An algorithm for quadratic eigenproblems with low rank damping

Taslaman, Leo

2014

MIMS EPrint: **2014.21**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

AN ALGORITHM FOR QUADRATIC EIGENPROBLEMS WITH LOW RANK DAMPING *

LEO TASLAMAN †

Abstract. We consider quadratic eigenproblems $\left(M\lambda^2 + D\lambda + K\right) x = 0$, where all coefficient matrices are real and positive semidefinite, (M, K) is regular and D is of low rank. Matrix polynomials of this form appear in the analysis of vibrating structures with discrete dampers. We develop an algorithm for such problems, which first solves the undamped problem $\left(M\lambda^2 + K\right) x = 0$ and then accommodates for the low rank term $D\lambda$. For the first part, we modify an algorithm proposed by Wang and Zhao [SIAM J. Matrix Anal. Appl. 12-4 (1991), pp. 654–660]. The modified algorithm is then used to solve the undamped problem such that all eigenvalues are computed in a backward stable manner. We then use the solution to the undamped problem to compute all eigenvalues of the original problem, and the associated eigenvectors if requested. To this end, we use an Ehrlich-Aberth iteration that works exclusively with vectors and tall skinny matrices and contributes only with lower order terms to the flop count. Numerical experiments show that the proposed algorithm is both fast and accurate. Finally we discuss the application to the large scale case and the possibility of generalizations.

Key words. quadratic eigenvalue problem, eigenvalue algorithm, matrix polynomial, discrete damper, vibrating system

AMS subject classifications. 15A18, 15A22, 65F15, 70J10, 70J30, 70J50

1. Introduction.

Some eigenproblems would be a lot easier to solve if it were not for an aggravating low rank term. We consider one family of such problems: the quadratic eigenproblems (QEPs)

$$\left(M\lambda^2 + D\lambda + K\right) x = 0, \tag{1.1}$$

where M, D and K are real, $n \times n$ and positive semidefinite matrices, (M, K) is regular (that is, $\det(M\lambda + K) \not\equiv 0$) and $r := \operatorname{rank}(D) \ll n$. The QEP (1.1) has a lot of nice properties that gets "ruined" by the low rank term $D\lambda$. Without it, a simple substitution, $\omega = -\lambda^2$, turns (1.1) into a definite generalized eigenproblem (GEP):

$$Kx = \omega Mx. \tag{1.2}$$

Hence, when D is absent, solving (1.1) is not harder than solving a definite GEP. Our goal is to design an algorithm that can handle the presence of the low rank matrix D without increasing the complexity significantly.

1.1. Motivation.

QEPs on the form (1.1) appears naturally in modal analysis of physical structures. We now discuss this application briefly. The discussion serves not only as a motivation, but also allows us to use our intuition of mechanical systems to understand the choice of starting points used in our algorithm later on.

The study of free vibrations (that is, vibrations caused by initial displacements) plays an important role in structural engineering. To find out how a structure moves when it vibrates freely a finite element discretization is typically made. After this simplification, the displacements of the nodes in the model are given by the solutions

*Version of May 13, 2014. This work was supported by Engineering and Physical Sciences Research Council grant EP/I005293

†School of Mathematics, The University of Manchester, Manchester, M13 9PL, UK (leo.taslaman@manchester.co.uk)

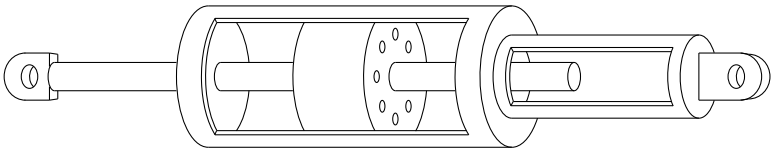

FIG. 1.1. *A model of a viscous damper. The larger cylinder is filled with a fluid. When the piston rod moves horizontally the fluid is forced through the holes of the piston head which causes friction.*

to system of ordinary differential equations:

$$\left(M\frac{\mathrm{d}^2}{\mathrm{d}t^2} + D\frac{\mathrm{d}}{\mathrm{d}t} + K\right)u(t) = 0. \tag{1.3}$$

Here M, D and K are real positive semidefinite matrices corresponding to mass, damping and stiffness respectively.

In theory, M should be strictly positive definite, but singular M is common in practice due to further simplifications by the engineers. The stiffness matrix K is positive definite when boundary conditions that prevent rigid body motions are present; this is the case for buildings and bridges but not airplanes, say. The damping matrix depends on what kind of damping is modeled. We consider the case of discrete (linear) damping, which refers to the physical objects called viscous dampers (see Figure 1.1). A viscous damper can be embedded into a structure to prevent movement in a certain direction. When a viscous damper is modeled with finite elements, it appears as a positive semidefinite rank one term of the damping matrix. Hence, if the structure only has a few viscous dampers, their contribution to the damping matrix is of low rank. This is indeed the case in several applications. It is not uncommon that less than ten dampers are used, and in some cases as few as one or two [1].

If only a few viscous dampers are used, and M and K does not have a common nontrivial nullspace, then we get the solution to (1.3) by solving the corresponding quadratic eigenproblem given by (1.1). In particular, if (λ, x) is an eigenpair of (1.1), then $u(t) = e^{\lambda t}x$ is a solution to (1.3). Now, since the coefficient matrices are real and positive semidefinite, the spectrum is symmetric with respect ot the real axis and lies in the left half plane. Thus, if $(-d + i\omega, x)$ is an eigenpair of (1.1) and $\omega > 0$, then so is its complex conjugate. It follows that the real function

$$u(t) = e^{-d}(\cos(t\omega)\mathrm{Re}(x) + \sin(t\omega)\mathrm{Im}(x)).$$

is a solution to (1.3). Notice how the real and imaginary parts of the eigenvalue correspond to damping and frequency respectively.

1.2. Existing algorithms and our objective. The conventional way of solving (1.1) is through linearization, which means that the problem is rewritten as GEP of twice the size. This approach does not respect the special structure of problem (1.1). There do exist symmetric linearizations, but no stable algorithm that can preserve this symmetry is currently available.

Recently, Hammarling, Munro and Tisseur proposed a linearization based algorithms for finding all eigenpairs of general regular quadratic eigenproblems [8]. Their algorithm, called `quadeig`, is backward stable in the unstructured sense described in section 2.3, as long as the damping is not too strong. The bulk of the computation

lies in solving the linearized problem, for which the QZ algorithm is used. The QZ algorithm is estimated to use $50m^3$ flops ($30m^3$ flops if we only want the eigenvalues), where m is the size of the matrices [7, p. 413]. Since `quadeig` works on a linearization we have $m = 2n$, where n is size of the coefficient matrices M, D and K. We get an estimated complexity of $400n^3$ flops ($240m^3$ flops for eigenvalues only).

We shall develop an algorithm that exploits the structure of problem (1.1) and whose main complexity lies in finding all eigenpairs of the definite GEP (1.2). There are several fast methods for solving (1.2). We use a modified version of the algorithm proposed by Wang and Zhao [21], which is estimated to need about $26n^3$ flops if M and K are nonsingular, and up to $43n^3$ flops otherwise.

1.3. Outline. In section 2 necessary background material is discussed. This includes how to detect defective eigenvalues, definitions of backward errors for QEPs, and the Ehrlich-Aberth method. In section 3 we review Wang and Zhao's algorithm for definite GEPs and propose an improvement and a generalization. In section 4 our algorithm for solving (1.1) is described and in section 5 we present the results from numerical experiments. In section 6 we discuss the application to the large scale case and the possibility of generalizations.

2. Preliminaries.

2.1. Defective eigenvalues. Let $P(\lambda)$ be a regular matrix polynomial. The definition of a *left eigenvector* of $P(\lambda)$ varies in the literature; sometimes the complex conjugate transpose is used [18], and sometimes the transpose [13]. We use the latter definition, so a left eigenvector of $P(\lambda)$ is a vector y, such that $y^T P(\lambda_0) = 0$ for some λ_0. Left and right eigenvectors can be used to determine whether or not an eigenvalue is defective. For constant matrices, this follows from the Jordan canonical form: if x and y are right and left eigenvectors, respectively, corresponding to the same Jordan block, then $y^T x = 0$ if and only if that Jordan block is nontrivial. Furthermore, if x and y are right and left eigenvectors corresponding to different Jordan blocks, then $y^T x = 0$. Hence, an eigenvalue is defective if and only if there exists an associated right eigenvector x such that $y^T x = 0$ for all left eigenvectors y. This result can be generalized to matrix polynomials with invertible leading coefficient.

THEOREM 2.1 (Lancaster [13, p. 63]). *Let $P(\lambda)$ be a matrix polynomial with invertible leading coefficient. If λ_0 is an eigenvalue of $P(\lambda)$ then λ_0 is defective if and only if there exists an associated right eigenvector x such that $y^T P'(\lambda_0)x = 0$ for all left eigenvectors y.*

REMARK 2.2. *From the proof of Theorem 2.1, it follows that x comes from an arbitrary Jordan decomposition of an associated* real *linearization. For real eigenvalues of real matrix polynomials, we can use the real Jordan form and hence assume that x is real.*

The assumption that the leading coefficient is invertible may be too strong. A way to get around this is to employ a Möbius transformation. Let λ_0 denote an arbitrary (possibly infinite) eigenvalue of $P(\lambda)$ and let $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ be its partial multiplicity sequence (that is, the sizes of the associated Jordan blocks). If

$$m(\lambda) = \frac{a\lambda + b}{c\lambda + d}$$

is an invertible Möbius transformation, then $Q(\lambda) := (c\lambda + d)^{\deg P} P(m(\lambda))$ has the same eigenvectors as $P(\lambda)$ and $m^{-1}(\lambda_0)$ is an eigenvalue of $Q(\lambda)$ with partial multiplicity sequence $(\alpha_1, \alpha_2, \ldots, \alpha_k)$ [14, 23]. Suppose now that σ is not an eigenvalue

of $P(\lambda)$. Then the choice $m(\lambda) = 1/\lambda + \sigma$ implies that $Q(\lambda)$ has invertible leading coefficient. Since $m^{-1}(\lambda) = 1/(\lambda - \sigma)$ we arrive at the following corollary.

COROLLARY 2.3. *Assume that* $\det P(\sigma) \neq 0$ *and define* $Q(\lambda) = \lambda^{\deg P} P(1/\lambda + \sigma)$. *If* λ_0 *is an eigenvalue of* $P(\lambda)$ *then* λ_0 *is defective if and only if there exists an associated eigenvector* x *such that* $y^T Q'(1/(\lambda_0 - \sigma))x = 0$ *for all left eigenvectors* y.

From Remark 2.2, it follows that x in Corollary 2.3 may be chosen to be real if $P(\lambda)$, λ_0 and σ are real.

2.2. Backward errors for polynomial eigenproblems. Let $(\widetilde{x}, \widetilde{\lambda})$ denote a computed eigenpair of a matrix polynomial

$$P(\lambda) = \sum_{k=0}^{\ell} A_k \lambda^k \quad \text{and let} \quad \Delta P(\lambda) = \sum_{k=0}^{\ell} \Delta A_k \lambda^k$$

denote a perturbation of $P(\lambda)$. If $\widetilde{\lambda}$ is finite, we follow [18] and define the relative backward error of the computed eigenpair $(\widetilde{x}, \widetilde{\lambda})$ as

$$\eta_P(\widetilde{x}, \widetilde{\lambda}) = \min\{\epsilon : (P + \Delta P)(\widetilde{\lambda})\tilde{x} = 0,\ \|\Delta A_i\| \leq \epsilon \|A_i\|,\ i = 0\colon \ell\},$$

and the relative backward error of the computed eigenvalue $\tilde{\lambda}$ as

$$\eta_P(\widetilde{\lambda}) = \min_{x \neq 0} \eta_P(x, \widetilde{\lambda}). \tag{2.1}$$

In general $\|\cdot\|$ can be any matrix norm; in this paper, however, we will only use the spectral norm, so $\|\cdot\| = \|\cdot\|_2$. For the spectral norm, it was proved in [18] that

$$\eta_P(\widetilde{x}, \widetilde{\lambda}) = \|P(\widetilde{\lambda})\widetilde{x}\| \left(\|\widetilde{x}\| \sum_{k=0}^{\ell} \|A_k\| |\widetilde{\lambda}|^k \right)^{-1}$$

and

$$\eta_P(\widetilde{\lambda}) = \left(\|P(\widetilde{\lambda})^{-1}\| \sum_{k=0}^{\ell} \|A_k\| |\widetilde{\lambda}|^k \right)^{-1}. \tag{2.2}$$

Notice that $\eta_P(\widetilde{x}, \widetilde{\lambda}) = \eta_{\operatorname{rev} P}(\widetilde{x}, 1/\widetilde{\lambda})$ and $\eta_P(\widetilde{\lambda}) = \eta_{\operatorname{rev} P}(1/\widetilde{\lambda})$ for $\widetilde{\lambda} \neq 0$, where

$$\operatorname{rev} P(\lambda) := \sum_{k=0}^{\ell} A_{\ell - k} \lambda^k.$$

Since infinite eigenvalues of $P(\lambda)$ are defined as the zero eigenvalues of $\operatorname{rev} P(\lambda)$, it is natural to define $\eta_P(\widetilde{x}, \infty) = \eta_{\operatorname{rev} P}(\widetilde{x}, 0)$ and $\eta_P(\infty) = \eta_{\operatorname{rev} P}(0)$. We also note that if $Q(\lambda)$ is related to $P(\lambda)$ via an eigenvalue parameter scaling, so

$$Q(\lambda) = \sum_{k=0}^{\ell} (s^k A_k) \lambda^k,$$

then

$$\eta_P(\widetilde{x}, s\widetilde{\lambda}) = \eta_Q(\widetilde{x}, \widetilde{\lambda}) \quad \text{and} \quad \eta_P(s\widetilde{\lambda}) = \eta_Q(\widetilde{\lambda}). \tag{2.3}$$

2.3. Ehrlich-Aberth iteration. The Ehrlich-Aberth method [6, 2] is an algorithm for simultaneously finding all roots of a scalar polynomial. If

$$p(\lambda) = 0$$

is the scalar polynomial equation we want to solve, then the algorithm takes starting points $\lambda_1^{(0)}, \ldots, \lambda_\ell^{(0)}$, where $\ell = \deg(p)$, and then update these points via

$$\lambda_k^{(i+1)} = \lambda_k^{(i)} - \frac{N(\lambda_k^{(i)})}{1 - N(\lambda_k^{(i)}) \sum\limits_{j \neq k} \dfrac{1}{\lambda_k^{(i)} - \lambda_j^{(i)}}}, \tag{2.4}$$

where $N(\lambda) := p(\lambda)/p'(\lambda)$. Clearly these updates can be done in parallel, which is nice, but if we insist to update in sequential order we might as well use the latest approximations available. This leads to the slightly faster Gauss-Seidel version:

$$\lambda_k^{(i+1)} = \lambda_k^{(i)} - \frac{N(\lambda_k^{(i)})}{1 - N(\lambda_k^{(i)}) \left(\sum\limits_{j<k} \dfrac{1}{\lambda_k^{(i)} - \lambda_j^{(i+1)}} - \sum\limits_{j>k} \dfrac{1}{\lambda_k^{(i)} - \lambda_j^{(i)}} \right)}. \tag{2.5}$$

In practice, the Ehrlich-Aberth method exhibits rapid convergence to isolated simple eigenvalues if good starting points are provided. The algorithm also converges for multiple and tightly clustered eigenvalues, but more iterations are generally required in these cases.

Recently Bini and Noferini [5] used the Ehrlich-Aberth method for finding the eigenvalues of regular matrix polynomials. If $P(\lambda)$ is such matrix polynomial, their algorithm applies the Ehrlich-Aberth iteration to the equation $\det P(\lambda) = 0$, and for the selection of starting points, it make use of Newton polygons. For the evaluation of $p(\lambda)/p'(\lambda)$, which is the most expensive part of the updating process, they used Jacobi's formula

$$\frac{d}{d\lambda} \det P(\lambda) = \operatorname{trace}\left(P(\lambda)^{-1} P'(\lambda)\right) \det P(\lambda)$$

to obtain

$$p'(\lambda)/p(\lambda) = \operatorname{trace}\left(P(\lambda)^{-1} P'(\lambda)\right). \tag{2.6}$$

By using (2.6), each update costs $O(n^3)$ flops.

Since the method is iterative, some stopping criterion is needed. Bini and Noferini gave two suggestions: either stop updating λ_i when the condition number of $P(\lambda_i)$ is large enough, or when the associated backward error (2.2) is small enough. Both criteria require $O(n^3)$ flops to check.

The Ehrlich-Aberth method can only be used to find the eigenvalues. If also the eigenvectors are sought, these can be found afterwards using inverse iteration or the SVD—both techniques requires $O(n^3)$ flops per eigenvector.

The algorithm in [5] demonstrated superb accuracy in numerical tests, but is unfortunately an expensive alternative for solving QEPs. Applied to an $n \times n$ QEP the complexity is $O(n^4)$—assuming the starting points are good enough.

Algorithm 1: Wang and Zhao's algorithm.

Description: Solves $(A - \lambda B)x = 0$ where $A, B \in \mathbb{R}^{n\times n}$ are positive definite.

1 Compute Cholesky decompositions $A = L_A L_A^T$ and $B = L_B L_B^T$.
2 Compute the QR factorization $\begin{bmatrix} L_A^T \\ L_B^T \end{bmatrix} = QR$.
3 Define $Q_1 = [I_n \; 0_{n\times n}]Q$ and $Q_2 = [0_{n\times n} \; I_n]Q$.
4 Compute the singular values $\sigma_1(Q_1) \geq \sigma_2(Q_1) \geq \cdots \geq \sigma_n(Q_1)$ of Q_1.
5 Compute the singular values $\sigma_1(Q_2) \geq \sigma_2(Q_2) \geq \cdots \geq \sigma_n(Q_2)$ of Q_2 and a corresponding matrix V of right singular vectors.
6 Compute eigenvalues: $\lambda_i = \sigma_i(Q_1)/\sigma_{n-i+1}(Q_2)$ for $i = 1\colon n$.
7 Compute eigenvectors: $x_i = R^{-1}(Ve_i)$ for $i = 1\colon n$.

3. Wang and Zhao's algorithm. Wang and Zhao [21] proposed a algorithm for solving

$$Ax = \lambda Bx, \tag{3.1}$$

where $A, B \in \mathbb{R}^{n\times n}$ are positive definite. Their method is outlined in Algorithm 1. To see why Algorithm 1 works, we note that the matrix Q, has a CS decomposition

$$Q = \begin{bmatrix} U_1 & \\ & U_2 \end{bmatrix} \begin{bmatrix} C \\ S \end{bmatrix} V^T,$$

where $Q_1 = U_1 C V^T$ and $Q_2 = U_2 S V^T$ are SVDs. Since each column of Q has unit norm, so must be the case for each column of $[C^T \; S^T]^T$. In other words, it must hold that $c_{ii}^2 + s_{ii}^2 = 1$ for $i = 1\colon n$. If we define $X = R^{-1}V$, then we have

$$X^T A X = V^T R^{-T} L_A L_A^T R^{-1} V = V^T Q_1^T Q_1 V = C^2 \tag{3.2}$$

and similarly

$$X^T B X = S^2. \tag{3.3}$$

Now consider the case when A or B (possibly both) are singular but still positive semidefinite, and the pencil $A - \lambda B$ is regular. For such problems, Algorithm 1 still works after a small modification: instead of computing Cholesky factorizations on line 1, we compute any other factorizations such that $A = L_A L_A^T$ and $B = L_B L_B^T$, where L_A and L_B need not be triangular. If A, say, is singular we can, for example, use the factorization given by $L_A = U\Lambda^{1/2}$ where $A = U\Lambda U^T$ is a spectral decomposition. Regarding the eigenvectors, we have $Rx = 0$ only if $(A - \lambda B)x = 0$ independent of λ. Hence, the assumption that $A - \lambda B$ is regular implies that R is invertible.

Wang and Zhao showed that Algorithm 1 finds the the exact eigenvalues of a perturbed problem

$$(A + \Delta A)x = \lambda(B + \Delta B)x,$$

where ΔA and ΔB are *symmetric* and $\|\Delta A\|/(\|A\| + \|B\|)$ and $\|\Delta B\|/(\|A\| + \|B\|)$ are both small. Here (and below) "small" refers to a modest multiple of machine precision that depends on n. The backward error analysis in [21] is oblivious to which factorizations are being done on line 1 of Algorithm 1 as long as

$$L_A L_A^T = A + \Delta\widetilde{A} \quad \text{and} \quad L_B L_B^T = B + \Delta\widetilde{B},$$

where $\|\Delta\widetilde{A}\|/\|A\|$ and $\|\Delta\widetilde{B}\|/\|B\|$ are both small. Since the factorizations based on the spectral decomposition mentioned above, can be computed stably using e.g., the QR algorithm [17], [7, p. 464], the backward error analysis in [21] can be applied also to this case.

The above backward error result does not say anything about $\|\Delta A\|/\|A\|$ and $\|\Delta B\|/\|B\|$, and is therefore not satisfactory with respect to the backward error definition (2.1). Fortunately, this can be fixed by an eigenvalue parameter scaling. If we use Algorithm 1 (possibly with our modification to handle singular matrices) to solve

$$Ax = \lambda(sB)x \quad \text{where} \quad s = \|A\|/\|B\|,$$

rather than (3.1), then we get computed eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ such that the backward errors $\eta_{A-(sB)\lambda}(\lambda_i)$, $i = 1\colon n$, are small. From (2.3), we see that $s\lambda_1, s\lambda_2, \ldots, s\lambda_n$ have small backward errors as eigenvalue approximations to (3.1).

We have suggested two improvements to Algorithm 1. The first one allows for positive semidefinite A and B as long as (A, B) is regular. The second one guarantees small backward errors for the eigenvalues, as defined in (2.1). Taken together, our modifications leads to a new algorithm which is summarized in Algorithm 2; the corresponding flop count is shown in Table 17.[1] We remark that the two if-then-else statements in Algorithm 2 can be executed in parallel. Similarly, the computation of the SVD quantities of Q_1 and Q_2 (line 14 and 15) can be done in parallel.

Algorithm 2: The Modified Wang-Zhao algorithm.

Description: Solves $(A - \lambda B)x = 0$ where $A, B \in \mathbb{R}^{n\times n}$ are positive semidefinite and $A - \lambda B$ is regular.

1 **if** A is nonsingular **then**
2 | Compute Cholesky factorizations $A = L_A L_A^T$.
3 **else**
4 | Compute a spectral decomposition $A = U_A \Lambda_A U_A^T$ and set $L_A = U_A \Lambda_A^{1/2}$.
5 **end**
6 **if** B is nonsingular **then**
7 | Compute Cholesky factorizations $B = L_B L_B^T$.
8 **else**
9 | Compute a spectral decomposition $B = U_B \Lambda_B U_B^T$ and set $L_B = U_B \Lambda_B^{1/2}$.
10 **end**
11 Let $s = \|A\|/\|B\|$ (If $\|A\|$ or $\|B\|$ are unknown, estimations suffice).
12 Compute the QR factorization $\begin{bmatrix} L_A^T \\ L_B^T \end{bmatrix} = \text{QR}$.
13 Define $Q_1 = [I_n\ 0_{n\times n}]Q$ and $Q_2 = [0_{n\times n}\ I_n]Q$.
14 Compute the singular values $\sigma_1(Q_1) \geq \sigma_2(Q_1) \geq \cdots \geq \sigma_n(Q_1)$ of Q_1.
15 Compute the singular values of $\sigma_1(Q_2) \geq \sigma_2(Q_2) \geq \cdots \geq \sigma_n(Q_2)$ of Q_2 and a corresponding matrix V of right singular vectors.
16 Compute eigenvalues $\lambda_i = s\sigma_i(Q_1)/\sigma_{n-i+1}(Q_2)$ for $i = 1\colon n$.
17 Compute eigenvectors $x_i = R^{-1}(Ve_i)$ for $i = 1\colon n$.

[1] Wang and Zhao pointed out the cost of the QR factorization can be reduced if we can take advantage of the triangular structure of L_A and L_B (assuming A and B are nonsingular). For simplicity, this will not be exploited in this paper.

TABLE 3.1
Flop count estimation for Algorithm 2.

Cholesky factorization	$(1/3)n^3$	[7, p. 164]
Symmetric QR Algorithm	$9n^3$	[7, p. 463]
Householder QR factorization ($2n \times n$)	$(12 + 2/3)n^3$	[7, p. 249]
Singular values	$(2 + 2/3)n^3$	[7, p. 493]
Singular values + right singular vectors	$12n^3$	[7, p. 493]
Triangular linear system	n^2	[7, p. 107]
Alg. 2: A and B nonsingular	$29n^3$	
Alg. 2: A xor B singular	$(37 + 2/3)n^3$	
Alg. 2: A and B singular	$(46 + 1/3)n^3$	

The backward error analysis in [21] only concerns the eigenvalues, not the eigenvectors. Since the eigenvectors are given by $R^{-1}V$ the quality of the computed eigenvectors depends on the triangular matrix R. As mentioned above, this matrix is always invertible, but it may be ill-conditioned. In exact arithmetic we have,

$$R^T R = A + sB, \quad s = \|A\|/\|B\|,$$

so R is ill-conditioned exactly when there exists a vector v such that both $v^T Av/\|A\|$ and $v^T Bv/\|B\|$ are small.

4. An algorithm for QEPs with low rank damping. The proposed algorithm for solving (1.1) is outlined briefly in Algorithm 3.

Algorithm 3: The main algorithm

Description: Computes all eigenvalues/eigenpairs of (1.1).

1 Compute an $S \in \mathbb{R}^{n\times r}$ such that $D = SS^T$.
2 Compute the *undamped eigenvalues* (that is, the eigenvalues of $M\lambda^2 + K$) and a nonsingular $X \in \mathbb{R}^{n\times n}$ such that

$$X^T(M\lambda^2 + SS^T\lambda + K)X = M_d\lambda^2 + \widehat{S}\widehat{S}^T\lambda + K_d =: P(\lambda), \qquad (4.1)$$

where M_d and K_d are diagonal.
3 Lock undamped eigenvalues that are also eigenvalues of (1.1).
4 Compute the eigenvalues of (4.1) by the Ehrlich-Aberth iteration. Return the computed eigenvalues if the eigenvectors are not requested.
5 Compute the eigenvectors of (4.1) by inverse iteration.
6 Return (λ_i, Xv_i), $i = 1\colon 2n$, where (λ_i, v_i) is a computed eigenpair of (4.1).

For the first step of Algorithm 3, we can find S by, for instance, computing the spectral decomposition of D.

The second step of Algorithm 3 essentially reduces to solving a definite GEP. It is easy to see that X must be an eigenvector matrix corresponding to $K - M\omega$. Furthermore, if ω_k, $k = 1\colon n$, are the eigenvalues of $K - M\omega$, then the undamped eigenvalues are given by $\pm i\sqrt{\omega_k}$ if ω_k is finite, and ∞ otherwise. We use Algorithm 2 to find all eigenpairs of $K - M\omega$. Note that there is no need to form the matrices $X^T MX$ and $X^T KX$ explicitly: from (3.2) and (3.3) we see that M_d and M_d are given by the singular values computed in Algorithm 2.

The description of step 3, 4 and 5 are more involved, so we discuss these in separate subsections.

4.1. Step 3: Initial locking. It may happen that some undamped eigenvalues are also eigenvalues to (1.1). Since there is no need to do any further work on such eigenvalues, we declare them "locked." When deciding which eigenvalues to lock, we treat zero and infinite eigenvalues separately from nonzero finite eigenvalues. The reason for this is that a zero and infinite eigenvalue of (1.1) may be defective, while all eigenvalues of the form $\pm i\sqrt{\omega_k}$ can be shown to be semisimple. Matrix polynomials (and constant matrices for that matter) with defective eigenvalues are often regarded as degenerate cases. Indeed, if we randomly generate the coefficient matrices of a matrix polynomial, it will almost surely have no defective eigenvalues. We will see that this is not necessarily the case if we impose rank constraints on the coefficient matrices and force them to be positive semidefinite.

Suppose (λ_k, x_k), where $\lambda_k \neq 0, \infty$, is an eigenpair of $M\lambda^2 + K$ and let $\eta(\lambda_k, x_k)$ denote the corresponding backward error with respect to $M\lambda^2 + D\lambda + K$. We declare λ_k "locked" if $\eta(\lambda_k, x_k)$ is small enough. In our code, "small enough" is defined as less than $n\epsilon$ where ϵ is machine precision.

The next proposition provides a method to determine how many of the zero and infinite eigenvalues to lock.

PROPOSITION 4.1. *Let $Q(\lambda) = M\lambda^2 + D\lambda + K$ be the matrix polynomial in* (1.1)*, so (M, K) is regular. The number of zero eigenvalues is given by*

$$\dim \operatorname{null}(K) + \dim(\operatorname{null}(D) \cap \operatorname{null}(K)),$$

and the number of infinite eigenvalues is given by

$$\dim \operatorname{null}(M) + \dim(\operatorname{null}(D) \cap \operatorname{null}(M)).$$

Proof. For readability, we introduce the following variables

$$k := \dim \operatorname{null}(K) \quad \text{and} \quad \ell := \dim(\operatorname{null}(D) \cap \operatorname{null}(K)).$$

Pick a real invertible X_1 such that $X_1^T M X_1$ and $X_1^T K X_1$ are diagonal and the first k diagonal elements of $X_1^T K X_1$ are zero. This can be done since (M, K) is a definite pencil. Further, pick an invertible $X_2 \in \mathbb{R}^{k\times k}$ such that the first $\ell \leq k$ columns $X_2 \oplus I_{n-k}$ is a basis for $\operatorname{null}(X_1^T D X_1) \cap \operatorname{null}(X_1^T K X_1)$ and the first k columns is a basis for $\operatorname{null}(X_1^T K X_1)$. Let $X = X_1(X_2 \oplus I_{n-k})$ and note that $X^T Q(\lambda) X$ decomposes into a direct sum

$$X^T Q(\lambda) X = M_1\lambda^2 \oplus (M_2\lambda^2 + D_2\lambda + K_2),$$

where M_1 is $\ell \times \ell$ and $\operatorname{null}(D_2) \cap \operatorname{null}(K_2) = \{0\}$. Note that $M_1\lambda^2$ has exactly 2ℓ zero eigenvalues and $Q_2(\lambda) := M_2\lambda^2 + D_2\lambda + K_2$ has at least $k - \ell$ zero eigenvalues. We need to show that $Q_2(\lambda)$ has exactly $k - \ell$ zero eigenvalues, or equivalently, that all its zero eigenvalues are semisimple. To this end, we observe that $Q_2(\lambda)$ is real and symmetric, so all right eigenvectors associated with zero are also left eigenvectors. Next, we pick $\sigma > 0$ such that $\det Q_2(\sigma) \neq 0$ and define

$$\widehat{Q}(\lambda) = \lambda^2 Q_2(1/\lambda + \sigma).$$

From Corollary 2.3 it follows that zero is a defective eigenvalue of $Q_2(\lambda)$ only if there exists a real right eigenvector x such that

$$x^T \widehat{Q}'(-1/\sigma)x = x^T(D_2 + K_2/\sigma)x = 0.$$

Since D_2 and K_2 are both positive semidefinite, such x must lie in $\operatorname{null}(D_2) \cap \operatorname{null}(K_2)$ and hence cannot exist.

The number of infinite eigenvalues equals the number of zero eigenvalues of $\operatorname{rev} Q(\lambda) := K\lambda^2 + D\lambda + M$. Thus, the other half of the theorem can be shown analogously if we consider $\operatorname{rev} Q(\lambda)$ instead of $Q(\lambda)$. ∎

REMARK 4.2. *The number of "missing eigenvectors" corresponding to the zero and infinite eigenvalues are given by* $\dim(\operatorname{null}(D) \cap \operatorname{null}(K))$ *and* $\dim(\operatorname{null}(D) \cap \operatorname{null}(M))$*, respectively. Hence, defective eigenvalues are always present if, for example,* $\operatorname{rank}(D) = 1$ *and* $\dim \operatorname{null}(K) = 2$.

By Proposition 4.1, the number of zero and infinite eigenvalues depends on $\operatorname{null}(K)$ and $\operatorname{null}(M)$, respectively. These spaces are available from the corresponding spectral decompositions, which are computed in Algorithm 2. If the columns of $N_1 \in \mathbb{R}^{n \times k_1}$ and $N_2 \in \mathbb{R}^{n \times k_2}$ constitute bases for $\operatorname{null}(K)$ and $\operatorname{null}(M)$, respectively, then there are $2k_1 - \operatorname{rank}(DN_1)$ zero eigenvalues and $2k_2 - \operatorname{rank}(DN_2)$ infinite eigenvalues. These quantities can be computed numerically using the SVD.

4.2. Step 4: Computing eigenvalues. We now discuss how to use the Ehrlich-Aberth method to exploit the structure of (4.1) in order to find all eigenvalues. We focus on the following three questions.

1. How do we pick the starting points?
2. How do we compute (2.6) efficiently?
3. Which stopping criterion should we use?

For starting points we use the undamped eigenvalues with small (in a relative sense) random perturbations added to the unlocked eigenvalues. These perturbations are added to break symmetries, since it is well-known that the Ehrlich-Aberth method may fail to converge due to certain symmetries [2]. Suppose, for example, that (1.1) has two real simple eigenvalues and all undamped eigenvalues are finite and nonzero. Assume further we want to use the update rule (2.4). If we do not add the perturbations, then starting points can be paired into complex conjugates, and the update rule (2.4) preserves this symmetry. Hence, convergence to real simple eigenvalues is impossible.

The rationale behind using the undamped eigenvalues as starting points becomes more clear if we think about the application discussed in section 1.1. In this case the eigenvalues correspond to vibrational properties (frequency and damping) of a physical structure, and the undamped eigenvalues correspond to vibrational properties of the *same structure*, but with the strength of the dampers set to zero. If the damping is small or moderate, our the choice of starting points seems reasonable. But what if the damping is strong? In this case we note that a strong viscous damper (that is, one with small holes in its piston head, see Figure 1.1) is in some sense similar to a rigid component. We expect the spectrum to respect this similarity. The link between strong dampers and the spectrum was studied to some extent in [16]. In particular it was shown that all eigenvalues of $M\lambda^2 + sD\lambda + K$, where M and K are positive definite and D positive semidefinite, converge to points on the imaginary axis as $s \to \infty$, with the exception of $\operatorname{rank}(D)$ eigenvalues which goes to $-\infty$. This means that $\operatorname{rank}(D)$ eigenvalues can be arbitrarily far from all the staring points.

Fortunately, as will be seen in section 5.3, Ehrlich-Aberth works quite well in practice when only a few starting points are "way off."

The computation of trace $\left(P(\lambda)^{-1}P'(\lambda)\right)$ for fix values of λ is the core of our Ehrlich-Aberth iteration. We compute this trace using the Sherman-Morrison-Woodbury formula in combination with standard trace laws. The precise procedure is outlined in Algorithm 4; we leave out the tedius algebra that justifies it. If M_d and K_d are stored as vectors and Algorithm 4 is implemented accordingly, then total flop count is $4n + 2rn + 4r^2n$ (counting only terms with a factor n). Since there are $2n$ eigenvalues, and we expect each eigenvalue to converge in a few steps, the complexity in n of our Ehrlich-Aberth iteration is quadratic.

Algorithm 4: Computation of trace $\left(P(\lambda)^{-1}P'(\lambda)\right)$.

Description: Computes $t = \text{trace}\left(P(\lambda)^{-1}P'(\lambda)\right)$ where
$$P(\lambda) = M_d\lambda^2 + \widehat{S}\widehat{S}^T\lambda + K_d.$$

1 $A := M_d\lambda^2 + K_d$
2 $B := A^{-1}\widehat{S}$
3 $C := \widehat{S}^T B$
4 $D := I_r + \lambda C$
5 $E := M_d B$
6 $F := (B^T E)D^{-1}$
7 $G := CD^{-1}C$
8 $t := 2\lambda\text{trace}(M_d A^{-1}) + \text{trace}(C) - 2\lambda^2\text{trace}(F) - \lambda\text{trace}(G)$

When an eigenvalue has converged, we mark it as "locked" and do not update it in subsequent iterations. We are done when all eigenvalues are locked. The obvious question is "When do we declare an eigenvalue 'converged'?" One approach is to estimate the backward error (2.2) with respect to (4.1), and lock an eigenvalue if the backward error is smaller than some tolerance, say machine precision. If we use the `normest1` algorithm [10] in combination with the Sherman-Morrison-Woodbury formula, such estimation requires only $O(n)$ flops if we count r as a small constant. We found, however, that we often get better results (both in terms of accuracy and speed) with the following heuristic strategy: lock $\lambda_k^{(i)}$ when

$$\left|\lambda_k^{(i)} - \lambda_k^{(i+1)}\right| < \text{tol} \times \left|\lambda_k^{(i)}\right|,$$

where tol is initially set to be machine precision, and is then relaxed by a factor 10 each 50th iteration. This is the stopping condition used in our numerical experiments. Here the number 50 is somewhat arbitrary. From experience, convergence of most eigenvalues is usually obtained within 10 iterations. Some eigenvalues requires more iterations, but the idea is that if 50 is not enough then the problem is most likely not the number of iterations, but rather that the tolerance is too stringent. We stress that the argument is based purely on experience, so there may very well exist pathological examples where this strategy fails. We remark, however, that some kind of relaxation strategy for the tolerance is necessary also when the eigenvalue backward error is used as a stopping condition—otherwise the iteration may go on forever. We comment more on this at the end section 5.3.

4.3. Step 5: Computing eigenvectors. When all eigenvalues have been found we compute the corresponding eigenvectors. Since the computation of eigenvectors

corresponding to different eigenvalues are completely decoupled, this phase of the algorithm is embarrassingly parallel. We now discuss how to determine an eigenvector v_i of $P(\lambda)$ corresponding to a computed eigenvalue λ_i. If λ_i is an undamped eigenvalue, then v_i is just a column of the identity matrix; otherwise, more work is required. The next proposition provides one method for computing v_i.

PROPOSITION 4.3. *Let λ_i be an eigenvalue of $P(\lambda)$ but not of $Q(\lambda) := P(\lambda) - SS^T\lambda$. Then all eigenvectors associated with λ_i lie in the range of $Q(\lambda_i)^{-1}S$.*

Proof. Suppose (v_i, λ_i) is an eigenpair and write $v_i = Q(\lambda_i)^{-1}Sx + y$ where $y \perp \text{range}(Q(\lambda_i)^{-1}S)$. We need to show that $y = 0$. We have

$$\begin{aligned} P(\lambda_i)v_i &= P(\lambda_i)(Q(\lambda_i)^{-1}Sx + y) \\ &= (Q(\lambda_i) + \lambda_i SS^T)(Q(\lambda_i)^{-1}Sx + y) \\ &= S(I_r + \lambda_i S^T Q(\lambda_i)^{-1}S)x + Q(\lambda_i)y + \lambda_i SS^T y \\ &= 0, \end{aligned}$$

which implies that $Q(\lambda_i)y \in \text{range}(S)$, or equivalently, that $y \in \text{range}(Q(\lambda_i)^{-1}S)$. The result now follows from the definition of y. ∎

REMARK 4.4. *A consequence of Proposition 4.3 is that the geometric multiplicity of λ_i cannot exceed the rank(S).*

Proposition 4.3 implies that it if λ_i is computed exactly, then it is enough to look for eigenvectors in the r dimensional subspace range$(Q(\lambda_i)^{-1}S)$. Furthermore, we see from the proof that $Q(\lambda_i)^{-1}Sx$ is an eigenvector of $P(\lambda)$ for any $x \in \text{null}(I_r + \lambda_i S^T Q(\lambda_i)^{-1}S)$. Since x can be computed cheaply from the SVD of $I_r + \lambda_i S^T Q(\lambda_i)S$, this yields a fast method for computing v_i. In practice, however, the computed eigenvalues contain errors so Proposition 4.3 is strictly speaking not applicable, and the discussed method may lead to inaccurate eigenvectors. The computed eigenvectors are, however, often very good (frequently with perfect backward errors of order 10^{-16}) and serves as excellent starting vectors for inverse iteration.

There are several approaches to inverse iteration for polynomial eigenproblems, see [15]. The approach we take is (to our knowledge) new. It is designed for real symmetric matrix polynomials and is slightly cheaper than the alternatives—although it may be argued that the savings are negligible in our context. The idea is to iterate according to

$$v_i^{(k+1)} = P(\lambda_i)^{-1}\overline{v}_i^{(k)}/\|v_i^{(k)}\|. \tag{4.2}$$

So, why does this work? To answer this, we note that $P(\lambda_i)$ is complex symmetric and hence enjoys an SVD on the form $\overline{U}\Sigma U^*$ [11, Corollary 4.4.4]. If $U = [u_1, u_2, \ldots, u_n]$, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ and $v_i^{(k)} = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_n u_n$, then

$$P(\lambda_i)^{-1}\overline{v}_i^{(k)} = U\Sigma^{-1}\overline{U}^*\overline{v}_i^{(k)} = \sum_{j=1}^{n} \frac{\alpha_j}{\sigma_j} u_j. \tag{4.3}$$

Since σ_n is tiny when λ_i is close to an eigenvalue, we expect (4.3) to be huge in the direction of u_n. This is delightful, since the vector u_n is the best possible eigenvector approximation we can hope for in the sense that $\eta_P(\lambda_i, u_n) = \eta_P(\lambda_i)$. As usual with inverse iteration, the ill-conditioning of $P(\lambda_i)$ is benign since the matrix magnifies errors in the direction of the desired vector.

To compute (4.2) we use the Sherman-Morrison-Woodbury formula with the starting vector described above. Since the starting vector already is a good approximation,

we only take one step of inverse iteration in our code. The complexity for computing one eigenvector of (4.1) with this technique is linear in n.

Another way to solve the linear systems from (4.2) is to use QR factorization and back substitution. This is an attractive option from a stability point of view, albeit more expensive. If the technique used in [19] is employed, each QR factorization can be computed with $O(rn^2)$ flops. The idea is to compute, in a bottom-up fashion, a sequence of (real) Givens rotations $U_1U_2 \dots U_{n-1} =: U$ such that US is trapezoidal and UM_d and UK_d are r-Hessenberg. This implies that $UP(\lambda_i)$ is r-Hessenberg for any λ_i, so its QR factorization can be computed efficiently using Givens rotations. We did not use this approach for our numerical experiments. It may, however, be the method of choice when only some eigenvectors are sought, or when the eigenvectors are computed in parallel.

We end this section with a negative remark: the approach to first find the eigenvalues, and then the eigenvectors via inverse iteration, is flawed when multiple eigenvalues are present. In this case we may approximate the same eigenvector several times. An obvious "solution" is to compute an invariant subspace rather than individual eigenvectors; inverse iteration and our choice of starting vector can indeed be generalized to subspaces. The problem is that it is hard to a priori decide what the dimension of the subspaces should be.

5. Numerical experiments. We implemented Algorithm 3 in MATLAB 2012b. Our code is written in serial, so it does not, for instance, exploit that the workload in steps 4 and 5 of the algorithm is embarrassingly parallel. Individual MATLAB functions that are being called, may, however, be multithreaded. For the Ehrlich-Aberth iteration, we used the Gauss-Seidel updates shown in (2.5). The first part of our algorithm (step 1–3) make use of MATLAB's core routines `svd` and `qr`. The second part of our algorithm (step 4–6) is written in "pure" MATLAB code (except for the computation of small $r \times r$ SVDs) and is sometimes slower than the first part even though the flop count is much lower. Since we expect this speed difference to wane if the entire algorithm is implemented in a low-level language, we sometimes state explicitly how much time is spend on the second part.

We compared our algorithm to `quadeig`, the MATLAB implementation of the eigensolver in [8] for unstructured QEPs. In the core of this implementation we find MATLAB's `eig` routine, which performs the real QZ algorithm in this case.

The numerical experiments were carried out in MATLAB R2012b in IEEE double precision arithmetic on a machine with the following specifications.

Memory	16GB (4X4GB) 1333 MHz DDR3 Non-ECC
Processor	Intel® Core™ i7-2600 (8M Cache, 3.40 GHz)
Operating System	Windows® 7 Professional (64Bit)

5.1. The damped beam. The damped beam problem can be found in the collection of nonlinear eigenvalue problems called NLEVP [4]. This QEP arises when studying the vertical displacements of a beam that is supported at its ends and has a viscous damper attached to it in the middle, see Figure 5.1.

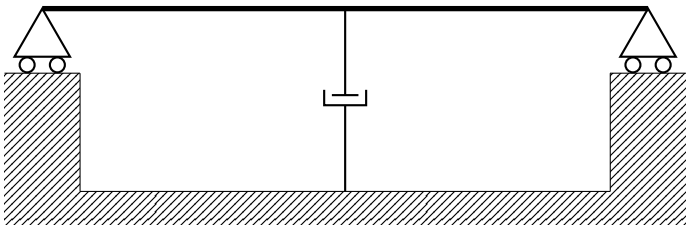

FIG. 5.1. *The damped beam.*

The construction of the coefficient matrices is explained in [9], where it is also shown that half of the eigenvalues are undamped eigenvalues. This makes it an ideal problem for our algorithm. We modeled the problem such that the coefficient matrices were of size 1000×1000. Algorithm 3 computed all eigenpairs in 2 seconds while `quadeig` needed 112 seconds. The backward errors for the computed eigenpairs are shown in Figure 5.2. We remark that there is no guarantee that two backward errors plotted with the same x-coordinate correspond to the same eigenvalue. We see that both algorithm performed well in terms of backward stability (all backward errors are less than n times the machine precision). The spectrum, as it was computed by the two algorithms, are shown in Figure 5.3.

Let us pause a for a while and discuss Figure 5.3. We know that all eigenvalues must lie in the left half plane, and half of them on the imaginary axis. Hence the real parts of some of the computed eigenvalues from `quadeig` must be inaccurate, even though Figure 5.2 shows all backward errors are about 10^{-14}. In terms of relative errors, this is consistent with the "approximate bound"

$$\text{forward error} \lesssim \text{backward error} \times \text{condition number},$$

for the *unstructured* forward error, if we define the condition number conformably

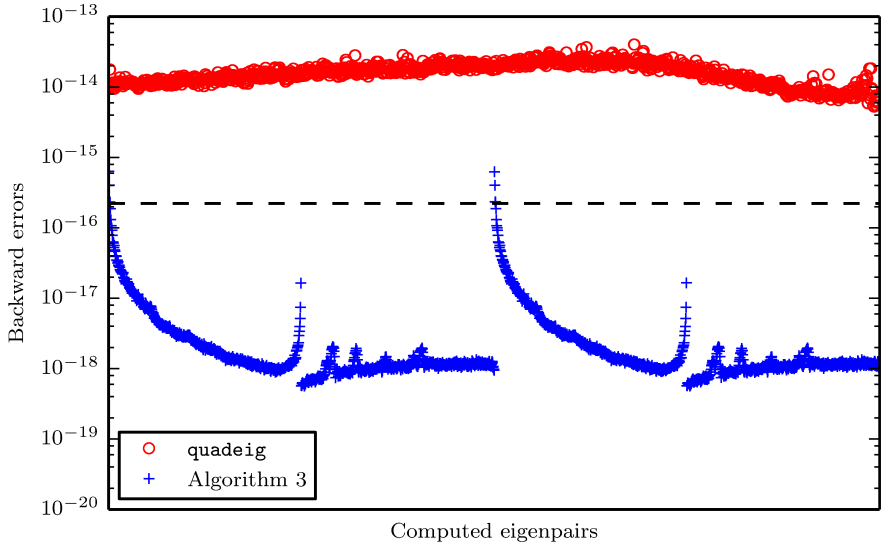

FIG. 5.2. *Backward errors of computed eigenpairs for the damped beam. The dashed line indicates the machine precision.*

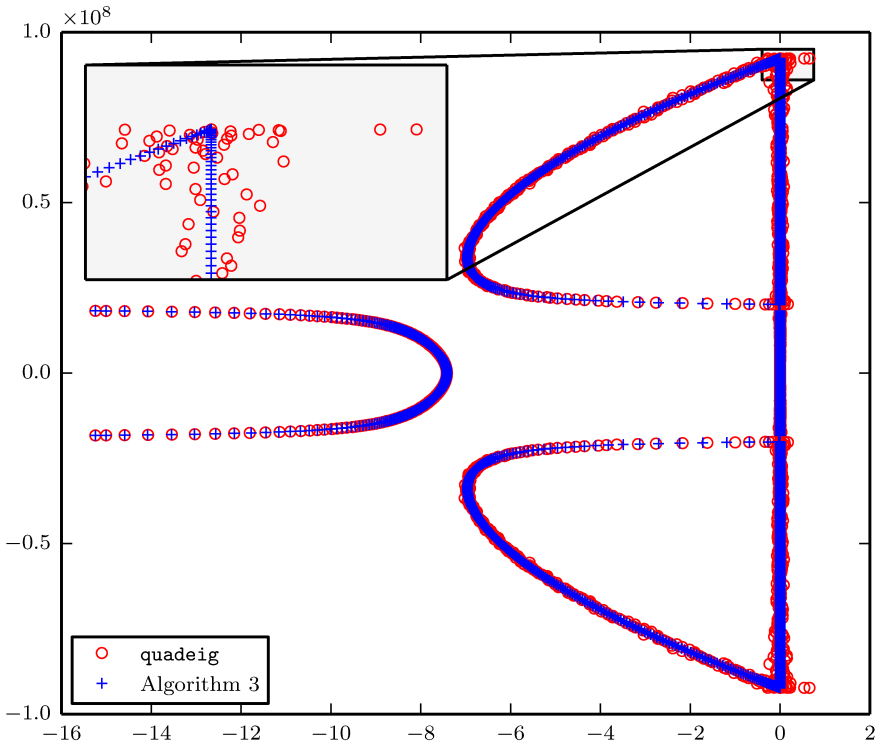

FIG. 5.3. *Computed spectra for the damped beam.*

with the backward error introduced in section 2.2. This condition number is given by

$$\kappa(\lambda) = \frac{\|M\||\lambda|^2 + \|D\||\lambda| + \|K\|}{|\lambda||v^T(2M\lambda + D)v|},$$

if λ is a simple nonzero eigenvalue of (1.1) and v is an associated normalized eigenvector [18]. If we, for example, evaluate the condition number of the upper-right-most eigenvalue using the computed quantities from Algorithm 3, we find that the condition number is of order 10^7. Assuming this answer is of the correct order of magnitude, the *relative* forward error is at most of order $10^{-14} \times 10^7 = 10^{-7}$. For the *absolute* forward error, we note that the modulus of the eigenvalue in question is about 10^8, so the absolute forward error is at most of order $10^{-14} \times 10^7 \times 10^8 = 10$. This explains why we see some red circles in the right half plane. The unstructured forward error bound does not, however, explain the nice pattern in the spectrum produced by Algorithm 3. One explanation is the problem has a lot of structure that Algorithm 3 respects.

Finally, recall that each undamped eigenvalue is computed as $\pm i\omega_k$, for some real eigenvalue ω_k of $K - M\omega$, and is then locked if it is also an eigenvalue of the damped problem. This explains the straight line of blue crosses on the imaginary axis in Figure 5.3. However, even if we bypass the initial locking phase and add relative perturbations of order 10^{-8} to *all* eigenvalues, our Ehrlich-Aberth iteration returns half the eigenvalues in a strip centered at the imaginary axis of width about 10^{-13}.

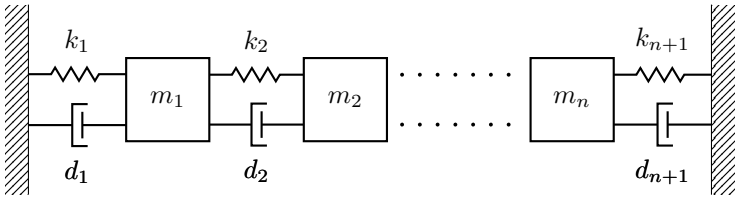

FIG. 5.4. *A simple mass-spring-damper system.*

5.2. A mass-spring-damper system. Our next QEP comes from a simple mass-spring-damper system; the particular setup is shown in Figure 5.4. To make the problem more interesting, we introduced defective infinite eigenvalues by setting some of the masses, as well as most damping coefficients, to zero. We defined $n = 1000$,

$$m_i = \begin{cases} 0 & \text{if } i \in \{1, n\} \\ 1 & \text{otherwise,} \end{cases} \qquad d_i = \begin{cases} 1/100 & \text{if } i \in \mathcal{J} := \{12, n/2+1, n-10\} \\ 0 & \text{otherwise} \end{cases}$$

and $k_i = 1$ for $i = 1\colon n$. Notice that there only are three effective dampers, that is, with nonzero damping coefficients d_i. The corresponding mass, damping and stiffness matrices are given by

$$M = I - e_1 e_1^T - e_n e_n^T, \quad D = \frac{1}{100} \sum_{i \in \mathcal{J}} (e_{i-1} - e_i)(e_{i-1} - e_i)^T$$

and

$$K = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix},$$

respectively [20, p. 2]. Further, Proposition 4.1 implies that the associated QEP has four defective infinite eigenvalues. We solved the eigenproblem using Algorithm 3 and `quadeig`. The computation time for Algorithm 3 was 5 seconds, where more than half the time was spend on the second part (step 4–6) of the algorithm; the computation time for `quadeig` was 110 seconds. The backward errors for the computed eigenpairs are shown in Figure 5.5. As in Figure 5.2, two backward errors plotted with the same x-coordinate may correspond to different eigenvalues.

5.3. QEPs with random coefficient matrices. We created random coefficient matrices using the MATLAB commands

```
M = randn(n);  M = M*M';
D = randn(n,5); D = D*D';
K = randn(n); K = K*K';
```

and solved the corresponding problem for different values on n. Note that the rank of the damping matrix is 5 for each test problem. The results are shown in Table 5.1. As expected, Algorithm 3 scales much better with n than `quadeig`.

Our next experiment concerns strongly damped problems, or more precisely, problems for which $\|D\|$ is much larger than $\|M\|$ and $\|K\|$. Such problems have badly

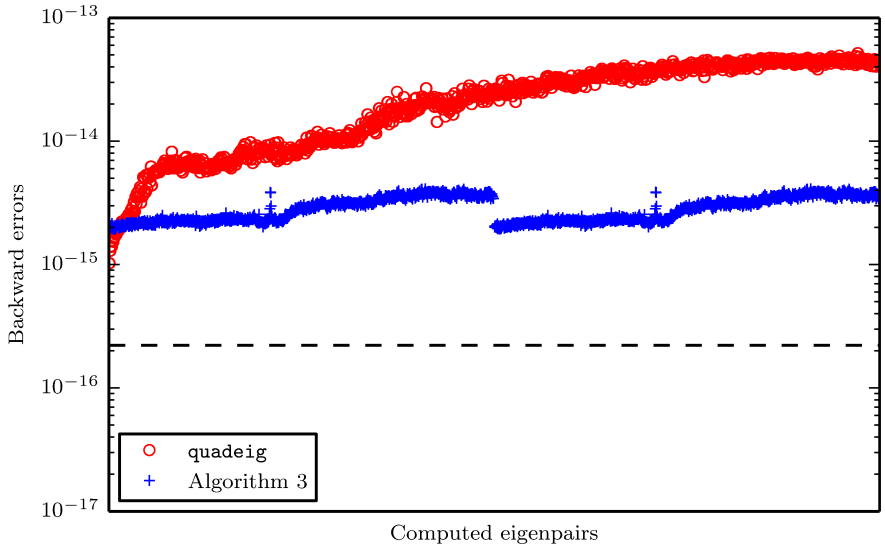

FIG. 5.5. *Backward errors of computed eigenpairs for the mass-spring-damper system described in section 5.2. The dashed line indicates the machine precision.*

TABLE 5.1
Backward errors and execution times for the tested algorithms. The last columns shows how many times, on average, each eigenvalue approximation was updated in the Ehrlich-Aberth iteration.

	`quadeig`		Algorithm 3		
n	max $\eta(\lambda, v)$	time	max $\eta(\lambda, v)$	time (step 4–6)	Av. #upd
200	6.3e−15	0.4	1.7e−15	0.6 (0.6)	8.2
600	1.6e−14	20.4	1.3e−15	3.9 (3.4)	7.9
1000	2.9e−14	110.3	1.3e−15	8.7 (7.2)	7.9
1400	3.7e−14	313.5	2.1e−15	15.8 (12.2)	7.9
1800	5.5e−14	702.2	1.7e−15	43.2 (34.4)	7.7
2200	6.3e−14	1296.6	1.7e−15	42.0 (26.1)	7.7
2600	7.0e−14	2143.2	1.8e−15	58.7 (34.5)	7.7
3000	7.8e−14	3299.6	1.9e−15	84.1 (44.3)	7.7

scaled linearizations, even if parameter scalings are employed [12]. This implies that linearization based algorithms, such as `quadeig`, cannot compute all eigenpairs backward stably, unless the same problem is solved twice using two different linearizations [24]. The proposed algorithm is "linearization free" and does not share this drawback. However, it is still worth investigating the performance on strongly damped problems. There are three reasons for this:

1. There are rank(D) eigenvalues that are far away from all starting points.
2. Eigenvalues may cluster around the origin [16].
3. The proposed algorithm make use of the Sherman-Morrison-Woodbury formula, which stability depends on the scale of the involved quantities [22].

We generated test problems using the following MATLAB code:

```
M = randn(250); M = M*M';
D = randn(250,r); D = s*(D*D');
K = randn(250); K = K*K';
```

TABLE 5.2
Backward errors and execution time for Algorithm 3. As in Table 5.1, Av. #upd denotes the number of average Ehrlich-Aberth updates per eigenvalue.

	$r = 5$		$r = 25$	
s	max $\eta(\lambda, v)$	Av. #upd	max $\eta(\lambda, v)$	Av. #upd
1e+00	1.1e−15	7.9	1.5e−15	17.0
1e+02	3.0e−15	6.9	1.4e−14	12.8
1e+04	1.8e−14	7.3	8.4e−13	17.0
1e+06	4.6e−14	7.2	8.2e−13	20.9
1e+08	3.7e−14	7.0	6.2e−13	26.0
1e+10	4.3e−14	7.8	1.4e−12	29.8
1e+12	2.3e−14	8.3	9.0e−13	36.1
1e+14	1.3e−14	9.0	3.7e−13	42.2

The results for different values of s and r are shown in Table 5.2. We see that the norm of D do affect the accuracy. However, the increase in the worst case backward error is modest (a factor 10 or 100) and appears to stagnate as s grows. The results in Table 5.2 can be explained as follows. When s is large, there are $2r$ real eigenvalues; half of them cluster around the origin and the other half are large and negative. As s grows, our algorithm fail satisfy the initial stopping condition for some eigenvalues, in particular the real ones, and therefore relaxes the tolerance (after 50 iterations). This is the reason for the growth in the worst case backward errors. It also explains the increase in average number of Ehrlich-Aberth steps taken per eigenvalue. We remark that taking more steps before relaxing the tolerance does not necessarily improve the accuracy. The problem is not that the iterates are "lost" and far away from the eigenvalues they should approximate, but rather that the Ehrlich-Aberth corrections, which are computed using the Sherman-Morrison-Woodbury formula, are not accurate enough. This is why some kind of tolerance relaxation is needed also if the eigenvalue backward error is used as stopping condition.

6. Discussion.

6.1. The large scale case. Todays models of vibrating structures are often so large that it becomes unfeasible to find all eigenpairs. Even in cases when it is possible, all eigenpairs are rarely of interest. Instead subspace based methods are used to target the most important eigenvalues. When a good subspace has been found, a smaller "projected" eigenproblem needs to be formed. There are several ways of forming this smaller problem. A good approach is to project directly onto the coefficient matrices, in style of an orthogonal Rayleigh-Ritz procedure [3]. This leads to a smaller system that shares the essential structure that all coefficient matrices are positive semidefinite. More precisely, if the columns of U span a computed subspace of dimension k, then we form

$$Q(\lambda) := U^T \left(M\lambda^2 + D\lambda + K \right) U,$$

which is a matrix polynomial of size $k \times k$. The next step is to find *all* eigenpairs of $Q(\lambda)$. If k is significantly larger than the number of discrete dampers (recall that less than 10 dampers is not uncommon in practice) then we have a problem on same form as (1.1) to which the proposed algorithm can be applied.

6.2. Generalizations. The main idea behind this work was that the structure "diagonal plus low rank" can be exploited to quickly compute eigenvalues and eigenvectors. We considered a rather special QEP, but it is also possible to apply this idea to other types of eigenvalue problems. A major obstacle, however, is the choice of starting points for the Ehrlich-Aberth iteration. Consider, for example, a rank one modification of a standard eigenvalue problems $Ax = \lambda x$. If we already have a spectral decomposition $A = S\Lambda S^{-1}$, then for any u and v, $S^{-1}(A + vu^T)S$ is the sum of a diagonal matrix and a rank one matrix, so we can apply the techniques discussed in this paper to quickly compute all its eigenpairs—*assuming good starting points are available.* The analog of the starting points used in our algorithm, would be the eigenvalues of A. Unfortunately, we cannot, without further insight into the problem, argue that this choice is any good. In fact, it can be arbitrarily bad: Ackermann's formula (for pole placement) states that a rank one modification—albeit an extreme one— is enough to change the spectrum of any given nonderogatory matrix arbitrarily.

One situation where good starting points are available appears in homotopy methods. Consider, for example, the following problem: Given a vector u and spectral decomposition $A = S\Lambda S^{-1}$ such that all eigenvalues of A have negative real part, find the smallest $t \geq 0$ such that $A + tuu^*$ has a purely imaginary eigenvalue. If we define $x = S^{-1}u$ and $y^* = u^*S$, then $\Lambda + txy^*$ is similar to $A + tuu^*$ and on the form "diagonal plus rank one." Hence one way to attack the problem is to solve a sequence of eigenvalue problems

$$\Lambda + t_i xy^*, \quad i = 0, 1, 2, \ldots, \quad \text{where} \quad 0 = t_0 < t_1 < t_2 < \cdots,$$

by an Ehrlich-Aberth iteration that exploits the structure and uses the eigenvalues of the previous step as starting points.

Acknowledgements. I thank Dario Bini, Vanni Noferini and Françoise Tisseur for helpful discussions. I also thank Vanni for sharing his MATLAB codes, and Françoise for reading the paper and giving valuable comments.

REFERENCES

[1] Structural applications of fluid viscous dampers. October 2012. Retrieved from `http://taylordevices.com/dampers-seismic-protection.html` Accessed: 2014-05-13.

[2] O. Aberth. Iteration methods for finding all zeros of a polynomial simultaneously. *Mathematics of Compution*, 27(112):339–344, 1973.

[3] Z. Bai and Y. Su. Soar: A second-order arnoldi method for the solution of the quadratic eigenvalue problem. *SIAM J. Matrix Anal. Appl.*, 26(3):640–659, 2005.

[4] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur. NLEVP: A collection of nonlinear eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)*, 39(2):7:1–7:28, 2013.

[5] D. A. Bini and V. Noferini. Solving polynomial eigenvalue problems by means of the ehrlich-aberth method. *Linear Algebra and its Applications*, 439(4):1130–1149, 2013.

[6] L. W. Ehrlich. A modified newton method for polynomials. *Communications of ACM*, 10(2):107–108, 1967.

[7] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 4th edition, 2013.

[8] S. Hammarling, C. J. Munro, and F. Tisseur. An algorithm for the complete solution of quadratic eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)*, 38(3):18:1–18:19, 2013.

[9] N. J. Higham, D. S. Mackey, F. Tisseur, and S. D. Garvey. Scaling, sensitivity and stability in the numerical solution of quadratic eigenvalue problems. *Internat. J. Numer. Methods Eng.*, 73(3):344–360, 2008.

[10] N. J. Higham and F. Tisseur. A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. *SIAM J. Matrix Anal. Appl.*, 21(4):1185–1201, 2000.
[11] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.
[12] W.-W. L. Hung-Yuan Fan and P. V. Dooren. Normwise scaling of second order polynomial matrices. *SIAM J. Matrix Anal. Appl.*, 26(1):252–256, 2004.
[13] P. Lancaster. *Lambda-matrices and Vibrating Systems*. Dover Publications, inc., New York, USA, 2002.
[14] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann. Möbius transformations of matrix polynomials. MIMS EPrint 2014.2, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, January 2014.
[15] G. Peters and J. H. Wilkinson. Inverse iteration, ill-conditioned equations and newton's method. *SIAM Review*, 21(3):339–360, 1979.
[16] L. Taslaman. Strongly damped quadratic matrix polynomials. MIMS EPrint 2014.10, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, March 2014.
[17] F. Tisseur. Backward stability of the QR algorithm. Technical report 239, Equipe d'Analyse Numerique, Université Jean Monnet de Saint-Etienne, Cedex 02, France, October 1996.
[18] F. Tisseur. Backward error and condition of polynomial eigenvalue problems. *Linear Algebra and its Applications*, 309:339–361, 2000.
[19] F. Tisseur. Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 22(4):1038–1057, 2001.
[20] K. Veselić. *Damped Oscillations of Linear Systems*. Springer-Verlag, Berlin Heidelberg, Germany, 2011.
[21] S. Wang and S. Zhao. An algorithm for $Ax = \lambda Bx$ with symmetric and positive-definite A and B. *SIAM J. Matrix Anal. Appl.*, 12(4):654–660, 1991.
[22] E. L. Yip. A note on the stability of solving a rank-p modification of a linear system by the sherman-morrison-woodbury formula. *SIAM J. Sci. Stat. Comput.*, 7(2):507–513, 1986.
[23] I. Zaballa and F. Tisseur. Finite and infinite elementary divisors of matrix polynomials: a global approach. MIMS EPrint 2012.78, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, August 2012.
[24] L. Zeng and Y. Su. A backward stable algorithm for quadratic eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 35(2):499–516, 2014.