carvalho, B. N. Datta, W.-W. Lin, and C.-S. Wang, Symmetry preserving eigenvalue embedding in finite element model updating of vibrating structures, *Journal of Sound and Vibration* 290, pp. 839–864, 2006.

[14] M. T. Chu, B. N. Datta, W.-W. Lin, and S.-F. Xu, Spillover phenomenon in quadratic model updating, *AIAA Journal* 46, pp. 420–428, 2008.

[15] M. T. Chu, W.-W. Lin, and S.-F. Xu, Updating quadratic models with no spill-over effect on unmeasured spectral data, *Inverse Problems* 23, pp. 243–256, 2007.

[16] B. N. Datta, Finite element model updating, eigenstructure assignment and eigenvalue embedding techniques for vibrating systems, *Mechanical Systems and Signal Processing* 16, pp. 83–96, 2002.

[17] B. N. Datta, *Numerical Linear Algebra and Applications*, second edition, SIAM, Philadelphia, 2010.

[18] B. N. Datta, *Numerical methods for linear control systems*, Elsevier Academic Press, San Diego, CA, 2004.

[19] B. N. Datta, S. Deng, V. O. Sokolov, and D. R. Sarkissian, An optimization technique for damped model updating with measured data satisfying quadratic orthogonality constraint, *Mechanical Systems and Signal Processing* 23, pp. 1759–1772, 2009.

[20] B. N. Datta, S. Elhay, Y. M. Ram, and D. R. Sarkissian, Partial eigenstructure assignment for the quadratic pencil, *Journal of Sound and Vibration* 1, pp. 101–110, 2000.

[21] B. N. Datta and D. R. Sarkissian, Multi-input partial eigenvalue assignment for the symmetric quadratic pencil, *Proceedings of the 1999 American Control Conference* 4, pp. 2244–2247, 1999.

[22] B. N. Datta and D. R. Sarkissian, Theory and computations of some inverse eigenvalue problems for the quadratic pencil, *Contemporary Mathematics, Volume Structured Matrices in Operator Theory, Control and Signal an Image Processing* 280, American Mathematical Society, pp. 221–240, 2001.

[23] B. N. Datta and D. R. Sarkissian, A computational method for feedback control in distributed parameter systems, *Proceedings of the 8th IEEE International Conference on Methods and Models in Robotics*, pp. 139–144, 2002.

[24] B. N. Datta and V. O. Sokolov, Quadratic inverse eigenvalue problems, active vibration control and model updating, *Applied and Computational Mathematics* 8, pp. 170–191, 2009.

[25] B. N. Datta and V. O. Sokolov, A solution of the affine quadratic inverse eigenvalue problem, *Linear Algebra and its Applications* 434, pp. 1745–1760, 2011.

[26] M. A. Diniz-Ehrhardt, M. A. Gomes-Ruggiero, J. M. Martínez, and S. A. Santos, Augmented Lagrangian algorithms based on the spectral projected gradient for solving nonlinear programming problems, *Journal of Optimization Theory and Applications* 123, pp. 497–517, 2004.

[27] M. Friswell and J. E. Mottershead, *Finite Element Model Updating in Structural Dynamics*, Kluwer Academic Publishers, London, 1995.

[28] J. C. Geromel, On the determination of a diagonal solution of the Lyapunov equation, *IEEE Transactions on Automatic Control* AC-30, pp. 404–406, 1985.

[29] M. E. Hochstenbach and H. A. van Der Vorst, Alternatives to the Rayleigh quotient for the quadratic eigenvalue problem, *SIAM Journal on Scientific Computing* 25, pp. 591–603, 2003.

[30] H. Hu, Positive definite constrained least-squares estimation of matrices, *Linear Algebra and its Applications* 229, pp. 167–174, 1995.

[31] D. J. Inman, *Vibrations: with Control, Measurement, and Stability*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[32] D. J. Inman and A. Kress, Eigenstructure assignment using inverse eigenvalue methods, *Journal of Guidance, Control, and Dynamics* 18, pp. 625–627, 1995.

[33] Y.-C. Kuo and B. N. Datta, Quadratic model updating with no spill-over and incomplete measured data: Existence and computation of solution, *Linear Algebra and its Applications* 436, pp. 2480–2493, 2012.

[34] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, Society for Industrial and Applied Mathematics, Philadelphia, 1998.

[35] K. Miettinen, *Nonlinear Multiobjective Optimization*, Kluwer academic Publishers, Boston, MA, 1999.

[36] J. Moreno, B. N. Datta, and M. Raydan, A symmetry preserving alternating projection method for matrix model updating, *Mechanical Systems and Signal Processing* 23, pp. 1784–1791, 2009.

[37] N. K. Nichols and J. Kautsky, Robust eigenstructure assignment in quadratic matrix polynomials: nonsingular case, *SIAM Journal on Matrix Annalysis and Applications* 23, pp. 77–102, 2001.

[38] F. Tisseur and K. Meerbergen, The quadratic eigenvalue problem, *SIAM Review* 43, pp. 235–286, 2001.