VERIFIED ERROR BOUNDS FOR ALL EIGENVALUES AND EIGENVECTORS OF A MATRIX

SIEGFRIED M. RUMP*

Abstract. A verification method is presented to compute error bounds for all eigenvectors and eigenvalues including clustered and/or multiple ones of a general, real or complex matrix. In case of a narrow cluster error bounds for an invariant subspace are computed because computation of a single eigenvector may be ill-posed.

Computer algebra and verification methods have in common that the computed results are correct with mathematical certainty. Unlike a computer algebra method, a verification method may fail in the sense that only partial or no inclusions at all are computed. That may happen for very ill-conditioned problems being too sensitive for the arithmetical precision in use. That cannot happen for computer algebra methods which are "never-failing" because potentially infinite precision is used. In turn, however, that may slow down computer algebra methods significantly and may impose limitations on the problem size. In contrast, verification methods solely use floating-point operations so that their computing time and treatable problem size is of the order of that of purely numerical algorithms.

For our problem it is proved that the union of the eigenvalue bounds contains the whole spectrum of the matrix, and bounds for corresponding invariant subspaces are computed. The computational complexity to compute inclusions of all eigenpairs of an $n \times n$-matrix is $\mathcal{O}(n^3)$.

Key words. Verification method, eigenvalue clusters, eigenvector, invariant subspace, all eigenpairs, INTLAB

AMS subject classifications. 65F05, 65G20, 65G50

1. Introduction.

This note presents a verification method to compute error bounds for all eigenvectors and eigenvalues of a general, real or complex matrix. In case of clustered and/or multiple eigenvalues an inclusion of a basis of the corresponding invariant subspace is computed. This means that the computed bounds do contain the true result with mathematical certainty. In particular, it is proved that the computed bounds cover all eigenpairs of the matrix.

Verification methods [17, 22, 18] are completely rigorous, including all procedural and/or rounding errors, however, they are restricted to well-posed problems. For example, it is possible to prove that a small circle in the complex plane contains two eigenvalues of a matrix, but verification methods do not allow to prove that it contains a double eigenvalue because that problem is ill-posed: an arbitrarily small perturbation of the input matrix may have two simple eigenvalues. Similarly, it is not possible to prove reasonably sharp bounds for an eigenvector of an eigenvalue of algebraic multiplicity 2, regardless of the geometric multiplicity. That is even true for a symmetric input matrix where, in contrast, very sharp bounds of eigenvalues follow directly by perturbation bounds.

Verification methods should not be confused with computer algebra methods. The latter provide never-failing algorithms. This is true independent of the condition number because computer algebra methods potentially work in infinite precision. Both approaches have in common that the computed results are always mathematically correct. Verification methods intentionally use floating-point arithmetic for speed with the drawback that failures occur, meaning, that no inclusion can be computed. In a good verification method that happens only if the problem is too hard, i.e., ill-conditioned for the floating-point accuracy available.

*Institute for Reliable Computing, Hamburg University of Technology, Am Schwarzenberg-Campus 3, 21073 Hamburg, Germany, and Visiting Professor at Waseda University, Faculty of Science and Engineering, 3–4–1 Okubo, Shinjuku-ku, Tokyo 169–8555, Japan (rump@tuhh.de).

The first verification method for the algebraic eigenproblem is Krawczyk's paper [12] which applies his method for nonlinear problems [12] to $Ax - \lambda x = 0$ with some normalization of x. Krawczyk's method, however, is a refinement of given bounds. Moore proposed to use Brouwer's fixed point theorem [16] and proof of nonsingularity of some matrix to derive an existence test. Krawczyk's operator and Moore's ansatz are already contained in [10, p. 12f].

In [19] this method was improved in three ways: An interval iteration with so-called epsilon-inflation computes an inclusion if the problem is not too ill-conditioned, the proof of nonsingularity is omitted by requiring a self-mapping into the interior in Brouwer's fixed point theorem, and an inclusion for the error with respect to an approximation is computed rather than an inclusion of the solution itself. Those three techniques are today's standard for verification methods from the solution of linear systems to partial differential equations. For an overview see [22, 18].

Based on that, in [19] a verification method for one eigenvalue/eigenvector pair of a real or complex $n \times n$ matrix is introduced with complexity $\mathcal{O}(n^3)$. One might apply that method n times, but besides the complexity $\mathcal{O}(n^4)$ that fails for multiple eigenvalues and cannot guarantee that all eigenvalues are covered. Several publications concentrate on verified error bounds on one eigenpair, for example [28, 4, 5, 9, 26, 25]; in [2] a method is introduced for double eigenvalues.

Historically, the next step are verification methods for multiple eigenvalues and corresponding invariant subspaces introduced in [21]. Bounds are computed regardless of the Jordan structure, but for only one cluster. Verification methods for all eigenpairs of a symmetric positive definite matrix are given in [15].

In this note we are interested in bounds for all eigenpairs of a general real or complex matrix. The first and seemingly only paper to that is Miyajima's [14], which is based on nonlinear matrix equations derived from the eigenproblem with special emphasis on multiple eigenvalues including the defective case. The nonlinear system and Newton correction are similar to [21]. His paper gives two methods. The first method uses numerical spectral decomposition and computes eigenvalue inclusions based on Gershgorin circles of the preconditioned matrix. Then $A - \lambda I$ is singular for λ in an eigenvalue inclusion, so that an eigenvector inclusion follows by solving a linear system with one row and column of $A - \lambda I$ deleted. The linear system is solved using a transformation of variables. For clusters, a basis of an invariant subspace is enclosed by solving the nonlinear matrix equation. The second method uses numerical block diagonalization analogous to [3], in which a numerical Jordan decomposition of A is included. The decomposition is known to be ill-posed, occasionally leading to computational problems. The method encloses all eigenvalues and eigenvectors (bases of invariant subspaces in the cluster case) by solving the nonlinear matrix equation. Miyajima's methods are also suitable for the generalized algebraic eigenproblem. That is also true for our method, but for simplicity we refrain from presenting that.

The approach presented in this note is based on [21]. As for Miyajima's approach, inclusions for some or all eigenspaces may be computed. In contrast, we do not rely on a numerical Jordan decomposition. In the next section the method will be presented, and in the final section on computational results we show the new method to be faster and more stable than Miyajima's.

2. Main result. Denote by $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ the field of real or complex numbers, and by $\mathbb{K}^n$ and $\mathbb{K}^{n,k}$ the set of n-vectors and $n \times k$ matrices over $\mathbb{K}$, respectively. We also use the short notation $M_{n,k}$ to denote a (real or complex) $n \times k$ matrix, and M_n if $k = n$.

Denote by $\mathbb{I}$ a set of real or complex intervals, then similarly we write $\mathbb{I}^n$ for interval n-vectors and $\mathbb{I}^{n,k}$ for $n \times k$ interval matrices. We will use boldface letters for interval quantities, and denote by $int(\cdot)$ the topological interior. The $n \times n$ identity matrix is denoted by I_n, where the subindex is omitted if clear from the context, and its j-th column by e_j. Entrywise matrix multiplication (the Hadamard product) is denoted by $\circ$.

The principles of verification methods and interval arithmetic can be found in [17, 22, 18]. However, we only use that interval operations $op \in \{+, -, \cdot, /\}$ are defined such that for compatible interval quantities $\mathbf{A}, \mathbf{B}$ the inclusion property

$$\forall A \in \mathbf{A}\ \forall B \in \mathbf{B}\colon \qquad A\ op\ B \in \mathbf{A}\ op\ \mathbf{B} \tag{2.1}$$

is satisfied. Intervals may be represented by infimum-supremum or midpoint-radius, where the former is often used for real and the latter for complex intervals. The representation may influence the quality of the bounds, however, the only important property for the mathematical correctness is (2.1).

Concerning notation, we write $\mathrm{mig}(\mathbf{A}) := \min\{|a|\colon a \in \mathbf{A}\}$ and $\mathrm{mag}(\mathbf{A}) := \max\{|a|\colon a \in \mathbf{A}\}$ for a scalar interval $\mathbf{A}$, and the definition extends to interval vectors and matrices entrywise. Properties like, for example, being non-singular extend to an interval matrix $\mathbf{A}$ by requesting that all $A \in \mathbf{A}$ are non-singular.

We will use Matlab notation [13] and INTLAB [20], the Matlab-Octave toolbox for reliable computing. In INTLAB, real interval quantities are represented by infimum-supremum, complex interval quantities by midpoint-radius, but the following applies, *mutatis mutandis*, to other representations as well.

In [21, Theorem 3.2] we proved the following.

THEOREM 2.1. *Let $A, R \in M_n$, $\widetilde{X} \in M_{n,k}$, $\tilde{\lambda} \in \mathbb{K}$ and $\mathbf{X} \in \mathbb{I}^{n,k}$ be given. Define $[n] := \{1, \ldots, n\}$, and for non-empty $\mu \subseteq [n]$ with $k := |\mu|$ denote a partition of the $n \times n$ identity matrix I by*

$$V := I(:, \mu) \in M_{n,k} \quad \text{and} \quad U = I(:, [n] \backslash \mu) \in M_{n,n-k}. \tag{2.2}$$

It follows that $UU^T + VV^T = I$. Define

$$f(\mathbf{X}) := -R(A\widetilde{X} - \tilde{\lambda}\widetilde{X}) + \{I - R\big((A - \tilde{\lambda}I)UU^T - (\widetilde{X} + UU^T \cdot \mathbf{X})V^T\big)\} \cdot \mathbf{X}. \tag{2.3}$$

Suppose

$$f(\mathbf{X}) \subseteq int(\mathbf{X}). \tag{2.4}$$

Then there exists $\widehat{M} \in M_k$ with $\widehat{M} \in \tilde{\lambda} I_k + V^T\mathbf{X}$ such that the Jordan canonical form of $\widehat{M}$ is identical to a $k \times k$ principal submatrix of the Jordan canonical form of A, and there exists $\widehat{Y} \in M_{n,k}$ with $\widehat{Y} \in \widetilde{X} + UU^T\mathbf{X}$ such that $\widehat{Y}$ spans the corresponding invariant subspace of A, i.e., $A\widehat{Y} = \widehat{Y}\widehat{M}$.

The $k \times k$ submatrix of $\mathbf{X}$ with rows in μ is the inclusion of a Jordan block of A shifted by $\tilde{\lambda}$, where for the invariant subspace that $k \times k$ submatrix is replaced by the corresponding submatrix of $\widetilde{X}$ for normalization.

Theorem 2.1 gives a sufficient criterion for $\mathbf{X} \in \mathbb{I}^{n,k}$ to contain $\widehat{M}$ and $\widehat{Y}$ with the described properties. The better the approximations $\tilde{\lambda}, \widetilde{X}$ for the eigenspace, the preconditioner R and the choice of $\mathbf{X}$, the more likely condition (2.4) is satisfied. If (2.3) is not satisfied for the initial $\mathbf{X}$, then an interval iteration with epsilon-inflation

starting with $\mathbf{X} := f(\mathbf{X})$ is applied, see [21]. The matrix R is an approximate inverse of the Jacobian of the underlying Newton iteration for $\mathbf{X}$. Note that the assertions are true regardless of the quality of $\widetilde{\lambda}, \widetilde{X}$ and R. The computational effort is $\mathcal{O}(n^3)$.

The quality of the inclusions depend on the Jordan structure of the matrix. There is a huge literature on the sensitivity of eigendecompositions, among them [23, 24, 1, 7, 11]. As a rule of thumb the sensitivity of an eigenvalue λ is of the order $\mathbf{u}^{1/k}$ for $\mathbf{u}$ denoting the relative rounding error unit and k the size its largest Jordan block [27, Section 2.23]. As a consequence, this is the minimum width of an inclusion of λ when using a floating-point arithmetic with relative rounding error unit $\mathbf{u}$.

Let $B \in M_n$. In order to derive an $\mathcal{O}(n^3)$ method for computing inclusions of some or all eigenpairs, the purpose of this note, we compute an approximate eigendecomposition `[V,D]=eig(B)` of B. Suppose V is nonsingular and apply Theorem 2.1 to $V^{-1}BV$, then inclusions of the spectrum and Jordan structure of B are obtained.

If B is not defective and V its exact eigenbasis, then $V^{-1}BV$ is diagonal. But even if B has multiple eigenvalues of geometric multiplicity 1, we may expect that computationally $V^{-1}BV$ is almost diagonal because of the following. For a $k \times k$ Jordan block to an eigenvalue λ, a numerical algorithm like Matlab's `eig` tends to compute a basis of the corresponding invariant subspace with all columns close to a multiple of the corresponding eigenvector. The sensitivity of λ and the condition number of the corresponding columns of V is about $\mathbf{u}^{1/k}$. A numerically singular eigenapproximation matrix V may occur, but to our experience only in special cases or when searching for it. For numerical evidence see the beginning of Section 3.

Therefore a reasonable approximation to the eigenvectors of $V^{-1}BV$ is the identity matrix. For each eigenvalue or cluster of eigenvalues, the corresponding matrix R in Theorem 2.1 is close to diagonal. Therefore we can combine the application of Theorem 2.1 to a set of eigenvalues or clusters into one matrix to achieve a total computing time of $\mathcal{O}(n^3)$.

The matrix $V^{-1}BV$ is usually not representable in floating-point, so that we have to work with an inclusion $\mathbf{A}$ of it. In verification methods it is a standard procedure to replace A in Theorem 2.1 by an interval matrix $\mathbf{A}$ and to conclude that, due to the inclusion property (2.1), all assertions are true for all $A \in \mathbf{A}$, in particular for $V^{-1}BV$. The eigenvalues and Jordan structure of B and $V^{-1}BV$ coincide, and the eigenvector inclusions transform by V. An inclusion $\mathbf{A}$ of $V^{-1}BV$ is computed by standard verification methods [22] including the proof that V is nonsingular. A corresponding INTLAB command is `A = verifylss(V,intval(A)*V)`. Here `verifylss(A,b)` is a verified inclusion of the linear system $Ax = b$, where the type cast `intval(A)` assures that the right hand side `intval(A)*V` is an inclusion of AV. For details see [20, 22].

Before we can state our main result, we need some notation. For simplicity we state our theorem for the computation of inclusions of all eigenvalues and corresponding invariant subspaces. If only partially successful, i.e., for some eigenvalues the verification failed, it will be clear how to state and apply the theorem only to these remaining eigenvalues. We refrain to state the result for such a partial set because the notation would be even more involved.

As before denote $[n] := \{1, \ldots, n\}$. For $1 \le m \le n$, let $\mu_1 \cup \ldots \cup \mu_m = [n]$ be a partition of $[n]$, i.e., $\mu_i \cap \mu_j = \emptyset$ for $i \ne j$. Then, for given $i \in [m]$ and $k := |\mu_i|$, the splitting of I into columns within and outside μ_i is denoted by $V_i := I(:, \mu_i) \in M_{n,k}$ and $U_i = I(:, [n] \backslash \mu_i) \in M_{n,n-k}$. Multiplying a matrix from the left by $V_i^T V_i$ sets all rows outside μ_i to zero, from the right the columns outside μ_i. Furthermore, $V_i^T V_j$ is

the zero matrix for $i \neq j$ and

$$U_i U_i^T + V_i V_i^T = I_n \quad \text{and} \quad V_i V_i^T V_i = V_i \quad \text{and} \quad U_i^T V_i = 0 \qquad \text{for } i \in [m]. \tag{2.5}$$

For a matrix $C \in M_n$ we define $C_{\mathcal{D}} \in M_n$ to be the block diagonal matrix composed of the μ_i-blocks of C, i.e., $C_{\mathcal{D}} := \sum_{i=1}^m V_i V_i^T C V_i V_i^T$. The subindex "$\mathcal{D}$" serves as an operator on C and applies to interval matrices as well; it means to extract the elements in the diagonal μ_i-blocks. Similarly we define $C_{\mathcal{O}} := C - C_{\mathcal{D}}$ to be the matrix of complement indices of C. Then (2.5) and $\sum_{i=1}^m V_i V_i^T = I_n$ yield

$$C_{\mathcal{O}} := \sum_{i=1}^m U_i U_i^T C V_i V_i^T. \tag{2.6}$$

Again, the subindex $\mathcal{O}$ serves as an operator to C and applies also to interval matrices.

THEOREM 2.2. *We use the notation just given. Let $A \in M_n$, let mutually distinct $\tilde{\lambda}_i \in \mathbb{K}$ for $i \in [m]$ be given, and let $D \in M_n$ be a diagonal matrix with $D_{jj} = \tilde{\lambda}_i$ for all $i \in [m]$ and for all $j \in \mu_i$. Then $D_{jj} \neq \tilde{\lambda}_i$ for all $j \notin \mu_i$ and*

$$DV_i = \tilde{\lambda}_i V_i \quad \text{for } i \in [m]. \tag{2.7}$$

Furthermore, let diagonal $R_i \in M_n$ be such that

$$R_i(D - \tilde{\lambda}_i I)U_i = U_i \quad \text{and} \quad R_i V_i = -V_i \qquad \text{for } i \in [m]. \tag{2.8}$$

Note that R_i is well defined because the $\tilde{\lambda}_i$ are mutually distinct. Let $\widetilde{R} \in M_n$ be such that

$$\forall i \in [m]: \quad j \in \mu_i \quad \Rightarrow \quad \widetilde{R} e_j = \operatorname{diag}(R_i). \tag{2.9}$$

Let $\mathbf{X} \in \mathbb{I}^{n,n}$ be given, define $E := A - D$ and set $\mathbf{Y} := \mathbf{X}_{\mathcal{O}}\mathbf{X}_{\mathcal{D}} - E - E\mathbf{X}_{\mathcal{O}}$, where $\mathbf{X}_{\mathcal{O}}$ and $\mathbf{X}_{\mathcal{D}}$ are the matrices extracted from $\mathbf{X}$ according to the index sets μ_i.

Then for each $i \in [m]$ the following is true. If

$$\mathbf{Z}_i := (\widetilde{R} \circ \mathbf{Y})V_i \subseteq \mathit{int}(\mathbf{X}V_i) \quad \text{for some } i \in [m], \tag{2.10}$$

then there exists a Jordan block $\widehat{M}_i \in \tilde{\lambda}_i I + V_i^T \mathbf{Z}_i$ with corresponding invariant subspace $\widehat{Y}_i \in V_i + U_i U_i^T \mathbf{Z}_i$ of A, i.e., $A\widehat{Y}_i = \widehat{Y}_i \widehat{M}_i$.

Denote the set of $i \in [m]$ satisfying (2.10) by Φ, define $\mathcal{J} := \bigcup\{\mu_i : i \in \Phi\}$ and $k := \sum_{i \in \Phi} |\mu_i| = |\mathcal{J}|$. Denote by $\mathbf{Z} \in \mathbb{I}^{k,k}$ the matrix collected of rows and columns $i \in \mathcal{J}$ of $\widetilde{R} \circ \mathbf{Y}$, and suppose $\max\{\varrho(Z) : Z \in \mathbf{Z}\} < 1$. Then for each $i \in \Phi$ the matrix $\widehat{M}_i$ corresponds to an individual Jordan block of A. If $k = n$, then $\bigcup\{\mathit{spec}(\widehat{M}_i) : i \in \Phi\} = \mathit{spec}(A)$, where $\mathit{spec}(A)$ denotes the spectrum of the matrix A.

Remark 2.3. The main point is that the $\mathbf{Z}_i$ in (2.10) are computed in one matrix $\mathbf{Z} := \widetilde{R} \circ \mathbf{Y}$ and $\mathbf{Z} \subseteq \mathit{int}(\mathbf{X})$ is checked. That requires $\mathcal{O}(n^2)$ operations, only the transformation $V^{-1}BV$ at the beginning costs $\mathcal{O}(n^3)$ operations. The eigenvalue and eigenvector inclusions correspond to the columns μ_i in $\mathbf{Z}$, where $|\mu_i| > 1$ for a cluster or multiple eigenvalue. Inclusions for a subset of k eigenvalues or clusters are obtained by computing only the corresponding columns of $\widetilde{R} \circ \mathbf{Y}$ in $\mathcal{O}(nk)$ operations.

Remark 2.4. The assumption $\max\{\varrho(Z) : Z \in \mathbf{Z}\} < 1$ may be certified by Perron-Frobenius theory and $\varrho(\operatorname{mag}(\mathbf{Z})) < 1$, or by $\max\{\|\mathbf{Z}\|_\infty, \|\mathbf{Z}\|_1\} < 1$. Fortunately, that test needs only $\mathcal{O}(n^2)$ operations.

Remark 2.5. The first part of Theorem 2.2 assures that for $i_1, i_2 \in \Phi$ there are matrices $\widehat{M}_{i_1}$ and $\widehat{M}_{i_2}$ corresponding to a Jordan block of the matrix A, respectively, and the last assertion certifies that these are different Jordan blocks. We think that this is always true, also without the additional condition $\max\{\varrho(Z)\colon Z \in \mathbf{Z}\} < 1$, but could not prove it.

Proof. Let $i \in [m]$ be fixed but arbitrary. Then (2.9) implies for every $T \in M_n$

$$(\widetilde{R} \circ T)V_i = R_i T V_i. \tag{2.11}$$

Let $X \in \mathbf{X}$ be fixed but arbitrary, and abbreviate $X_i := XV_i$. Then (2.5) and the definition of $X_{\mathcal{D}}$ and $X_{\mathcal{O}}$ imply

$$X_{\mathcal{D}}V_i = V_i V_i^T X_i \;\text{ and }\; X_{\mathcal{O}}V_i = X_i - V_i V_i^T X_i = U_i U_i^T X_i. \tag{2.12}$$

Moreover,

$$X_{\mathcal{O}}X_{\mathcal{D}}V_i = X_{\mathcal{O}}V_i V_i^T X_i = U_i U_i^T X_i V_i^T X_i. \tag{2.13}$$

Using (2.7) and $A = D + E$ give

$$R_i(A - \tilde{\lambda}_i I)V_i = R_i(D - \tilde{\lambda}_i I)V_i + R_i E V_i = R_i E V_i \;, \tag{2.14}$$

so that (2.8) and (2.5) imply

$$\begin{aligned} \{I - R_i((A - \tilde{\lambda}_i I)U_i U_i^T - (V_i + U_i U_i^T X_i)V_i^T)\}X_i &= \\ X_i - R_i((D - \tilde{\lambda}_i I)U_i U_i^T + E U_i U_i^T - V_i V_i^T - U_i U_i^T X_i V_i^T)X_i &= \\ X_i - (U_i U_i^T + R_i E U_i U_i^T + V_i V_i^T - R_i U_i U_i^T X_i V_i^T)X_i &= \\ -R_i(E U_i U_i^T - U_i U_i^T X_i V_i^T)X_i \;. & \end{aligned} \tag{2.15}$$

Setting $Y := X_{\mathcal{O}}X_{\mathcal{D}} - E - EX_{\mathcal{O}} \in \mathbf{Y}$ and using (2.11), (2.13), (2.12), (2.14) and (2.15) proves

$$\begin{aligned} (\widetilde{R} \circ Y)V_i &= R_i(X_{\mathcal{O}}X_{\mathcal{D}} - E - EX_{\mathcal{O}})V_i = R_i(U_i U_i^T X_i V_i^T X_i - EV_i - EU_i U_i^T X_i) \\ &= -R_i(A - \tilde{\lambda}_i I)V_i + \{I - R_i((A - \tilde{\lambda}_i I)U_i U_i^T - (V_i + U_i U_i^T X_i)V_i^T)\}X_i. \end{aligned}$$

Since this is true for every $X \in \mathbf{X}$, it follows that the assumptions of Theorem 2.1 with $\widetilde{X} := V_i$ and $\mathbf{X}$ replaced by $\mathbf{X}V_i$ are satisfied. This proves the first part of the theorem.

Denote by $\widehat{Y} \in M_{n,k}$ the matrix collected of block columns $\widehat{Y}_i$ for $i \in \Phi$. For $V \in M_{n,k}$ denoting the matrix of corresponding columns of the identity matrix, it follows $\widehat{Y} = V + Z$ for some $Z \in \mathbf{Z}$, so that $\varrho(Z) < 1$ implies that $\widehat{Y}$ has full rank. Therefore, $\widehat{Y}$ is a basis of an invariant subspace of A, and all assertions follow. ∎

The interval matrix $\mathbf{Z}$ contains inclusions of the error of the approximate eigenvalues and -vectors. Thus, the intervals can be expected to be narrow, one of the three main principles of verification methods mentioned in the introduction. Therefore it is most likely that the final condition $\max\{\varrho(Z)\colon Z \in \mathbf{Z}\} < 1$ is satisfied.

Following we give some implementation details. Let a matrix $B \in M_n$ be given, then Theorem 2.2 is applied to a similarity transformation of B as follows.

```
function [A,W] = transform(B)
  [W,X] = eig(B);
  X = X + W\prodK(B,W,-W,X);
  [Res,E] = prodK(B,W,-W,X);
  A = X + W\midrad(Res,E);
```

The approximate eigenvalue matrix `X` is improved by one Newton iteration. Here `prodK` in line 2 with one output parameter computes an accurate approximation of the residual $BW-WX$ using error-free transformations, and in line 3 with two output parameters an inclusion with midpoint `Res` and radius `E`, wherefore `midrad(Res,E)` is an inclusion of $BW - WX$.

As always in numerical analysis the residual should be calculated accurately. In Algorithm `transform` we use error-free transformations which are accurate but costly. Although using BLAS3 routines, as always in INTLAB, that takes about a third of the total computing time of the inclusion Algorithm `verifyeigall` to be presented. In the final line `W\midrad(Res,E)` computes an inclusion of $\{W^{-1}R\colon R \in \texttt{midrad}(\texttt{Res},\texttt{E})\}$, so that $\mathbf{A}$ is an inclusion of $X + W^{-1}(BW - WX) = W^{-1}BW =: A$. Hence, the spectra of B and A are identical, and the invariant subspaces transform by W.

Note that the input matrix B in the function `transform` may be an interval matrix $\mathbf{B}$ as well with assertions being true for all $B \in \mathbf{B}$. In that case use `[W,X] = eig(B.mid)` . But not much cancellation is expected in the computation of the residuals, so that `prodK(B,W,-W,X)` can be replaced by `B*W-W*X`. Multiple eigenvalues with small geometric multiplicity become sensitive to perturbations, and even narrow interval components of $\mathbf{B}$ may widen the computed inclusions of eigenvalues and eigenvectors significantly.

We then apply Theorem 2.2 to the interval matrix $\mathbf{A}$. By the inclusion principle (2.1) it follows that the assertions are true for all matrices within $\mathbf{A}$, in particular for $A = W^{-1}BW$. Hence (2.10) for some $i \in [m]$ implies that $B\widehat{Y}_i = \widehat{Y}_i\widehat{M}_i$ for $\widehat{M}_i \in \tilde{\lambda}_i I + V_i^T\mathbf{Z}_i$ and $\widehat{Y}_i \in (V_i + U_iU_i^T\mathbf{Z}_i)W$. By Perron-Frobenius theory, $\varrho(M) \le \varrho(|M|)$ for every matrix M. Thus, the eigenvalues of $\widehat{M}_i$ are included in $\tilde{\lambda}_i \pm \varrho(|V_i^T\mathbf{Z}_i|)$, where the spectral radius is bounded as in [21] by a few power iterations and Collatz's inclusion [8].

Based on that our algorithm is as follows, partly using Matlab and INTLAB notation.

function [L,X,mu] = verifyeigall(B)
1) Calculate A and W by Algorithm `transform`
2) `normA = norm(A.mid, inf); d = diag(A);`
   `dist = (mig(d - d.') <= 1e - 14 * normA);`
   `[mu, binsizes] = conncomp(graph(dist),'OutputForm','cell');`
   `J = find(binsizes > 1);`
3) `D = d.mid;`
   `E = A - diag(D);`
4) `RR = 1./(D - D.');`
   `RR(1 : n + 1 : n^2) = -1;`
   `for j = J, RR(mu{j}, mu{j}) = -1; end`
5) `Y = -RR. * E; cols = 0;`
6) `repeat`
   cols_old = cols;
   Compute an epsilon-inflation X of Y
   Compute $X_{\mathcal{D}}$ and $X_{\mathcal{O}}$ according to Theorem 2.2
   cols = # of columns satisfying (2.10)
   If cols=n then `[L,X] = final(D,Y,W)`, return
   If cols < cols_old, then apply verifyeigall recursively

In step 1) the input matrix B is transformed as described before. Step 2) computes

connected components of the graph of the matrix of distances of diagonal elements of A which is a guess of the Jordan structure. In step 3) the interval matrix $\mathbf{A}$ is splitted into $D + \mathbf{E}$ with $D \in M_n$ and $\mathbf{E} \in \mathbb{I}^{n,n}$. Thus for $A \in \mathbf{A}$ there exists $E \in \mathbf{E}$ with $A = D + E$. Note that the diagonal elements of $\mathbf{E}$ are, in general, nonzero.

Step 4) computes the matrix $\widetilde{R}$ as in Theorem 2.2. The main loop is in step 6). The loop stops with success if (2.10) is satisfied for all columns, where Algorithm `final` computes bounds for the eigenvalues of $\widehat{M_i}$ and transforms the invariant subspaces using W. If the number of successful columns does not increase, the function `verifyeigall` is applied recursively to the columns with no inclusion. The algorithm terminates if that recursion does not increase the number of successful columns.

The output `mu` identifies the clusters, i.e., L_j for $j \in \mu_i$ and $X(:, \mu_i)$ form an inclusion of an eigenvalue and corresponding invariant subspace. If the union of the indices in `mu` is equal to $\{1, \ldots, n\}$, then inclusions for all eigenvalues and invariant subspaces have been computed.

3. Numerical results. All computational results are produced using Matlab and INTLAB and double precision (binary64) with a relative rounding error unit $\mathbf{u} = 2^{-53} \approx 10^{-16}$ on a standard laptop. The relative error of an interval is the maximum relative distance between two members of the interval, so that a value of order 10^{-16} means that an inclusion is almost maximally accurate.

We start with some general remarks on the construction of test examples. Consider the set $\mathcal{D}_n \subseteq M_n$ of matrices with double eigenvalue, which is of measure zero within M_n. The set of diagonalizable matrices within $\mathcal{D}_n$ is again of measure zero. As a consequence, we may expect that if there are non-trivial Jordan blocks, they belong to mutually different eigenvalues.

Let `J = diag(randn(n,1))` and replace a $k \times k$ block with all diagonal elements equal to one random number λ and 1's on the superdiagonal. Let V be a nonsingular matrix, then $V^{-1}JV$ has a $k \times k$ Jordan block to the eigenvalue λ.

When computing $V^{-1}JV$ in floating-point arithmetic, likely the resulting matrix A has a cluster of eigenvalues with center not far from λ. In fact, it needs some effort to construct a matrix within $\mathbb{F}_n$ with multiple eigenvalues, see Subsection 3.2. The radius of the cluster is usually close to the sensitivity of the multiple eigenvalue, which is $\mathbf{u}^{1/k}$. For example, this attempt to construct a matrix with 3-fold eigenvalue generates a matrix with a cluster of radius 10^{-5}.

We therefore split this section of computational results into a first part with matrices the eigenvalues of which are generated as described above, and a second part using special methods to generate matrices with truly multiple eigenvalues and specified Jordan blocks. In the last subsection we compare our new algorithm with Miyajima's methods in [14].

3.1. Numerical results for eigenvalue clusters. Suppose the input matrix B has multiple eigenvalues. Then `[W,D] = eig(A)` produces W with almost linearly dependent columns for each Jordan block, and one may expect W to be ill-conditioned. However, those columns are only linear dependent up to the sensitivity of the clusters: a $k \times k$ Jordan block produces eigenvector approximations becoming linearly dependent for a perturbation of order $\mathbf{u}^{1/k}$. In that sense floating-point arithmetic has a regularizing effect, the condition number of the eigenvector approximation matrix is of the order $\mathbf{u}^{-1/k}$.

We first compare the accuracy of the eigenvalue inclusions by `verifyeigall` with those of Gershgorin circles. The latter provide verified inclusions of the eigenvalues, but not of eigenvectors. In that respect the comparison is not fair. However, as an

advantage, the Gershgorin approach cannot fail.

The computational results displayed in Table 1 are as follows. For different dimensions n, we generate a random matrix being diagonal except one Jordan block of size k (which is just a random matrix for $k = 1$), and perform a similarity transformation by some random matrix. That is one test matrix B. Then inclusions of all eigenvalues are first computed by Gershgorin circles applied to `transform(B)`, and second by `verifyeigall(B)`. The results displayed in Table 1 are the mean of all means, the median of all medians and the maximum of all maxima of the relerr of the inclusions, calculated over 100 samples. In this test set `verifyeigall` could not compute inclusions of all eigenvalues in 1 out of the 100 test cases for $n = 1000$ and $k = 3$.

		relerr Gershgorin			relerr new		
n	k	mean	median	max	mean	median	max
10	1	$7.4 \cdot 10^{-13}$	$1.8 \cdot 10^{-14}$	$9.6 \cdot 10^{-11}$	$2.3 \cdot 10^{-16}$	$2.3 \cdot 10^{-16}$	$3.3 \cdot 10^{-16}$
	2	$7.5 \cdot 10^{-7}$	$3.0 \cdot 10^{-14}$	$1.9 \cdot 10^{-4}$	$6.4 \cdot 10^{-8}$	$3.0 \cdot 10^{-16}$	$7.1 \cdot 10^{-6}$
	3	$3.9 \cdot 10^{-5}$	$3.9 \cdot 10^{-14}$	$1.0 \cdot 10^{-3}$	$1.5 \cdot 10^{-5}$	$3.6 \cdot 10^{-16}$	$1.1 \cdot 10^{-3}$
100	1	$2.5 \cdot 10^{-10}$	$1.9 \cdot 10^{-12}$	$5.2 \cdot 10^{-7}$	$2.1 \cdot 10^{-16}$	$2.1 \cdot 10^{-16}$	$3.3 \cdot 10^{-16}$
	2	$8.6 \cdot 10^{-7}$	$2.2 \cdot 10^{-12}$	$1.5 \cdot 10^{-3}$	$1.5 \cdot 10^{-8}$	$3.0 \cdot 10^{-16}$	$1.8 \cdot 10^{-5}$
	3	$2.9 \cdot 10^{-4}$	$2.7 \cdot 10^{-12}$	$2.3 \cdot 10^{-1}$	$6.5 \cdot 10^{-6}$	$3.2 \cdot 10^{-16}$	$2.4 \cdot 10^{-3}$
1000	1	$2.0 \cdot 10^{-7}$	$7.3 \cdot 10^{-11}$	$4.3 \cdot 10^{-3}$	$1.9 \cdot 10^{-16}$	$1.9 \cdot 10^{-16}$	$3.3 \cdot 10^{-16}$
	2	$3.3 \cdot 10^{-6}$	$7.3 \cdot 10^{-11}$	$8.8 \cdot 10^{-2}$	$8.3 \cdot 10^{-9}$	$3.1 \cdot 10^{-16}$	$2.0 \cdot 10^{-4}$
	3	$2.4 \cdot 10^{-4}$	$7.9 \cdot 10^{-11}$	$1.3 \cdot 10^{0}$	$1.2 \cdot 10^{-6}$	$3.1 \cdot 10^{-16}$	$5.5 \cdot 10^{-3}$

TABLE 1

Eigenvalue bounds by Gershgorin circles and the new method `verifyeigall`

As can be seen, the accuracy of eigenvalue inclusions of both methods decrease with dimension and size of cluster, and the new method is generally more accurate than inclusions by Gershgorin circles. The medians of the medians is better than the mean because the size of the cluster is small compared to the dimension. For $n = 1000$ and $k = 3$ some inclusions by Gershgorin circles are very wide.

Next we test the performance of `verifyeigall`, first for real and complex clusters of size k, respectively. Again, for $k = 1$ this is just a random matrix. For one test matrix we compute the mean and the median of the relative error of all eigenvalue and of all eigenvector inclusions. Then, for different dimensions and 100 test cases each, the median of those numbers is displayed in columns 3 to 6 in Tables 2 and 3. The number of test cases where inclusions could not be computed for all eigenpairs is listed in column 'fail'.

The mean of the ratio of computing times between `verifyeigall` and Matlab's `eig` is displayed in the last column. That compares apples and oranges because `verifyeigall` computes verified inclusions of all results, whereas Matlab's `eig` calculates approximations without error bound. For random matrices the error bounds computed by `verifyeigall` are pretty accurate; for a cluster of size 3 the mean relative error of the approximations by Matlab's `eig` against a multiple precision calculation is about 10^{-8} and the maximum about 10^{-6}. In any case, the comparison gives an impression on the necessary effort for verified inclusions.

The accuracy of the inclusions corresponds to the sensitivity of the cluster $\mathbf{u}^{-1/k}$. That is true for the eigenvalues and for the invariant subspaces. The ratio of computing time increases with the cluster size because likely a recursive call of `verifyeigall`

n	k	relerr L		relerr X		fail	$t_{\text{new}}/t_{\text{eig}}$
100	1	$3.2 \cdot 10^{-16}$	$3.2 \cdot 10^{-16}$	$1.2 \cdot 10^{-15}$	$1.1 \cdot 10^{-15}$	-	6.8
200	1	$3.2 \cdot 10^{-16}$	$3.2 \cdot 10^{-16}$	$1.7 \cdot 10^{-15}$	$1.9 \cdot 10^{-15}$	-	5.0
500	1	$3.3 \cdot 10^{-16}$	$3.3 \cdot 10^{-16}$	$2.6 \cdot 10^{-15}$	$2.6 \cdot 10^{-15}$	-	9.2
1000	1	$3.1 \cdot 10^{-16}$	$3.1 \cdot 10^{-16}$	$3.4 \cdot 10^{-15}$	$3.8 \cdot 10^{-15}$	-	10.9
100	2	$6.0 \cdot 10^{-9}$	$4.0 \cdot 10^{-14}$	$1.8 \cdot 10^{-11}$	$3.0 \cdot 10^{-12}$	-	12.1
200	2	$3.1 \cdot 10^{-9}$	$4.5 \cdot 10^{-14}$	$4.7 \cdot 10^{-11}$	$6.9 \cdot 10^{-12}$	-	8.3
500	2	$1.0 \cdot 10^{-9}$	$4.7 \cdot 10^{-14}$	$1.4 \cdot 10^{-10}$	$1.9 \cdot 10^{-11}$	-	9.2
1000	2	$7.5 \cdot 10^{-10}$	$5.1 \cdot 10^{-14}$	$3.4 \cdot 10^{-10}$	$3.9 \cdot 10^{-11}$	-	19.6

TABLE 2
Random matrix with real cluster of size k

n	k	relerr L		relerr X		fail	$t_{\text{new}}/t_{\text{eig}}$
100	1	$3.1 \cdot 10^{-16}$	$3.1 \cdot 10^{-16}$	$1.0 \cdot 10^{-15}$	$1.1 \cdot 10^{-15}$	-	3.7
200	1	$3.3 \cdot 10^{-16}$	$3.3 \cdot 10^{-16}$	$1.4 \cdot 10^{-15}$	$1.4 \cdot 10^{-15}$	-	3.2
500	1	$3.2 \cdot 10^{-16}$	$3.1 \cdot 10^{-16}$	$2.1 \cdot 10^{-15}$	$2.1 \cdot 10^{-15}$	-	4.5
1000	1	$3.3 \cdot 10^{-16}$	$3.3 \cdot 10^{-16}$	$3.0 \cdot 10^{-15}$	$3.7 \cdot 10^{-15}$	-	5.0
100	2	$7.4 \cdot 10^{-9}$	$4.3 \cdot 10^{-14}$	$1.7 \cdot 10^{-11}$	$3.0 \cdot 10^{-12}$	-	17.1
200	2	$5.1 \cdot 10^{-9}$	$4.5 \cdot 10^{-14}$	$4.4 \cdot 10^{-11}$	$6.7 \cdot 10^{-12}$	-	11.7
500	2	$3.6 \cdot 10^{-9}$	$4.9 \cdot 10^{-14}$	$1.4 \cdot 10^{-10}$	$1.9 \cdot 10^{-11}$	-	18.9
1000	2	$1.9 \cdot 10^{-9}$	$5.2 \cdot 10^{-14}$	$3.3 \cdot 10^{-10}$	$4.0 \cdot 10^{-11}$	-	20.6

TABLE 3
Random matrix with complex cluster of size k

is necessary. The ratio is a little better for complex clusters, seemingly because `eig` slows down. There is no failure, i.e., verified inclusions have been computed for all eigenpairs in all test cases.

n	k	relerr L		relerr X		fail	$t_{\text{new}}/t_{\text{eig}}$
100	3	$1.4 \cdot 10^{-6}$	$4.1 \cdot 10^{-14}$	$1.8 \cdot 10^{-11}$	$3.0 \cdot 10^{-12}$	-	19.7
200	3	$7.2 \cdot 10^{-7}$	$4.4 \cdot 10^{-14}$	$4.7 \cdot 10^{-11}$	$6.7 \cdot 10^{-12}$	-	12.6
500	3	$3.8 \cdot 10^{-7}$	$4.7 \cdot 10^{-14}$	$1.5 \cdot 10^{-10}$	$1.9 \cdot 10^{-11}$	-	22.1
1000	3	$2.9 \cdot 10^{-7}$	$5.2 \cdot 10^{-14}$	$3.7 \cdot 10^{-10}$	$4.1 \cdot 10^{-11}$	2	26.0
100	5	$1.9 \cdot 10^{-4}$	$4.2 \cdot 10^{-14}$	$5.8 \cdot 10^{-11}$	$3.2 \cdot 10^{-12}$	-	23.4
200	5	$1.1 \cdot 10^{-4}$	$4.8 \cdot 10^{-14}$	$9.0 \cdot 10^{-11}$	$6.9 \cdot 10^{-12}$	4	15.5
500	5	$3.3 \cdot 10^{-5}$	$4.9 \cdot 10^{-14}$	$9.1 \cdot 10^{-10}$	$1.9 \cdot 10^{-11}$	13	26.8
1000	5	$2.1 \cdot 10^{-5}$	$5.4 \cdot 10^{-14}$	$2.3 \cdot 10^{-9}$	$4.3 \cdot 10^{-11}$	34	33.3
100	10	$9.0 \cdot 10^{-4}$	$4.3 \cdot 10^{-14}$	$2.9 \cdot 10^{-7}$	$3.2 \cdot 10^{-12}$	66	45.0
200	10	$6.1 \cdot 10^{-5}$	$6.2 \cdot 10^{-14}$	$7.8 \cdot 10^{-6}$	$9.1 \cdot 10^{-12}$	72	28.8
500	10	$9.5 \cdot 10^{-7}$	$8.2 \cdot 10^{-14}$	$7.0 \cdot 10^{-6}$	$3.4 \cdot 10^{-11}$	78	42.3
1000	10	$3.2 \cdot 10^{-7}$	$1.0 \cdot 10^{-13}$	$9.3 \cdot 10^{-6}$	$8.1 \cdot 10^{-11}$	65	41.9

TABLE 4
Random matrix with real cluster of size k

Results for clusters of size up to 10 are displayed in Table 4. Now we observe failures. That means, that not for all eigenpairs inclusions could be computed, usually

for those close to the cluster. Again the relative accuracy corresponds to the sensitivity of the clusters.

For separated eigenvalues usually one repeat-loop in step 6) of `verifyeigall` suffices. For clustered eigenvalues it may be executed a number of times in order to identify and separate the clusters. Therefore, for larger clusters we observe a significant increase of computing time relative to Matlab's `eig`. That is mainly due to Matlab's interpretation overhead and in particular the use of the operator concept. That can be improved significantly by using function calls and/or calculating left and right bounds individually using directed rounding as Florian Bünger did for the Taylor model and AWA toolbox in INTLAB [6]. We refrained from doing this for the sake of better readability of the code.

n	k	relerr L		relerr X		fail	$t_{\text{new}}/t_{\text{eig}}$
100	2	$9.4 \cdot 10^{-9}$	$4.1 \cdot 10^{-14}$	$2.4 \cdot 10^{-11}$	$3.0 \cdot 10^{-12}$	-	18.9
200	2	$6.1 \cdot 10^{-9}$	$4.5 \cdot 10^{-14}$	$7.1 \cdot 10^{-11}$	$7.0 \cdot 10^{-12}$	-	12.2
500	2	$2.1 \cdot 10^{-9}$	$4.8 \cdot 10^{-14}$	$1.8 \cdot 10^{-10}$	$1.9 \cdot 10^{-11}$	-	20.2
1000	2	$2.0 \cdot 10^{-9}$	$5.2 \cdot 10^{-14}$	$4.0 \cdot 10^{-10}$	$4.1 \cdot 10^{-11}$	-	22.1
100	5	$5.2 \cdot 10^{-4}$	$4.9 \cdot 10^{-14}$	$5.0 \cdot 10^{-7}$	$3.3 \cdot 10^{-12}$	5	29.1
200	5	$3.4 \cdot 10^{-4}$	$5.1 \cdot 10^{-14}$	$5.7 \cdot 10^{-7}$	$7.7 \cdot 10^{-12}$	12	18.3
500	5	$7.1 \cdot 10^{-5}$	$5.2 \cdot 10^{-14}$	$2.5 \cdot 10^{-7}$	$2.1 \cdot 10^{-11}$	34	30.5
1000	5	$5.6 \cdot 10^{-5}$	$8.9 \cdot 10^{-14}$	$3.9 \cdot 10^{-6}$	$7.1 \cdot 10^{-11}$	65	37.7

TABLE 5

Random matrix with two real clusters of size k to different eigenvalues

In Table 5 results for two clusters of size k to different eigenvalues are reported. Again, for clusters of size 5 sometimes inclusions could not be computed for all eigenpairs, otherwise the results correspond to the previous ones.

n	k	relerr L		relerr X		fail	$t_{\text{new}}/t_{\text{eig}}$
100	1	$5.2 \cdot 10^{-16}$	$3.1 \cdot 10^{-16}$	$1.5 \cdot 10^{-16}$	$1.4 \cdot 10^{-16}$	-	5.2
200	1	$1.1 \cdot 10^{-15}$	$3.1 \cdot 10^{-16}$	$2.8 \cdot 10^{-16}$	$2.6 \cdot 10^{-16}$	-	5.1
500	1	$1.5 \cdot 10^{-13}$	$4.2 \cdot 10^{-14}$	$1.2 \cdot 10^{-10}$	$1.7 \cdot 10^{-11}$	-	7.4
1000	1	$1.9 \cdot 10^{-13}$	$4.8 \cdot 10^{-14}$	$3.1 \cdot 10^{-10}$	$3.8 \cdot 10^{-11}$	-	7.6
100	2	$1.4 \cdot 10^{-8}$	$4.3 \cdot 10^{-14}$	$1.8 \cdot 10^{-11}$	$3.0 \cdot 10^{-12}$	2	19.2
200	2	$9.6 \cdot 10^{-9}$	$4.5 \cdot 10^{-14}$	$4.7 \cdot 10^{-11}$	$6.8 \cdot 10^{-12}$	4	13.5
500	2	$4.0 \cdot 10^{-9}$	$4.8 \cdot 10^{-14}$	$1.5 \cdot 10^{-10}$	$1.9 \cdot 10^{-11}$	1	19.8
1000	2	$3.5 \cdot 10^{-9}$	$5.2 \cdot 10^{-14}$	$3.9 \cdot 10^{-10}$	$4.1 \cdot 10^{-11}$	-	24.8

TABLE 6

Random matrix with two real clusters of size k to the same eigenvalue

Finally we generate matrices with two clusters of size k to the same eigenvalue. The sensitivity is about $\mathbf{u}^{-1/k}$ for k denoting the largest Jordan block. So again the accuracy corresponds to the sensitivity, for cluster size 2 not always inclusions for all eigenpairs could be computed, and the computing time increases due to a recursive call of `verifyeigall`. The results are shown in Table 6.

3.2. Numerical results for truly multiple eigenvalues. Next we perform similar tests but knowing the true Jordan structure of the test matrices. The construction of the test matrices is as follows. First a diagonal matrix is generated with

small integer entries divided by a small power of 2, and with a $k \times k$ block with equal diagonal entries and superdiagonal set to 1. Call that matrix J. Next sparse lower and upper unit triangular matrices L and U with integer entries are generated, so that their inverses have integer entries. Finally it is tested that $A = U^{-1}L^{-1}JLU$ is computed without rounding errors.

The results displayed in Table 7 are structured as those in the previous section. Here k is the size of the Jordan block, where 1/1 refers to a double eigenvalue of (algebraic and) geometric multiplicity 2.

n	k	relerrL		relerrX		fail	$t_{\text{new}}/t_{\text{eig}}$
100	1/1	$3.2 \cdot 10^{-16}$	$3.0 \cdot 10^{-16}$	$2.7 \cdot 10^{-9}$	$4.4 \cdot 10^{-10}$	-	15.2
200	1/1	$1.3 \cdot 10^{-15}$	$3.2 \cdot 10^{-16}$	$5.9 \cdot 10^{-9}$	$1.1 \cdot 10^{-9}$	-	13.9
500	1/1	$5.0 \cdot 10^{-15}$	$3.6 \cdot 10^{-16}$	$1.2 \cdot 10^{-8}$	$4.2 \cdot 10^{-9}$	1	20.2
1000	1/1	$1.9 \cdot 10^{-14}$	$4.1 \cdot 10^{-16}$	$3.6 \cdot 10^{-8}$	$1.4 \cdot 10^{-8}$	3	20.7
100	2	$1.2 \cdot 10^{-11}$	$1.1 \cdot 10^{-12}$	$9.2 \cdot 10^{-2}$	$7.7 \cdot 10^{-3}$	-	33.2
200	2	$2.9 \cdot 10^{-11}$	$2.2 \cdot 10^{-12}$	$9.3 \cdot 10^{-2}$	$3.8 \cdot 10^{-3}$	-	31.0
500	2	$5.5 \cdot 10^{-11}$	$5.5 \cdot 10^{-12}$	$9.8 \cdot 10^{-2}$	$8.4 \cdot 10^{-9}$	-	49.3
1000	2	$8.4 \cdot 10^{-11}$	$9.1 \cdot 10^{-12}$	$1.0 \cdot 10^{-1}$	$2.8 \cdot 10^{-9}$	2	48.5
100	3	$1.7 \cdot 10^{-11}$	$1.6 \cdot 10^{-12}$	$9.6 \cdot 10^{-2}$	$1.8 \cdot 10^{-3}$	-	41.0
200	3	$2.2 \cdot 10^{-11}$	$2.7 \cdot 10^{-12}$	$9.6 \cdot 10^{-2}$	$8.6 \cdot 10^{-4}$	1	33.4
500	3	$5.0 \cdot 10^{-11}$	$5.4 \cdot 10^{-12}$	$9.9 \cdot 10^{-2}$	$3.3 \cdot 10^{-4}$	-	50.1
1000	3	$8.2 \cdot 10^{-11}$	$9.4 \cdot 10^{-12}$	$1.0 \cdot 10^{-1}$	$2.7 \cdot 10^{-9}$	4	49.0
100	5	$2.4 \cdot 10^{-11}$	$1.9 \cdot 10^{-12}$	$9.3 \cdot 10^{-2}$	$1.8 \cdot 10^{-3}$	11	42.2
200	5	$2.0 \cdot 10^{-11}$	$2.6 \cdot 10^{-12}$	$9.7 \cdot 10^{-2}$	$3.2 \cdot 10^{-3}$	22	26.9
500	5	$4.2 \cdot 10^{-11}$	$5.1 \cdot 10^{-12}$	$1.0 \cdot 10^{-1}$	$2.7 \cdot 10^{-3}$	29	51.1
1000	5	$8.2 \cdot 10^{-11}$	$7.5 \cdot 10^{-12}$	$1.0 \cdot 10^{-1}$	$6.4 \cdot 10^{-4}$	31	49.9

TABLE 7
Random matrix with true Jordan blocks of size k

For larger block size, the eigenvalue inclusions are more accurate, where the eigenvector inclusions are less accurate than expected. As before the number of cases where not all eigenpairs are included increases with the the block size and dimension, and the computing time as well. For larger block size and larger dimension the relative accuracy of some eigenvector and/or invariant subspace inclusions is poor, sometimes only one digit can be verified.

3.3. Comparison to Miyajima's methods. Finally we compare our algorithm `verifyeigall` with the, to our knowledge, only competitor published in [14]. Miyajima presents two algorithms `VAE_NSD` and `VAE_NJD`. The source code of both algorithms was kindly provided by the author. He also uses error-free transformations to improve the accuracy of the inclusions, and he uses NAClab [29], a publically available Matlab toolbox which, in particular, offers algorithms to compute the Jordan canonical form of a matrix. That is an ill-posed problem, thus the true Jordan structure may not be determined correctly in floating-point arithmetic.

We mention that there are also block versions of the algorithms in [14], however, the results shown in [14] are similar to the unblocked version so we refrain from comparisons to save space.

We encountered hard Matlab errors or infinite loops when testing `VAE_NSD` and

VAE_NJD, and were advised by Miyajima to use the older 2013 version of NAClab rather than the newer 2018 version. That improved the situation, but still infinite loops occur and sometimes the routines stopped with a Matlab error. Moreover, the routines VAE_NSD and VAE_NJD are time consuming. We therefore had to reduce the

dimension n	50	100	200	500	1000
# test cases	100	100	50	15	10

TABLE 8
Number of test cases for Tables 9, 10 and 11

number of test cases according to Table 8, and for larger sizes of clusters the dimension had to be reduced.

We generate several test matrices, both with eigenvalue clusters and true Jordan blocks as described in Subsections 3.1 and 3.2, respectively. For one test matrix we compute the mean of the relative errors of all eigenvalue and of all eigenvector inclusions, and for different dimensions the median of those numbers is displayed in the blocks "relerr L" for the eigenvalues and in the blocks "relerr X" for the eigenvectors or invariant subspaces. As before, the case $k = 1$ is added in which case the test matrix is just a random matrix.

The number of test cases where no inclusion could be computed for one eigenpair is listed in the column "failure". More precisely, if, as before, only for one eigenpair no inclusion could be computed it is considered as failure. If an algorithm fails for all test cases, the relative errors for L and X are noted as NaN.

For some test cases three numbers are displayed for VAE_NSD and VAE_NJD in the block "failure". In that case the first number is the total number of failures as just described, the second the number of cases where the algorithm runs into an infinite loop, and the third where Matlab stops with an error. For example, for $n = 200$ and $k = 5$, VAE_NSD failed in 17 of 50 cases, no infinite loop was encountered and 1 Matlab error. Similarly, again for $n = 200$ and $k = 5$, VAE_NJD failed in 12 out of 50 test cases, in 11 cases the algorithm ran into an infinite loop, and 1 test case ended with a Matlab error.

Finally, in the last two columns, the mean time ratio of VAE_NSD and VAE_NJD to our new routine is displayed. For example, for $n = 500$ and $k = 3$, VAE_NSD required on the average 10.2 times the computing time of verifyeigall, and VAE_NJD was on the average 29.3 times slower.

The results are shown in Table 9, where the numbers above the horizontal line refer to clusters of eigenvalues as in Subsection 3.1, and below to true Jordan blocks as in Subsection 3.2. For increasing dimension and size of clusters or Jordan blocks, the number of failures of all algorithms increase, more moderately for verifyeigall. Algorithm VAE_NSD failed for cluster size 10 completely, where VAE_NJD failed for all true Jordan blocks with no inclusion, infinite loop and/or Matlab error. If successful, the quality of the eigenvalue inclusions of VAE_NJD and verifyeigall are comparable, those of VAE_NSD are a little weaker. The quality of the eigenvector inclusions of both VAE_NSD and VAE_NJD are weaker than those of verifyeigall in our test cases. Algorithm VAE_NSD is generally slower than verifyeigall, whereas VAE_NJD is much slower in our test cases.

In the previous tables we computed the mean of the relative errors of all inclusion components and took the median over all test cases. We think that gives a general impression of the performance of the methods. Since the size of the clusters or Jordan blocks is relatively small compared to the dimension, those numbers favor the majority

		relerr L			relerr X			failure			time/t_new	
n	k	VAE_NSD	VAE_NJD	new	VAE_NSD	VAE_NJD	new	VAE_NSD	VAE_NJD	new	VAE_NSD	VAE_NJD
50	1	$1.7 \cdot 10^{-13}$	$2.2 \cdot 10^{-15}$	$3.2 \cdot 10^{-16}$	$7.5 \cdot 10^{-14}$	$7.6 \cdot 10^{-14}$	$1.0 \cdot 10^{-15}$	-	-	-	5.4	11.4
100	1	$2.9 \cdot 10^{-13}$	$2.7 \cdot 10^{-15}$	$3.2 \cdot 10^{-16}$	$1.7 \cdot 10^{-13}$	$1.8 \cdot 10^{-13}$	$1.1 \cdot 10^{-15}$	-	-	-	7.9	21.9
200	1	$5.4 \cdot 10^{-13}$	$3.6 \cdot 10^{-15}$	$3.2 \cdot 10^{-16}$	$4.3 \cdot 10^{-13}$	$4.3 \cdot 10^{-13}$	$1.9 \cdot 10^{-15}$	-	-	-	6.8	22.5
500	1	$1.6 \cdot 10^{-12}$	$4.7 \cdot 10^{-15}$	$3.3 \cdot 10^{-16}$	$1.3 \cdot 10^{-12}$	$1.3 \cdot 10^{-12}$	$2.6 \cdot 10^{-15}$	-	-	-	19.3	59.8
1000	1	$3.6 \cdot 10^{-12}$	$5.5 \cdot 10^{-15}$	$3.1 \cdot 10^{-16}$	$3.3 \cdot 10^{-12}$	$3.3 \cdot 10^{-12}$	$3.8 \cdot 10^{-15}$	-	4/-/4	-	29.5	90.3
50	2	$1.0 \cdot 10^{-6}$	$2.2 \cdot 10^{-8}$	$3.5 \cdot 10^{-8}$	$6.3 \cdot 10^{-8}$	$8.4 \cdot 10^{-13}$	$2.3 \cdot 10^{-16}$	-	3/2/1	-	3.0	5.3
100	2	$4.2 \cdot 10^{-6}$	$1.7 \cdot 10^{-8}$	$2.8 \cdot 10^{-8}$	$1.3 \cdot 10^{-7}$	$9.5 \cdot 10^{-12}$	$1.5 \cdot 10^{-16}$	-	-	-	5.0	12.1
200	2	$7.5 \cdot 10^{-7}$	$3.3 \cdot 10^{-9}$	$5.2 \cdot 10^{-9}$	$1.6 \cdot 10^{-7}$	$1.9 \cdot 10^{-11}$	$2.0 \cdot 10^{-16}$	-	-	-	6.3	17.8
500	2	$2.2 \cdot 10^{-6}$	$4.6 \cdot 10^{-9}$	$9.4 \cdot 10^{-9}$	$3.7 \cdot 10^{-7}$	$1.6 \cdot 10^{-10}$	$2.6 \cdot 10^{-16}$	-	-	-	14.6	36.8
1000	2	$4.5 \cdot 10^{-5}$	$1.9 \cdot 10^{-9}$	$6.2 \cdot 10^{-8}$	$2.4 \cdot 10^{-7}$	$3.6 \cdot 10^{-10}$	$2.7 \cdot 10^{-16}$	1	1/-/1	-	29.2	55.8
50	3	$2.6 \cdot 10^{-4}$	$1.0 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$	$1.8 \cdot 10^{-5}$	$1.2 \cdot 10^{-12}$	$2.7 \cdot 10^{-16}$	-	2/2/-	-	2.0	3.7
100	3	$7.5 \cdot 10^{-5}$	$1.3 \cdot 10^{-6}$	$2.5 \cdot 10^{-6}$	$2.4 \cdot 10^{-5}$	$6.6 \cdot 10^{-12}$	$2.7 \cdot 10^{-16}$	1	2/2/-	-	3.4	21.1
200	3	$2.6 \cdot 10^{-4}$	$1.2 \cdot 10^{-6}$	$3.6 \cdot 10^{-6}$	$1.8 \cdot 10^{-5}$	$3.3 \cdot 10^{-11}$	$2.7 \cdot 10^{-16}$	2	1/1/-	-	3.3	27.9
500	3	$1.2 \cdot 10^{-4}$	$4.9 \cdot 10^{-7}$	$7.4 \cdot 10^{-7}$	$1.8 \cdot 10^{-5}$	$8.3 \cdot 10^{-11}$	$2.7 \cdot 10^{-16}$	9	1/1/-	-	10.2	29.3
50	5	$5.8 \cdot 10^{-3}$	$5.9 \cdot 10^{-4}$	$1.2 \cdot 10^{-3}$	$9.0 \cdot 10^{-4}$	$1.5 \cdot 10^{-12}$	$3.4 \cdot 10^{-16}$	28	11/11/-	-	1.7	19.3
100	5	$6.3 \cdot 10^{-3}$	$2.7 \cdot 10^{-4}$	$6.3 \cdot 10^{-4}$	$6.9 \cdot 10^{-4}$	$9.2 \cdot 10^{-12}$	$3.8 \cdot 10^{-16}$	37	10/8/2	-	2.6	43.7
200	5	$4.3 \cdot 10^{-3}$	$6.1 \cdot 10^{-5}$	$2.5 \cdot 10^{-4}$	$6.5 \cdot 10^{-4}$	$4.5 \cdot 10^{-11}$	$4.6 \cdot 10^{-16}$	17/-/1	12/11/1	2	3.3	174.6
500	5	NaN	$3.5 \cdot 10^{-6}$	$1.9 \cdot 10^{-4}$	NaN	$1.6 \cdot 10^{-11}$	$4.2 \cdot 10^{-16}$	15/-/4	14/14/-	1	8.0	356.5
50	10	NaN	$8.1 \cdot 10^{-3}$	$2.0 \cdot 10^{-2}$	NaN	$5.4 \cdot 10^{-12}$	$2.9 \cdot 10^{-15}$	100/-/1	74/74/-	30	1.4	84.0
100	10	NaN	$5.3 \cdot 10^{-3}$	$9.3 \cdot 10^{-3}$	NaN	$1.4 \cdot 10^{-11}$	$5.0 \cdot 10^{-15}$	50/-/2	46/0/46	36	1.7	130.9
200	10	NaN	$1.0 \cdot 10^{-3}$	$6.8 \cdot 10^{-4}$	NaN	$3.8 \cdot 10^{-11}$	$1.2 \cdot 10^{-15}$	20/-/1	18/18/-	13	2.3	255.5
500	10	NaN	NaN	$2.3 \cdot 10^{-4}$	NaN	NaN	$4.5 \cdot 10^{-15}$	15	15/15/-	9	7.0	351.6
50	1/1	$1.2 \cdot 10^{-7}$	NaN	$5.1 \cdot 10^{-16}$	$4.2 \cdot 10^{-11}$	NaN	$1.4 \cdot 10^{-10}$	-	100/13/87	-	7.2	99.5
100	1/1	$2.5 \cdot 10^{-6}$	NaN	$1.4 \cdot 10^{-13}$	$1.4 \cdot 10^{-10}$	NaN	$4.3 \cdot 10^{-10}$	-	50/24/26	-	8.8	454.2
50	2	$1.5 \cdot 10^{-2}$	NaN	$2.6 \cdot 10^{-8}$	$7.2 \cdot 10^{-11}$	NaN	$1.9 \cdot 10^{-10}$	7	100/23/77	-	2.9	55.3
100	2	$1.6 \cdot 10^{-4}$	NaN	$1.9 \cdot 10^{-6}$	$4.3 \cdot 10^{-10}$	NaN	$4.8 \cdot 10^{-10}$	2	50/34/16	-	4.1	266.3
50	3	$3.3 \cdot 10^{-3}$	NaN	$1.6 \cdot 10^{-4}$	$2.8 \cdot 10^{-10}$	NaN	$3.1 \cdot 10^{-10}$	7	100/20/80	-	2.1	34.6
100	3	$2.8 \cdot 10^{-3}$	NaN	$5.6 \cdot 10^{-5}$	$3.8 \cdot 10^{-10}$	NaN	$6.5 \cdot 10^{-10}$	14	50/35/15	-	3.2	229.9
50	5	$4.6 \cdot 10^{-2}$	NaN	$3.9 \cdot 10^{-3}$	$3.4 \cdot 10^{-8}$	NaN	$5.4 \cdot 10^{-10}$	66	100/48/52	2	1.8	78.1
100	5	NaN	NaN	$1.7 \cdot 10^{-3}$	NaN	NaN	$1.2 \cdot 10^{-9}$	50	50/40/10	4	2.7	240.2

TABLE 9

Comparison with VAE_NSD *and* VAE_NSD *for clusters and true Jordan blocks with number of test cases according to Table 8*

		relerr L			relerr X		
n	k	VAE_NSD	VAE_NJD	new	VAE_NSD	VAE_NJD	new
50	1	$5.0 \cdot 10^{-11}$	$1.9 \cdot 10^{-13}$	$4.4 \cdot 10^{-16}$	$1.2 \cdot 10^{-9}$	$8.4 \cdot 10^{-10}$	$2.0 \cdot 10^{-15}$
100	1	$7.9 \cdot 10^{-11}$	$2.4 \cdot 10^{-13}$	$4.4 \cdot 10^{-16}$	$2.7 \cdot 10^{-8}$	$1.8 \cdot 10^{-8}$	$2.2 \cdot 10^{-15}$
200	1	$3.8 \cdot 10^{-11}$	$7.7 \cdot 10^{-14}$	$4.4 \cdot 10^{-16}$	$3.1 \cdot 10^{-8}$	$2.3 \cdot 10^{-8}$	$2.7 \cdot 10^{-15}$
500	1	$6.1 \cdot 10^{-10}$	$9.0 \cdot 10^{-13}$	$4.4 \cdot 10^{-16}$	$1.6 \cdot 10^{-7}$	$2.8 \cdot 10^{-7}$	$5.2 \cdot 10^{-15}$
1000	1	$6.9 \cdot 10^{-10}$	$3.5 \cdot 10^{-13}$	$4.4 \cdot 10^{-16}$	$8.3 \cdot 10^{-6}$	$8.4 \cdot 10^{-6}$	$7.7 \cdot 10^{-15}$
50	2	$2.3 \cdot 10^{-4}$	$5.1 \cdot 10^{-6}$	$1.2 \cdot 10^{-5}$	$2.7 \cdot 10^{-1}$	$1.7 \cdot 10^{-4}$	$3.0 \cdot 10^{-6}$
100	2	$3.6 \cdot 10^{-3}$	$9.8 \cdot 10^{-5}$	$2.0 \cdot 10^{-4}$	$4.2 \cdot 10^{-1}$	$3.1 \cdot 10^{-4}$	$3.3 \cdot 10^{-7}$
200	2	$1.3 \cdot 10^{-3}$	$5.1 \cdot 10^{-6}$	$5.8 \cdot 10^{-6}$	$9.5 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$	$4.4 \cdot 10^{-7}$
500	2	$9.7 \cdot 10^{-4}$	$1.2 \cdot 10^{-6}$	$3.0 \cdot 10^{-6}$	$6.7 \cdot 10^{-1}$	$1.1 \cdot 10^{-2}$	$3.4 \cdot 10^{-6}$
1000	2	$6.6 \cdot 10^{-3}$	$8.3 \cdot 10^{-6}$	$2.6 \cdot 10^{-5}$	$9.3 \cdot 10^{-1}$	$4.8 \cdot 10^{-2}$	$1.9 \cdot 10^{-7}$
50	3	$4.2 \cdot 10^{-2}$	$1.4 \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$	$1.0 \cdot 10^{0}$	$7.9 \cdot 10^{-2}$	$6.8 \cdot 10^{-9}$
100	3	$3.0 \cdot 10^{-2}$	$3.1 \cdot 10^{-4}$	$6.7 \cdot 10^{-4}$	$9.5 \cdot 10^{-1}$	$2.4 \cdot 10^{-2}$	$3.6 \cdot 10^{-9}$
200	3	$2.5 \cdot 10^{-2}$	$1.4 \cdot 10^{-4}$	$2.8 \cdot 10^{-4}$	$9.8 \cdot 10^{-1}$	$7.4 \cdot 10^{-3}$	$2.0 \cdot 10^{-8}$
500	3	$3.5 \cdot 10^{-1}$	$5.7 \cdot 10^{-4}$	$2.2 \cdot 10^{-3}$	$1.0 \cdot 10^{0}$	$9.5 \cdot 10^{-1}$	$1.1 \cdot 10^{-7}$
1000	3	$2.9 \cdot 10^{-1}$	$2.7 \cdot 10^{-5}$	$1.6 \cdot 10^{-3}$	$1.0 \cdot 10^{0}$	$6.7 \cdot 10^{-1}$	$1.1 \cdot 10^{-7}$
50	5	$7.3 \cdot 10^{-1}$	$9.0 \cdot 10^{-2}$	$2.2 \cdot 10^{-1}$	$1.2 \cdot 10^{0}$	$7.8 \cdot 10^{-1}$	$1.3 \cdot 10^{-1}$
100	5	$7.2 \cdot 10^{-1}$	$5.1 \cdot 10^{-2}$	$9.8 \cdot 10^{-2}$	$1.0 \cdot 10^{0}$	$7.2 \cdot 10^{-1}$	$3.6 \cdot 10^{-1}$
200	5	$8.0 \cdot 10^{-1}$	$6.9 \cdot 10^{-3}$	$5.0 \cdot 10^{-2}$	$6.3 \cdot 10^{0}$	$8.6 \cdot 10^{-1}$	$9.2 \cdot 10^{-1}$
500	5	$2.1 \cdot 10^{0}$	$3.5 \cdot 10^{-4}$	$1.2 \cdot 10^{-1}$	$2.1 \cdot 10^{0}$	$3.8 \cdot 10^{-6}$	$9.9 \cdot 10^{-1}$

TABLE 10
Extract of Table 9 taking the maximum relative error over all samples

of eigenpairs which belong to simple eigenvalues. However, if in 51 out of 100 test cases narrow intervals are computed but not in the remaining, that may remain undetected by taking the median.

For completeness we display therefore part of the results of Table 9, again but now taking the maximum relative error of all inclusions of one test case and then the maximum of that number over all test cases. That means, the displayed relative errors in Table 10 are the maximum over all inclusions and over all test cases.

Again we can see a correspondence between the sensitivity of the clusters to the accuracy of the inclusions. For the eigenvalue inclusions, `verifyeigall` is a little better than both `VAE_NSD` and `VAE_NJD`, the eigenvector inclusions are significantly better than both `VAE_NSD` and `VAE_NJD` for clusters up to size 3. For size 5 the inclusions of all methods are poor in the worst case.

For the same test cases as in Table 10 we calculate the time ratios

$$t_{\texttt{VAE\_NSD}}/t_{\texttt{verifyeigall}} \qquad \text{and} \qquad t_{\texttt{VAE\_NJD}}/t_{\texttt{verifyeigall}}$$

and display the minimum, mean, median and maximum over all test cases in Table 11. As before, `verifyeigall` is on the average significantly faster than `VAE_NSD` and `VAE_NJD`, in the best case the ratio is close to 1.

Next we show numerical evidence that an approximated eigenmatrix approximation is hardly ill-conditioned. The first candidate is an integer matrix with 3 Jordan blocks to eigenvalues $1, 2$ and 3, respectively, each of size 10, and no other eigenvalues. An integer similarity transformation is applied to produce a matrix without zero entries retaining the anticipated Jordan structure. For this matrix there are only 3 linear independent eigenvectors, the remaining 27 are corresponding principal vectors.

		$t_{\texttt{VAE\_NSD}}/t_{\texttt{verifyeigall}}$				$t_{\texttt{VAE\_NSD}}/t_{\texttt{verifyeigall}}$			
n	k	min	mean	median	max	min	mean	median	max
50	1	4.4	5.3	5.4	6.1	9.2	11.4	11.6	12.6
100	1	7.1	7.9	7.8	8.5	19.9	21.8	21.7	23.0
200	1	6.0	6.8	6.8	7.4	19.5	22.4	22.5	24.3
500	1	15.2	19.5	19.4	23.5	48.6	60.4	60.7	69.5
1000	1	27.2	29.4	29.5	31.1	83.8	90.2	91.2	94.1
50	2	1.2	3.7	3.8	7.8	2.2	6.3	6.3	12.3
100	2	3.6	5.4	5.5	7.8	8.7	13.2	12.9	18.7
200	2	3.5	7.7	8.4	13.2	10.7	22.5	24.4	36.3
500	2	8.0	18.0	13.4	31.5	19.8	47.6	35.2	88.5
1000	2	14.8	36.3	40.8	57.6	36.9	94.3	103.2	147.9
50	3	1.5	2.6	2.7	3.7	3.3	4.7	4.7	16.5
100	3	2.3	3.6	3.6	4.7	6.0	9.6	9.5	21.9
200	3	3.2	4.2	4.3	4.9	10.6	57.9	14.0	458.8
500	3	7.2	9.9	9.6	12.5	25.0	105.4	28.5	680.3
1000	3	12.6	14.7	13.5	18.1	34.8	40.6	37.1	50.0
50	5	1.1	1.7	1.7	3.4	2.1	16.7	3.4	231.1
100	5	1.4	2.8	2.7	4.7	4.5	40.0	7.3	265.7
200	5	2.1	3.3	3.2	5.0	8.9	176.8	128.7	517.1
500	5	5.0	8.4	8.4	11.1	28.9	380.6	392.9	572.6

TABLE 11
Extract of Table 9: detailed timing

The eigenvalue approximations computed by Matlab's `eig` are displayed in Figure 1.

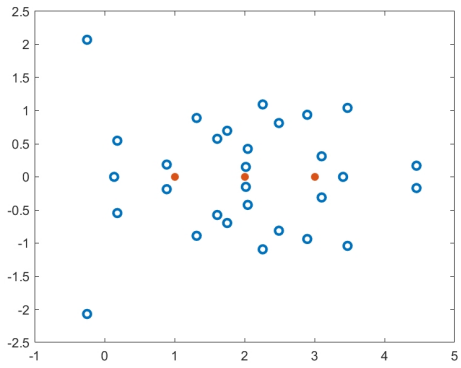

FIG. 1. *Approximate eigenvalues 'o' of the matrix "Jordan" with* 3 *Jordan blocks with* 10*-fold eigenvalues* $1, 2$ *and* 3 *depicted in red '*'.*

Recall that there should be only the 3 dots at $1, 2$ and 3 depicted in red. Given these approximations verified inclusions can hardly be computed - and they are not.

Since the Matlab function `eig` can only approximate eigenvectors, not principal vectors, we may expect that `[W,D] = eig(A)` produces a matrix W of numerical rank 3. However, as displayed in the first line of Table 12, $\mathrm{cond}(W) \approx 3.5 \cdot 10^9$. That is because of the smoothening effect of floating-point arithmetic, some kind of regularization. We remark that in that specific case of an integer matrix a computer

algebra system such as the symbolic toolbox in Matlab computed easily the correct eigenvalues together with the Jordan structure.

In Table 12 the maximal errors of the inclusions of the eigenvalues and of the eigenvectors are displayed, each for Algorithms `VAE_NSD` and `VAE_NJD` in 3.2, and for the new algorithm, followed by the time in seconds. An entry "-" means that not for all eigenpairs inclusions could be computed.

Next we try some test matrices from the Matlab gallery which may cause problems. In a number of cases no inclusion could be computed, in some cases the inclusions are very wide. For example, the eigenvalue inclusions for the matrix `gallery/lesp` of `VAE_NSD` and the new algorithm are of reasonable quality, where `VAE_NJD` fails. However, the eigenvector inclusions are poor with a relative error close to 1.

For the "frank 0" and "frank 1" matrices we observe ill-conditioned W. However, these are integer Hessenberg matrices for which, because of the zeros below the first subdiagonal, the smoothing of floating-point arithmetic is not as effective.

The timing is self-explaining; in three cases, the "Jordan" matrix with 3 Jordan blocks of size 10, "frank 0" and "frank 1" matrices of dimension 30 Algorithm `VAE_NJD` was caught in an infinite loop, for the "lesp" matrix of dimension 30 Algorithm `VAE_NJD` caused a Matlab error.

Acknowledgment. The author wishes to thank Florian Bünger for his thorough reading and valuable comments. I am indebted to the anonymous referees for their thorough reading and many valuable comments.

REFERENCES

[1] R. Alam and S. Bora, *On sensitivity of eigenvalues and eigendecompositions of matrices*, Linear Algebra and its Applications (LAA), 396 (1983), pp. 273–301.
[2] G. Alefeld and H. Spreuer, *Iterative improvement of componentwise errorbounds for invariant subspaces belonging to a double or nearly double eigenvalue*, Computing, 36 (1986), pp. 321–334.
[3] A. Bavely and G. Stewart, *An algorithm for computing reducing subspaces by block diagonalization*, Siam Journal Numerical Analysis, 16 (1979), pp. 359–367.
[4] H. Behnke, *The determination of guaranteed bounds to eigenvalues with the use of variational methods II*, in Computer Arithmetic and Self-Validating Numerical Methods, C. Ullrich, ed., Academic Press, 1990, pp. 155–170.
[5] H. Behnke, *The calculation of guaranteed bounds to eigenvalues using complementary variational principles*, Computing, 47, No.1 (1991), pp. 11–27.
[6] F. Bünger, *A Taylor model toolbox for solving ODEs implemented in MATLAB/INTLAB*, Journal of Computational and Applied Mathematics, 368 (2020), p. 112511.
[7] R. Byers and D. Kressner, *Structured condition numbers for invariant subspaces*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 326–347.
[8] L. Collatz, *Einschließungssatz für die charakteristischen Zahlen von Matrizen*, Math. Z., 48 (1942), pp. 221–226.
[9] A. Deif, *Rigorous perturbation bounds for eigenvalues and eigenvectors of a matrix*, Journal of Computational and Applied Mathematics, 57 (1995), pp. 403–412.
[10] W. Kahan, *A more complete interval arithmetic*, Lecture notes for a summer course at the University of Michigan, (1968).
[11] M. Karow, D. Kressner, and F. Tisseur, *Structured eigenvalue condition numbers*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 1052–1068.
[12] R. Krawczyk, *Fehlerabschätzung reeller Eigenwerte und Eigenvektoren von Matrizen*, Computing, 4 (1969), pp. 281–293.
[13] MATLAB User's Guide, *The MathWorks Inc.*, 1987.
[14] S. Miyajima, *Fast enclosure for all eigenvalues and invariant subspaces in generalized eigenvalue problems*, SIAM J. Matrix Analysis (SIMAX), 35 (2014), pp. 1205–1225.
[15] S. Miyajima, T. Ogita, S. Rump, and S. Oishi, *Fast verification for all eigenpairs in symmetric positive definite generalized eigenvalue problems*, Reliable Computing, 14 (2010),

matrix	n	cond(W)	relerr L			relerr X			time [sec]		
			VAE_NSD	VAE_NJD	new	VAE_NSD	VAE_NJD	new	VAE_NSD	VAE_NJD	new
Jordan	30	$3.5 \cdot 10^{9}$	$2.5 \cdot 10^{4}$	-	-	0	-	-	0.04	∞	0.03
frank 0	10	$7.2 \cdot 10^{5}$	$6.1 \cdot 10^{-8}$	$4.7 \cdot 10^{-10}$	$3.3 \cdot 10^{-16}$	$1.3 \cdot 10^{-4}$	$6.6 \cdot 10^{-5}$	$1.9 \cdot 10^{-15}$	0.02	0.03	0.006
frank 0	20	$2.5 \cdot 10^{14}$	1.7	-	0.18	0.88	-	-	0.03	13	0.07
frank 0	30	$4.1 \cdot 10^{14}$	-	-	-	-	-	-	0.03	∞	0.09
frank 1	10	$8.4 \cdot 10^{5}$	$8.5 \cdot 10^{-8}$	$1.5 \cdot 10^{-9}$	$3.3 \cdot 10^{-16}$	$4.8 \cdot 10^{-5}$	$3.7 \cdot 10^{-5}$	$5.2 \cdot 10^{-14}$	0.02	0.04	0.007
frank 1	20	$2.8 \cdot 10^{13}$	0.99	-	-	1.8	-	-	0.03	21	0.07
frank 1	30	$1.3 \cdot 10^{14}$	-	-	-	-	-	-	0.03	∞	0.09
lesp	10	$4.4 \cdot 10^{2}$	$6.8 \cdot 10^{-14}$	$2.0 \cdot 10^{-15}$	$3.1 \cdot 10^{-16}$	$1.2 \cdot 10^{-11}$	$8.0 \cdot 10^{-5}$	$3.9 \cdot 10^{-12}$	0.02	0.03	0.007
lesp	20	$4.5 \cdot 10^{5}$	$8.3 \cdot 10^{-12}$	$2.5 \cdot 10^{-15}$	$3.2 \cdot 10^{-16}$	0.93	0.58	0.57	0.03	0.29	0.008
lesp	30	$4.4 \cdot 10^{8}$	$8.6 \cdot 10^{-10}$	-	$3.2 \cdot 10^{-16}$	0.51	-	0.88	0.04	-	0.007
wilk	21	1	$3.6 \cdot 10^{-14}$	$4.4 \cdot 10^{-10}$	$3.6 \cdot 10^{-15}$	$2.3 \cdot 10^{-2}$	$1.4 \cdot 10^{-5}$	$8.2 \cdot 10^{-9}$	0.03	0.09	0.01

TABLE 12
Selected matrices from the Matlab gallery

pp. 24–45.
[16] R. Moore, *A test for existence of solutions for non-linear systems*, SIAM J. Numer. Anal. (SINUM), 4 (1977), pp. 611–615.
[17] A. Neumaier, *Interval methods for systems of equations*, Encyclopedia of Mathematics and its Applications, Cambridge University Press, 1990.
[18] S. Oishi, K. Ichihara, M. Kashiwagi, K. Kimura, X. Liu, H. Masai, Y. Morikura, T. Ogita, K. Ozaki, S. Rump, K. Sekine, A. Takayasu, and N. Yamanaka, *Principle of verified numerical computations*, Corona publisher, Tokyo, Japan, 2018 [in Japanese].
[19] S. Rump, *Kleine Fehlerschranken bei Matrixproblemen*, PhD thesis, Universität Karlsruhe, 1980.
[20] S. Rump, *INTLAB - INTerval LABoratory*, in Developments in Reliable Computing, T. Csendes, ed., Kluwer Academic Publishers, Dordrecht, 1999, pp. 77–104. http://www.ti3.tuhh.de/intlab.
[21] S. Rump, *Computational error bounds for multiple or nearly multiple eigenvalues*, Linear Algebra and its Applications (LAA), 324 (2001), pp. 209–226.
[22] S. Rump, *Verification methods: Rigorous results using floating-point arithmetic*, Acta Numerica, 19 (2010), pp. 287–449.
[23] J. Sun, *Multiple eigenvalue sensitivity analysis*, Linear Algebra and its Applications (LAA), 137/138 (1990), pp. 183–211.
[24] J. Sun, *On condition numbers of a nondefective multiple eigenvalue*, Numer. Math., 61 (1992), pp. 265–275.
[25] Y. Watanabe, K. Nagatou, M. Plum, and M. Nakao, *Some eigenvalue excluding methods for infinite dimensional operators*. manuscript, 2010.
[26] Y. Watanabe, N. Yamamoto, and M. Nakao, *Verification methods of generalized eigenvalue problems and its applications (in Japanese)*, Trans JSIAM, 9 (2009), pp. 137–150.
[27] J. Wilkinson, *The algebraic eigenvalue problem*, Clarendon Press, Oxford, 1965.
[28] T. Yamamoto, *Error bounds for computed eigenvalues and eigenvectors*, Numer. Math., 34 (1980), pp. 189–199.
[29] Z. Zeng, T. Li, L. Chen, T. Chen, W. Hao, T. Lee, W. Wu, and A. Sommese, *NAClab - Numerical Algebraic Computing toolbox for Matlab*, 2012 and 2018. http://homepages.neiu.edu/~zzeng/naclab.html#download.