Rendiconti di Matematica, Serie VII
Volume 27, Roma (2007), 155-182

Zero-structured matrix modifications keeping essentially unaltered sets of eigenvalues

SILVIA NOSCHESE – LIONELLO PASQUINI

ABSTRACT: *We show how to construct zero-structured matrices that modify a given matrix A keeping essentially unaltered a selected set of eigenvalues. An accurate report on a significant collection of numerical tests is included.*
Applications to the study of the behavior of the inherent error are discussed.

1 – Introduction

In [14] we considered a matrix $A \in \mathbb{C}^{n\times n}$, and we showed how to determine unit-norm matrices E, belonging to an assigned subspace $\mathcal{S}$ (i.e. having an assigned zero-structure), in such a way that $A + \alpha\, E$ keeps essentially unaltered a selected set of simple eigenvalues of A for reasonably large values of α. We first transformed the problem into that of finding nontrivial solutions to a suitable linear homogeneous system which depends on the ν selected eigenvalues and on the subspace $\mathcal{S}$. Then we showed that nontrivial solutions to such a system exist – no matter how the number ν of the eigenvalues ($1 \le \nu \le n$), the eigenvalues themselves and the subspace $\mathcal{S}$ have been singled out – if and only if

$$m = \dim(\mathcal{S})$$

is large enough. A sufficient condition is $m > \nu$. The necessary and sufficient condition, which may turn out to be sensibly weaker, can be found below in

KEY WORDS AND PHRASES: *Eigenvalues – Structured matrix modifications – Conditioning.*
A.M.S. CLASSIFICATION: 65F15 – 65F30 – 65F50 (primary) – 65F35 (secondary)

Theorem 2.1, which is here reported from [14] in the version that best fits the contents of the present article.

Also, two draft algorithms having complementary qualities were defined.

In this paper we present substantially improved versions of the two algorithms sketched in [14, Section 3]. Their MATLAB codes are available upon request. Also, we discuss how to apply them to investigate the behavior of the error arising in the computation of a selected simple eigenvalue λ when the inherent error $\lambda(A+\alpha\, E)-\lambda(A)$ has been made negligible, and to analyze phenomena which can be observed in case of ill-conditioning of λ.

The main motivation to our work is that [10, Section 1]:

> There is a growing interest in structured perturbation analysis due to the substantial development of algorithms for structured problems.

Further reasons of interest are the following.

The first one consists in the characterization of the matrices E belonging to an assigned subspace $\mathcal{S}$ and having, for ν eigenvalues of interest, the property which is the opposite of the one enjoyed by the ν Wilkinson perturbation matrices, that is to say by the ν matrices which lead to the individual condition numbers of those ν eigenvalues [25], [3] (see Section 2 for some details). To be more precise, the above mentioned property is exactly opposite to the one enjoyed by the ν matrices in $\mathcal{S}$ that lead to the individual *zero-structured* condition numbers of those ν eigenvalues [15].

The second reason consists in the opportunity of constructing in reality and of using such matrices E. This is of interest, for instance, every time that the problem under investigation enables one to modify m assigned entries in a matrix A (for instance, the entries that are not machine numbers) and one wants to change those entries finding out a matrix modification that does not perturb "too much" ν selected eigenvalues (for instance, the eigenvalues closest to the imaginary axis in a spectrum $\sigma(A)$ that lies in the left half of the complex plane).

The third reason is that the opportunity of constructing and actually testing the matrix solutions E allows for a deep understanding of how substantially a selected subset of entries influences selected eigenvalues. In other words, it allows knowing how large values of α are consistent with an assigned tolerance on the ν eigenvalues if a certain zero-structure is chosen. For instance, a complete understanding of such a problem led to results in [15], [13].

The outline of the paper is as follows.

In Section 2 we report from [14] on the underlying theory. Starting with Wilkinson's theory of the perturbation of a simple eigenvalue [25], [3], unit-norm matrices $E \in \mathcal{S}$ that annihilate all of the coefficients of the first term in the expansions of $\lambda_h(A+\alpha\, E)-\lambda_h(A)$, $h=1:\nu$, in powers of α are characterized, the λ_h's being simple eigenvalues of A.

In Sections 3 and 4 we present the improved algorithms. The algorithm presented in Section 3 (Algorithm 1) solves a suitable direct search optimization problem and uses the MATLAB functions **mdsmax** and **nmsmax** by N. J. Higham [6]. The algorithm presented in Section 4 (Algorithm 2) solves a set of equations formed by ν homogeneous linear equations in m unknowns and by a nonlinear normalization equation.

Section 5 is devoted to report and comment on tests as well as to discuss advantages and drawbacks of the two algorithms. The tests have been carried out using matrices A taken from the literature [12], [22], [3, Section 13], [26, Section 5], [4], [17], [23] or randomly generated. Remarks concerning the detection of the inherent errors in the observed ones can also be found in that section.

Conclusions are drawn in Section 6.

2 – Report on theoretical results

We start with briefly reporting from the classical Wilkinson perturbation theory [25], [3]. Then results from [14] and remarks are collected in Sections 2.2 and 2.3.

2.1 – Wilkinson's perturbation theory of a simple eigenvalue

Let λ be a simple eigenvalue of a matrix $A \in \mathbb{C}^{n\times n}$ and let x and y be respectively the right and left eigenvector associated to λ with $\|x\|_2 = \|y\|_2 = 1$. Following [25], [3], we can write, for $\varepsilon > 0$ small enough,

$$(A + \varepsilon\, E)\, x(\varepsilon) = \lambda(\varepsilon)\, x(\varepsilon), \qquad \|E\|_2 = 1\,,$$

where x and λ are differentiable functions such that $\lambda(0) = \lambda$ and $x(0) = x$. It can be proved that one has

$$\left|\frac{d\lambda}{d\varepsilon}\right|_{\varepsilon=0} = \left|\frac{y^H E\, x}{y^H x}\right| \le \frac{\|y\|_2\, \|E\|_2\, \|x\|_2}{|y^H x|} = \frac{1}{|y^H x|}\,, \tag{2.1}$$

and that the upper-bound is attained if $E = y\, x^H$.

Remark 2.1. It is worth noticing that in the above outlined theory the Frobenius norm can replace everywhere the 2−norm.

In the sequel, the matrix $y\, x^H$ will be referred to as the Wilkinson perturbation matrix and denoted by W_λ.

The inner product $y^H x$ is usually denoted by $s(\lambda)$,

$$s(\lambda) := y^H x\,,$$

while $1\,/|s(\lambda)|$ is denoted by $\kappa(\lambda)$ and called the condition number of λ,

$$\kappa(\lambda) := \frac{1}{|y^H x|} = \frac{1}{|s(\lambda)|}\,.$$

2.2 – Our idea

In this paper, instead of considering the matrix E that yields the worst result in (2.1), we look for a unit-norm matrix E that annihilates the numerator in the ratio $\left|y^H E\,x\,/y^H x\,\right|$. More in general we look for a unit-norm matrix E that annihilates the numerators in the ratios

$$\frac{y^{(h)H} E\,x^{(h)}}{y^{(h)H} x^{(h)}}\,, \quad h = 1:\nu\,, \tag{2.2}$$

relevant to a set $\widetilde{\sigma}(A) \subseteq \sigma(A)$ of ν, $1 \le \nu \le n$, selected simple eigenvalues λ_h, $h = 1:\nu$. Here $x^{(h)}$ and $y^{(h)}$, with $\left\|x^{(h)}\right\|_2 = \left\|y^{(h)}\right\|_2 = 1$, respectively are the right and left eigenvector associated to λ_h.

Theorem 2.1 contains a result which is essential to define the two algorithms we deal with in Sections 3 and 4.

To introduce the theorem we first consider the simpler case $\nu = 1, \mathcal{S} = \mathbb{C}^{n\times n}$. In this case, to get $\left|y^H E\,x\,/y^H x\,\right| = 0$, we have to solve one linear equation in the n^2 unknowns e_{ij} plus the nonlinear equation $\|E\|_2 = 1$. To clarify the matter that follows, it is worth formally re-writing that linear equation in the form

$$\sum_{k=1}^{n^2} c_k\,\xi_k = 0\,, \tag{2.3}$$

where the ξ_k are the unknowns e_{ij} arranged by columns in a n^2-length vector ξ,

$$k = (j-1)n + i; \quad j = \frac{k+n-i}{n}\,, \quad i = k - n(j-1)\,, \tag{2.4}$$
$$i, j = 1:n, \quad k = 1:n^2\,,$$

$$\xi_k = e_{ij}\,, \tag{2.5}$$

$$c_k = c_k(\lambda) = \frac{\overline{y}_i\,x_j}{y^H x}\,. \tag{2.6}$$

Slightly more complicated is the case $\mathcal{S} \subset \mathbb{C}^{n\times n}$, $m = \dim(\mathcal{S}) < n^2$, which is the one we are mainly interested in. In this case, again, we arrange the m entries of $E = [e_{ij}]$ by columns into a vector ξ, but we take account of the zeros in the structure. So $\xi = [\xi_1\,\xi_2\,\dots\,\xi_m]^T$ is a vector of length m and equations (2.4) become

$$k = (j-1)n + i - \zeta(i,j); \quad j = \frac{k+n-i+\zeta(i,j)}{n}\,, \tag{2.7}$$
$$i = k - n(j-1) + \zeta(i,j)\,, \quad i, j = 1:n,\; k = 1:m\,,$$

where $\zeta(i,j)$ denotes the number of the zero entries in E that precede an e_{ij} that has been enclosed in the set of the selected m entries. Equations (2.5) and (2.6) remain unchanged whereas equation (2.3) becomes

$$\sum_{k=1}^{m} c_k\,\xi_k = 0\,. \tag{2.8}$$

Finally in the general case we have $1 \le \nu \le n$ and $1 \le m \le n^2$. Only the linear equation (2.8), (2.6) changes and becomes the linear homogeneous system of ν equations in m unknowns

$$\sum_{k=1}^{m} c_{hk}\,\xi_k = 0, \quad h = 1 : \nu\,, \tag{2.9}$$

$$c_{hk} = c_{hk}(\lambda_h) = \frac{\overline{y}_i^{(h)}\, x_j^{(h)}}{y^{(h)H}\, x^{(h)}}\,. \tag{2.10}$$

Of course, we are looking for nontrivial solutions to (2.9), (2.10) to construct and then normalize the corresponding nonzero matrix according to (2.7). Standard arguments of Linear Algebra say that a necessary and sufficient condition for the existence of nontrivial solutions to (2.9), (2.10) is

$$m > r\,, \tag{2.11}$$

where r denotes the rank of the coefficient matrix $C = [c_{hk}]$ in (2.9):

$$r = \operatorname{rank}(C)\,.$$

Summarizing the above arguments leads to the following

Theorem 2.1. *Unit-norm matrices $E \in \mathcal{S}$ that annihilate all of the ratios in* (2.2) *exist if and only if condition* (2.11) *is satisfied. They are characterized in terms of ξ as the ones satisfying the linear homogeneous system of ν equations in m unknowns* (2.9), (2.10) *and the condition $\|E\|_2 = 1$.*

Remark 2.2. It is worth stressing that the numbers ν and m in their ranges, as well as the ν eigenvalues and the positions of the m entries, can be arbitrarily selected.

Remark 2.3. Condition (2.11) is not satisfied when $m = \dim(\mathcal{S})$ is equal to r. In this case, unit-norm matrices E that yield small enough values of the ratios in (2.2) might give satisfactory results to our problem (see e.g. Test 8 in Section 5.2.2).

2.3 – Using the Frobenius norm

Interesting remarks, mainly concerning the conditioning of a simple eigenvalue, can be added in case the Frobenius norm is used (see Remark 2.1). They are a consequence of Theorem 2.2 if, slightly modifying an idea in [18], we define in the matrix space the inner product

$$(A, B) := \operatorname{trace}(B^H A),$$

which leads to the Frobenius norm.

THEOREM 2.2. *Let λ be a simple eigenvalue of A. Let x and y respectivel be the right and left eigenvector associated to λ. Let $\|x\|_2 = \|y\|_2 = 1$. Let $E = [e_{ij}]$ belong to $\mathbb{C}^{n\times n}$. One has*

$$y^H E\, x = \sum_{i,j=1:n} \overline{y}_i\, e_{ij}\, x_j = \operatorname{trace}(W_\lambda^H E) = (E, W_\lambda)\,.$$

PROOF. Standard arguments lead to the first equality. The second one easily follows by observing that

$$\sum_{i,j=1:n} \overline{y}_i\, e_{ij}\, x_j = \sum_{i,j=1:n} (\overline{y}_i\, x_j)\, e_{ij} = \sum_{i,j=1:n} (x\, y^H)_{ji}\, e_{ij} = \sum_{j=1:n} (W_\lambda^H E)_{jj}\,.$$

This concludes the proof. □

The mentioned remarks are listed below.

REMARK 2.4. We can read Theorem 2.2 saying that the matrices E that annihilate $y^H E\, x$ are those orthogonal to the Wilkinson perturbation matrix W_λ and that the closer a matrix E is to be orthogonal to W_λ, the smaller $|y^H E\, x|$ is.

REMARK 2.5. If λ is an ill-conditioned eigenvalue, a unit-norm matrix E that makes $|y^H E\, x \,/ y^H x\,|$ very small can be simply obtained by taking $E = W_\lambda^H$. In fact, W_λ and W_λ^H are unit-norm matrices (both in the 2 and in the Frobenius norm) and

$$\left|\frac{y^H W_\lambda^H\, x}{y^H x}\right| = \left|\frac{y^H\, x\, y^H x}{y^H x}\right| = |s(\lambda)| = \frac{1}{\kappa(\lambda)}\,.$$

Remark 2.6. A straightforward computation shows that

$$(W_\lambda^H, W_\lambda) = s(\lambda)^2\,,$$

so that the more the eigenvalue λ is ill-conditioned, the closer W_λ and W_λ^H get to being orthogonal to each other.

Remark 2.7. The limit case of a defective eigenvalue associated with a unique Jordan block ($s(\lambda) = 0$) can be viewed as the case of orthogonality between W_λ and W_λ^H.

Remark 2.8. On the opposite, the case $W_\lambda^H = W_\lambda$ (i.e. W_λ Hermitian) can be viewed as the case of a perfectly conditioned eigenvalue ($|s(\lambda)| = 1$).

3 – Algorithm 1

The first algorithm we present is the more elaborate of the two.

The basic idea is to use the MATLAB functions **mdsmax** and **nmsmax** by Higham [6], [8]. The two codes are fully documented in their leading comment lines. The latter function implements the Nelder-Mead simplex method for direct search optimization [11]. The former implements the multidirectional search method in [21]. Doing so we transform the problem of solving the nonlinear system considered in Theorem 2.1 into a direct search optimization one. In fact, both those functions attempt to maximize a real function **fun** prepared by the user and referenced by a function handle. Our function **fun** computes

$$1 \Bigg/ \sum_{h=1}^{\nu} \left| \frac{y^{(h)H} \widetilde{E}\, x^{(h)}}{y^{(h)H} x^{(h)}} \right| \,, \tag{3.1}$$

$\widetilde{E}$ being any unit-norm matrix belonging to $\mathcal{S}$.

Besides functions **mdsmax** and **fun** [besides functions **nmsmax** and **fun**], an algorithm that comes up to standard should be provided with

- two scripts,
  - an opening one, acting as main program,
  - a concluding one, managing the output,
- some further functions, each one corresponding to a basic zero-structure.

A brief description of a proposal including seven such functions is given below. We begin showing the organization of the twelve files in Scheme 3.1.

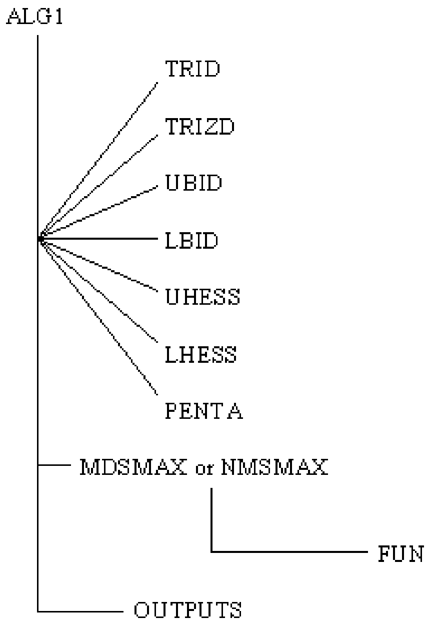


Fig. 1: Scheme 3.1.

3.1 – Script **alg1**

We divide this section into three parts.

3.1.1 – User's interaction

The user interacts with our files by means of **alg1**. Script **alg1** asks the user to:

1. enter the matrix A,
2. select the eigenvalues to be preserved,
3. specify the zero-structure of the matrix E,
4. specify the kind (complex/real) of the guess vector required by the functions **mdsmax** and **nmsmax** and mentioned in Section 3.2 below,
5. select one of the functions **mdsmax** and **nmsmax**.

To specify the ν eigenvalues, $1 \leq \nu \leq n$, the user is required to enter their relevant indices as they appear in the list displayed by **alg1**. The list is yielded by MATLAB's **eig** function. We disabled the default **balance** function – that implements a procedure in [16] – to avoid possible changes in the conditioning of the eigenvalues of the matrix A and of the modified matrix $A + F = A + \alpha\, E$.

To specify the zero-structure of the matrix E, the user is required to enter one of the following nine strings: 'trid', 'trizd', 'ubid', 'lbid', 'uhess', 'lhess', 'penta', 'full', 'sparse'.

a) The first seven strings ‘trid’, ‘trizd’, ‘ubid’, ‘lbid’, ‘uhess’, ‘lhess’ and ‘penta’ correspond to the names of the seven functions in the second column of Scheme 3.1 and stand for tridiagonal, tridiagonal with zero diagonal, upper bidiagonal, lower bidiagonal, upper Hessenberg, lower Hessenberg and pentadiagonal zero-structure, respectively.
b) String ‘full’ stands for the full zero-structure.
c) String ‘sparse’ stands for any sparse zero-structure. The user is required to enter the $2m-$length vector containing the m couples of indices that define the zero-structure of E.

In the cases in a) and b), the user is enabled to modify the previously selected basic zero-structure. In fact, it is possible to eliminate p entries among the m key ones, entering the relevant p couples of indices.

3.1.2 – Computation of the ν right and left eigenvectors

Besides the eigenvalues of the matrix A, MATLAB’s **eig** function produces a full matrix whose columns are the corresponding right eigenvectors. Also, applying such a function to the transposed and conjugated matrix A^H, one gets a full matrix whose columns are the left eigenvectors. We operated the right-left eigenvector matching as follows:

```
$[v, lam] = \mathbf{eig}(A, \text{‘nobalance’});$
$lam = \mathbf{diag}(lam);$
$[w, clam] = \mathbf{eig}(A', \text{‘nobalance’});$
$inds = \mathbf{input}(\text{‘Vector containing the indices of the selected eigenvalues:’});$
$X = \mathbf{cell}(\mathbf{zeros}(1, n)); Y = \mathbf{cell}(\mathbf{zeros}(1, n));$
for $h = inds$
    $[ignore, ind] = \mathbf{min}(\mathbf{abs}(lam(h) - \mathbf{conj}(diag(clam))));$
    $Y\{h\} = w(:, ind)/\mathbf{norm}(w(:, ind));$
    $X\{h\} = v(:, h)/\mathbf{norm}(v(:, h));$
end.
```

3.1.3 – Calls

If one of the seven strings ‘trid’, ‘trizd’, ‘ubid’, ‘lbid’, ‘uhess’, ‘lhess’, ‘penta’ has been entered (see a) in Section 3.1.1), script **alg1** calls the relevant function among the first seven in the second column of Scheme 3.1. Calls to **mdsmax** or to **nmsmax** (user’s choice) and to script **outputs** follow in any case.

The function selected among the first seven in the second column of Scheme 3.1 returns a properly defined matrix B. As we shall see below in Section 3.2.1, the B matrices play an important role in keeping the selected zero-structure of E.

3.2 – Interactions between Higham's functions and **fun**

Both **mdsmax** and **nmsmax** call **fun**. They interact with it in order to attempt to maximize the objective function (3.1) implemented in **fun**. The attempt is achieved iteratively. At each iteration, **fun** computes the objective function at the current data received by the calling routine. In detail, it derives a unit-norm matrix $\widetilde{E}$ from the data and returns the current value of the function in (3.1). Checks are carried out by **mdsmax** [by **nmsmax**] to verify whether the iteration has to be stopped or not. The data are conveyed by an m-length vector and by two parameters.

At first the m-length vector receives the guess vector required by **mdsmax** and **nmsmax** to start. Then it bears the current value of the vector ξ defined in Section 2. It is the only variable that changes during the iteration. The mentioned guess vector (see point 4 in Section 3.1.1) is defined using MATLAB's **rand** function. The real and the possible imaginary parts of the components range between -1 and 1. (The command **rand** ('seed',0) is inserted before to make the tests repeatable.)

The first parameter passed by **mdsmax** [by **nmsmax**] to **fun** is a matrix containing the right and left eigenvectors corresponding to the ν selected eigenvalues.

To the second parameter is dedicated the next section. This second parameter depends on the three cases distinguished above (see Section 3.1.1).

Both **mdsmax** and **nmsmax** give in output the vector which yielded the largest value of **fun** that had been found, the largest **fun** value found, and the total number of required evaluations.

3.2.1 – The second parameter passed to **fun**

The second parameter is called *struct* and deserves some explanation since, together with the above described m-length vector, it makes possible to keep the required zero-structure of E.

In the case of c), *struct* is an m-length vector. It is derived from the $2m$-length vector which is entered by the user and contains the indices that define the sparse zero-structure of E (see Section 3.1.1). The relationships among i, j and k shown in Section 2 are used.

In the remaining cases, *struct* is a matrix $B = B(\mathcal{S}) \in \mathbb{R}^{n^2 \times m}$ whose role in a general context is explained in [5], [20]. B is defined in such a way that

$$B\,\xi = \mathrm{vec}\,(E)\,,$$

where ξ is the current m-length vector, $E \in \mathcal{S}$ is the corresponding matrix, and vec is the operator that stacks the columns of a matrix into one long column vector.

In the cases in a), B is returned by the structuring function relevant to the selected zero-structure (i.e. relevant to the selected string). We provide seven structuring functions. They yield B matrices corresponding to:

the tridiagonal zero-structure (function **trid**),
the lower and upper bidiagonal zero-structure (functions **lbid** and **ubid**),
the tridiagonal with zero diagonal entries zero-structure (function **trizd**),
the lower and upper Hessenberg zero-structure (functions **lhess** and **uhess**),
the pentadiagonal zero-structure (function **penta**).

In the case of b), B is defined in **alg1** and it is the identity matrix I_{n^2}.

Finally, in both a) and b), p properly selected columns of B are dropped if the user asked to eliminate p entries from the chosen zero-structure (see Section 3.1.1). The resulting version of B is assigned to *struct*.

3.3 – Upper-bounds to the value of (3.1)

The iteration is terminated when a value of the objective function in (3.1) equals or exceeds an assigned upper-bound.

In the case of $\nu > 1$, we set such an upper-bound to 1/**eps**, **eps** being MATLAB's floating point relative accuracy.

In the case of $\nu = 1$, we modified the function in (3.1) to avoid unnecessary computations. We changed (3.1) in

$$\frac{1}{|y^H E\, x|}\,,$$

and then we divided the largest **fun** value found by $\kappa(\lambda)$. As a consequence, we set the upper-bound to $\kappa(\lambda)$/**eps**.

As we shall see below (Remark 3.2), different (lower) settings of the upper-bound may turn out to be useful.

3.4 – Reshaping and normalization

Function **mdsmax** [function **nmsmax**] does not return unit-norm matrices but the corresponding m-length vectors (see above for details about the output arguments of **mdsmax** and **nmsmax**). As a consequence, starting with the m-length vector and using *struct* too, both the reshaping procedure and the normalization are carried out in **fun** (at each iteration) and in **outputs** (at the end of the execution).

3.5 – Final remarks

We collect three concluding remarks. The first one deserves a particular attention.

Remark 3.1. Algorithm 1 turns out to be useful also in case no unit-norm matrix E satisfying the system in (2.9) exists. Even in such a case indeed, it can be of interest to know the maximum value reached by the function in (3.1) and, if it is large enough, the unit-norm matrix E yielding it. We have a surprising example on this subject (see Test 8 in Section 5).

REMARK 3.2. In case a lower accuracy is required, one can run **alg1** setting upper-bounds to the value of (3.1) adequately lower than the default ones specified in Section 3.3. This, of course, reduces the computational complexity of the whole procedure, which, normally, is quite expensive (there are $O((m+\nu)n^2)$ flops associated with each evaluation of **fun**).

REMARK 3.3. An improvement is recommended in [6] to best exploit functions **mdsmax** and **nmsmax**. It is advised in case unit-norm matrices E making the value in (3.1) to equal or to exceed the assigned upper-bound have not been found. It consists in repeating the whole procedure trying the function that has not been used before and taking as input the output obtained in the first application. Another trick consists in using the output obtained by the default regular simplex as starting guess for a further application that uses the optional right-angled simplex.

4 – Algorithm 2

This algorithm implements the most natural idea of finding a nontrivial solution of the $\nu \times m$ homogeneous linear system in (2.9) and then of normalizing the matrix that corresponds to that nontrivial solution in order to get one of the required matrices E. It works every time that a solution to our problem exists, that is every time that there are nontrivial solutions of the $\nu \times m$ homogeneous linear system in (2.9).

Since

$$r = \mathrm{rank}\,(C)\,,$$

$C = [c_{hk}]$ being the $\nu \times m$ coefficient matrix of the system, must be less than m (see Theorem 2.1), the above mentioned idea leads to solve a $r \times m$ underdetermined linear system. Thus, we decided to use a straightforward modification of Algorithm 5.7.1 in [2]. The modification is simply due to the fact that our system is homogeneous.

In detail, we start with applying to C the rank-revealing QR factorization with column pivoting (see e.g. [2, Section 5.4.1]). We get

$$C\,P = Q\,R\,,$$

P being the $m \times m$ permutation matrix generated during the factorization. A standard argument shows that

1. all the nontrivial solutions ξ of $C\,\xi = 0$ can be obtained by finding any nontrivial solution η of
$$R\,\eta = 0$$
and then computing
$$\xi = P\,\eta$$
(there is a one-to-one correspondence, defined by the perturbation P, between the ξ's and the η's);
2. a nontrivial solution η to the homogeneous linear system $R\,\eta = 0$ can be obtained by
   (a) leaving out the last $\nu - r$ equations in the system,
   (b) assigning arbitrary, not all vanishing, values to the last $m-r$ unknowns,
   (c) solving the reduced, nonsingular $r \times r$ *non* homogeneous triangular linear system obtained this way.

The complete $m-$length vector solution ξ is spread by columns into a matrix of the required zero-structure. Then this matrix is normalized yielding the matrix E. (See Section 4.5 below.)

4.1 – Computational cost

The amount of required flops can be estimated by adding to that related to Algorithm 5.7.1 and stated in [2, Section 5.4.1] an estimate of the number of flops needed to compute the column vector of the known terms of the $r \times r$ nonhomogeneous system in point 2 (c) above. An estimate of this number is $2\,(r \times m)$ and the addition yields the total amount of $2\,r^2 - r^3/3 + 2\,(r \times m) = 2\,r^2(m-1) + 2\,r\,m - r^3/3$ flops.

4.2 – Sensitivity and stability

As far as we know, there is no procedure more reliable than the one we adopted to find a solution to system (2.9). Indeed, many assertions, statements and declarations in literature are in favor of the line we chose.

Concerning the conditioning of the final $r \times r$ nonhomogeneous linear system in point 2 (c), the reader is referred to [7], [9, Chapter 8]. Here, promising sentences are quoted from [24, p. 105] and [19, p. 150] to emphasize the high accuracy that is frequently observed in practice in the solutions to triangular linear systems and, what is more, it is shown that the use of column pivoting in QR factorization applied to the solution of a linear system can greatly improve the conditioning of the resulting triangular system.

Regarding to the stability of the procedure, the reference is to [2, Section 5.7], [1, node42.html] and again to [7], [9, Chapter 8].

4.3 – Script **alg2**

Our code is a MATLAB script named **alg2**.

4.3.1 – User's interaction

Script **alg2** asks the user to:

1. enter the matrix A (as in script **alg1**),
2. choose the eigenvalues to be preserved (as in script **alg1**),
3. select the zero-structure of the matrix E (as in script **alg1**),
4. specify the kind (complex/real) of the values to be assigned to the $m - r$ unknowns mentioned at the beginning of this Section 4 in point 2 (b).

In addition, an information on the conditioning of the $r \times r$ coefficient matrix of the nonhomogeneous linear system in point 2 (c) is given upon request.

4.3.2 – Constructing and solving the reduced nonsingular $r \times r$ nonhomogeneous linear system mentioned in point 2 (c)

First the $\nu \times m$ matrix C in the underdetermined homogeneous linear system is computed taking into account the definition of the c_{hk} 's in Section 2.

Secondly, the upper trapezoidal $\nu \times m$ coefficient matrix R in the transformed underdetermined homogeneous linear system $R\eta = 0$ is computed using the QR factorization with column pivoting: $[Q, R, P] = \mathbf{qr}(C)$.

Subsequently, the rank-revealing property of the QR factorization with column pivoting is exploited to determine the rank r using a command strictly analogous to the one in MATLAB's **rank** function: $r = \mathbf{sum}(\mathbf{abs}(\mathbf{diag}(R)) > m * \mathbf{norm}(C) * \mathbf{eps})$.

Finally, in the case of $r < \nu$, the last $\nu - r$ rows of R are dropped, getting a full rank upper trapezoidal $r \times m$ matrix: $R(r + 1 : \nu, :) = [\,]$.

In order to transform the linear homogeneous system obtained this way into the $r \times r$ nonhomogeneous system mentioned in point 2 (c), **alg2** determines a $(m - r)$ – length random column vector η_2 and constructs the column vector b of the known terms: $b = -R(:, r + 1 : m) * \eta_2$. As usual, the command **rand**('seed',0) is inserted before to make the tests repeatable.

The resulting nonhomogeneous triangular linear system is solved by using MATLAB's $\backslash$ function. It is worth noticing that, in this case, such a function carries out the back substitution: $\eta_1 = R(:, 1 : r)\backslash b$.

An information on the conditioning of the coefficient matrix $R(:, 1 : r)$ is given upon request. MATLAB's **cond** function is used.

4.3.3 – Constructing a solution matrix E

The vector $\eta = [\eta_1; \eta_2]$ solves $R\eta = 0$ and the vector $\xi = P\eta$ solves $C\xi = 0$. The matrix E that corresponds to ξ is a solution to our problem and it is constructed in the following way.

- E is initialized with $1's$ in the m key-entries, and $0's$ elsewhere.
- A vector aux is defined. It contains the indices in $\mathrm{vec}\,(E)$ of the m entries defining the zero-structure of E.
- The components of ξ are put in the appropriate places in E: $E(aux) = \xi$.
- Finally E is normalized.

5 – Numerical tests

In this section we report on the tests we carried out. Both the algorithms outlined in Sections 3 and 4 were checked under MATLAB 6.5 (R13). The command **rand**('seed',0) (see Sections 3.2 and 4.3.2) allows the user to repeat tests.

A few tables summarize the results obtained in tests we singled out since they exemplify significant issues. Each test is described by declaring the algorithm that was used and specifying the interactions that occurred. To do this, we strictly refer to the numbered items in User's interaction sections (see Sections 3.1.1 and 4.3.1). Also, in each case, we report the condition numbers of the considered eigenvalues, evaluated using a function of ours which we defined on purpose to disable the **balance** default in **eig**. (The right-left eigenvector matching is operated as described in Section 3.1.2.) The information on the eigenvalue conditioning will be useful below when we draw some conclusions concerning relations between inherent and observed errors. Finally, we also report the value in (3.1) [the moduli of the ratios in (2.2)] obtained using Algorithm 1 [Algorithm 2].

Matrices $F = \alpha\, E$, with α large enough to make the inherent error become the dominant part of the error, have been used. We have taken $\alpha = 10^{\mu}$**eps**, **eps** being MATLAB's floating point relative accuracy and μ a nonnegative integer. Then, using a properly defined script along with spectral plots, we observed the relevant induced perturbations $|\lambda_h^*(A) - \lambda_h^*(A + \alpha\, E)|$. (Here and in the sequel the superscript * marks the computed version of an eigenvalue.) The values assigned to the integer μ are reported in the first row of each table, while the relevant induced perturbations are drawn up in columns, under the values of μ, often roughly reporting only the negative exponent in their normalized exponential representation. The first column shows the indices of the ν eigenvalues that we selected as they appear in the list displayed by **eig**. Often tables report in the last row, marked by "oths", global information related to the other (non selected) eigenvalues.

Algorithm 2 has been always extremely fast. Algorithm 1 has required a number of iterations ranging from 20 to 348.

5.1 – Randomly generated matrices

Here we collect tests chosen among the ones we carried out working on matrices generated by MATLAB's **rand** function. Before defining the matrix A, we inserted the command **rand**('seed',0).

5.1.1 – Real matrices

The two tests reported in this section both deal with matrices of quite large dimension ($n = 100$). Thus, we rather used Algorithm 2 since it is cheaper than the other one. The tables show a behavior of the observed errors $\lambda_h^*(A) - \lambda_h^*(A + \alpha\, E)$ related to the selected eigenvalues. Such behavior is typical of the inherent errors when the first term in the expansion in powers of α is extremely small, if not even zero.

– **Test 1.**

Algorithm 2. A=**toeplitz**([r 2*(**rand**(1,3)-.5) **zeros**(1,96)], [r 2*(**rand** (1,3)-.5) **zeros**(1,96)]), having previously defined r=2*(**rand**-.5). The eigenvalue closest to zero (77th). 'penta'. $p = 0$. Real.

$\kappa(\lambda^*(77))$=1.3651e+2. The modulus of the ratio in (2.2) is 7.e-18**.**

Table 1

μ/λ^*	$0 \nearrow 9$	10	11	12	13	14
77	-15,-14	5.e-12	5.e-10	5.e-8	5.e-6	5.e-4
oths	-16→-9$\nearrow$-10→-8	-9→-7	-8→-6	-7→-5	-6→-4	-5→-3

– **Test 2**

Algorithm 2. $A = L$+**diag**(2*(**rand**(100,1)-.5))+$L.'$, having previously defined L=**tril**(2*(**rand**(100)-.5),-1). The closest to zero (52nd) and the largest eigenvalue (100th). 'trid'. $p = 0$. Real.

$\kappa(\lambda^*(52))$=$\kappa(\lambda^*(100))$=1 (A is a normal matrix). The 2-norm condition number of the 2×2 coefficient matrix of the nonhomogeneous linear system in Section 4, point 2 (c), is 1.47725e+0. The moduli of the ratios in (2.2) respectively are 1.e-17 and 5.e-17.

Table 2

μ/λ^*	$0 \nearrow 9$	10	11	12	13	14	15
52	-16,-15	4.e-14	4.e-12	4.e-10	4.e-8	4.e-6	4.e-4
100	-15,-14	2.e-13	6.e-12	6.e-10	6.e-8	6.e-6	6.e-4
oths	-16→-14$\nearrow$-10→-8	-9→ -7	-8→-6	-7→-5	-6→-4	-6→-3	-4→-2

5.1.2 – Complex matrices

Even this section reports two tests. Since the size of A is not too large ($n = 30$), the first one has been carried out using both the algorithms, to make a comparison between the two solutions E. Only Algorithm 2 has been used in the second test ($n = 100$). Again, the results we obtained agree with those theoretically predictable.

– **Test 3**

Algorithm 2. A=2*(**rand**(30)-.5+i*(**rand**(30)-.5)). The largest eigenvalue (1st). 'full'. $p = 0$. Complex.

$\kappa(\lambda^*(1))$=2.2272e+0. The modulus of the ratio in (2.2) is 3.e-17.

Table 3

μ/λ^*	0↗9	10	11	12	13	14	15
1	-15,-14	4.7e-13	4.8e-11	4.8e-9	4.8e-7	4.8e-5	4.7e-3
oths	-15,-14↗-8,-7	-7,-6	-6,-5	-5,-4	-4,-3	-3,-2	-2,-1

Algorithm 1. As before. 'mds'.

The value of the ratio in (3.1), reached in 51 iterations (116101 function evaluations), is 1.e-16.

Table 3a

μ/λ^*	0↗9	10	11	12	13	14	15
1	-15,-14	1.9e-13	1.9e-11	1.9e-9	1.9e-7	1.9e-5	2.0e-3
oths	-15,-14↗-9→-7	-8→-6	-7→-5	-6→-4	-5→-3	-4→ -2	-3→-1

Comparison. In this case the two algorithms found two matrices E yielding essentially equivalent results.

– **Test 4**

Algorithm 2. A=2*(**rand**(100)-.5+i*(**rand**(100)-.5)). The closest to zero (99th) and the largest eigenvalue (1st). 'trizd'. $p = 0$. Complex.

$\kappa(\lambda^*(1))$=2.1923e+0; $\kappa(\lambda^*(99))$=4.3074e+0. The 2-norm condition number of the 2×2 coefficient matrix of the nonhomogeneous linear system in Section 4, point 2 (c), is 2.01301e+0. The moduli of the ratios in (2.2) respectively are 4.e-17 and 2.e-17.

Table 4

μ/λ^*	$0\nearrow 10$	11	12	13	14	15	16
1	-14	6.4e-13	5.9e-11	5.9e-9	5.9e-7	5.8e-5	5.3e-3
99	–16,-15,-14	8.0e-12	8.0e-10	8.0e-8	8.0e-6	8.1e-4	9.6e-2
oths	$-15,-14\nearrow-8,-7$	-7,-6	-6,-5	-5,-4	-4,-3	-3,-2	–2,-1

5.2 – Matrices taken from the literature

Here we discuss tests on matrices taken from the literature. The Bessel matrix and one of the famous Wilkinson matrices have been entered with the commands shown below. The commands to enter the other matrices we treat in this section have been taken from MATLAB's toolbox "gallery – Higham test matrices".

5.2.1 – A brief miscellany

In this section we treat four matrices and deliberately deal with well-conditioned eigenvalues.

– **Test 5. Lesp matrix**

This is a tridiagonal matrix with real, sensitive eigenvalues [12], [22]. The sensitivities of the eigenvalues increase exponentially as the eigenvalues grow more negative. The dimension of the matrix considered in this test is 15.

ALGORITHM 2. A=**gallery**('lesp',15). The first eight eigenvalues in the list displayed by **eig**. 'lbid'. $p = n$. The diagonal entries. Real.

$\kappa(\lambda^*(1))$=1.3221e+0; $\kappa(\lambda^*(2))$=3.0356e+0; $\kappa(\lambda^*(3))$=8.2739e+0; $\kappa(\lambda^*(4))$ =2.2689e+1; $\kappa(\lambda^*(5))$=6.2254e+1; $\kappa(\lambda^*(6))$=1.6684e+2; $\kappa(\lambda^*(7))$=4.1721e+2; $\kappa(\lambda^*(8))$=9.2481e+2. The 2-norm condition number of the 8×8 coefficient matrix of the nonhomogeneous linear system in Section 4, point 2 (c), is 8.94236e+0. The moduli of the ratios in (2.2) respectively are 2.e-17, 3.e-17, 3.e-17, 5.e-17, 3.e-17, 1.e-18, 4.e-17 and 9.e-18.

Table 5

μ/λ^*	$0\nearrow 9$	10	11	12	13	14	15
1	-15,-14	-15	-14	-16	-15	-15	-14
2	-16,-15,-14	-14	-14	-16	-14	-14	4.3e-13
3	-16,-15,-14	-14	-15	-16	-14	5.e-13	5.e-11
4	-16,-15,-14	-14	-14	-14	4.4e-13	4.6e-11	4.5e-9
5	-16,-15,-14	-15	-15	3.1e-13	3.1e-11	3.1e-9	3.1e-7
6	-16,-15,-14	-15	1.4e-13	1.4e-11	1.4e-9	1.4e-7	1.4e-5
7	-16,-15,-14	-14	2.7e-12	2.7e-10	2.7e-8	2.7e-6	2.5e-4
8	-16,-15,-14	9.4e-13	9.4e-11	9.4e-9	9.4e-7	9.2e-5	7.6e-3
oths	-16,-15$\nearrow$-7,-6	-6,-5	-5,-4	-4,-3	-3,-2	-2,-1	-1,0

The eigenvalues we selected suffer from perturbations that, as soon as the inherent errors become dominant, grow quadratically. The other eigenvalues suffer from perturbations growing linearly. It is worth adding some details. We observe that:

i) for all the selected eigenvalues, the inherent error becomes dominant when the order of magnitude of the observed error is 10^{-13};
ii) what observed in i) happens for values of μ that decrease with the index of the considered eigenvalue.

We infer from i) and ii) that:

a) the complement to the inherent error in the observed one has the same order of magnitude no matter what the considered eigenvalue and μ (the magnitude of the perturbation in A) are;
b) the coefficient of the quadratic term in the expansion of $\lambda(A+\alpha\, E)-\lambda(A)$ in powers of α becomes larger with the index of the selected eigenvalue (this seems to be linked with the observation that even the condition numbers do the same);
c) in the case of the first eigenvalue, the coefficient in point b) is small enough to make the inherent error not detectable in Table 5.

Algorithm 1. As before. ‘mds’.
The value of the ratio in (3.1), reached in 348 iterations (12559 function evaluations), is 2.e-13.

Table 5a

μ/λ^*	$0\nearrow 15$
1:8	-16,-15,-14
oths	-16,-15,-14$\nearrow$-9$\to$-1

Comparison. This time Algorithm 1 found a much better matrix E. This highlights the opportunity of having at our disposal two algorithms.

– **Test 6. Frank matrix**

We treated this famous upper Hessenberg matrix of dimension 12 [24], [3, Section 13], [26, Section 5] using both the algorithms.

Algorithm 2. A=**gallery**(‘frank’,12). The best conditioned eigenvalue (4th). ‘uhess’. $p = 0$. Real.
$\kappa(\lambda^*(4))$=1.7109e+0. The modulus of the ratio in (2.2) is 6.e-17.

Table 6

μ/λ^*	$1\nearrow 10$	11	12	13	14
4	-15,-14	4.9e-12	4.9e-10	4.9e-8	4.9e-6
oths	-15$\to$-8$\nearrow$-8$\to$-1	-7$\to$-1	-6$\to$-1	-5$\to$-1	-4$\to$0

Algorithm 1. As before. ‘mds’.
The value of the ratio in (3.1), reached in 20 iterations (7922 function evaluations), is 4.e-17.

Table 6a

μ/λ^*	$1\nearrow 10$	11	12	13	14
4	-16,-15,-14	9.8e-12	9.8e-10	9.8e-8	9.8e-6
oths	-15$\to$-8$\nearrow$-9$\to$-1	-8$\to$-1	-7$\to$-1	-6$\to$-1	-5$\to$-1

Comparison. In this case the two algorithms found two matrices E yielding essentially equivalent results. Both the tables show very satisfactory outcomes. Notice that two digits in the mantissa remain unaltered, while the exponent has a two-points decrease at each step.

– Test 7. Bessel matrix

The Bessel matrices are tridiagonal matrices associated with the Ordinary Bessel Polynomials. In fact, their eigenvalues are the zeros of the OBP's [4]. They differ from skew-symmetric tridiagonal matrices by the rank-one matrix $-e_1 e_1^T$, but, in spite of this, they have very badly conditioned eigenvalues [17]. Here the Bessel matrix of dimension 25 is considered. Its spectrum $\sigma(A)$ lies in the left half of the complex plane (see e.g. [4] or have a look at the squares in the figures related to Test 11).

This test is reported from [14], adding in Table 7 details to the Table 7 therein. At that time we tried a $m = \nu$ case to bring an example of a test that could not be treated with Algorithm 2. Now that Algorithm 2 has been improved (it works if the rank r of R (see Section 4) is less than m), it deserves to be tried.

Algorithm 1. A=**full**(**gallery**('tridiag',1./**sqrt** (4*(1:24).^2-1),[-1 **zeros** (1,24)],-1./**sqrt** (4*(1:24).^2-1))). The two complex eigenvalues closest to the imaginary axis (1st, 2nd). 'sparse'. [24 25 25 24]. Real. 'mds'.

$\kappa(\lambda^*(1)) = \kappa(\lambda^*(2))$ =5.6256e+2. The value of the ratio in (3.1), reached in 25 iterations (211 function evaluations), is 0.

Table 7

μ/λ^*	$0\nearrow 8$	9	10	11	12	13	14
1,2	-16,-15	4.6e-13	4.6e-11	4.6e-9	4.6e-7	4.5e-5	2.0e-3
oths	$-15\to-10\nearrow-13\to-4$	$-13\to -5$	$-11\to-5$	$-9\to-5$	$-7\to-6$	$-5\to-4$	$-3\to-2$

Again, our results agree with those theoretically predictable.

Algorithm 2. As before (except for 'mds'). The moduli of the ratios in (2.2) are both equal to 9.e-17.

The rank-revealing QR factorization with column pivoting says that the rank r of R (see Section 4) is 1 (this makes it useless to compute the condition number of the $r \times r$ coefficient matrix of the nonhomogeneous linear system in Section 4, point 2 (c)). The table is identical to Table 7 and is omitted.

Comparison. In this case both the algorithms yield a same (real) matrix E. This is connected with the fact that, according to Theorem 2.1, the dimension of the subspace of the solutions to the homogeneous linear system (2.9) is 1.

5.2.2 – A surprising test

Of course, Algorithm 2 does not work if $r = m = \nu$ since no matrix solution E exists in this case (see Theorem 2.1). However, appreciable results might be achieved in such situations by Algorithm 1. We refer the reader to [14, Table 8] for an example.

Even in the test we report in this section, Algorithm 2 does not work ($r = m = \nu = 4$), but the results yielded by Algorithm 1 are very good.

– **Test 8. Randhess matrix**

This is a random, orthogonal upper Hessenberg matrix. Since MATLAB's **rand** function is used in the code that constructs the matrix A, we inserted before the command **rand**('seed',0) to make the test repeatable.

Algorithm 1. A=**gallery**('randhess',50). The two complex and conjugate pairs of eigenvalues closest to the imaginary axis (1st, 2nd, 32th, 33th). 'sparse'. [9 36 11 38 13 40 15 42]. Real. 'mds'.

$\kappa(\lambda^*(1))$=$\kappa(\lambda^*(2))$=$\kappa(\lambda^*(32))$=$\kappa(\lambda^*(33))$=1 (A is a normal matrix). The value of the ratio in (3.1), reached in 14 iterations (109 function evaluations), is 1.e-16.

Table 8

μ/λ^*	$0\nearrow 16$
1,2	-16,-15
32,33	-16,-15
oths	-16,-15$\nearrow$-13$\rightarrow$-3

An explanation for these surprising results can be found by having a look at the weight distributions of the Wilkinson perturbation matrices $W_{\lambda(h)} = y^{(h)}\, x^{(h)H}$, $h = 1, 2, 32, 33$ (see [15] for more).

5.3 – Ill-conditioned eigenvalues and observed error

The eigenvalues we selected in the tests discussed until now were in any case quite well-conditioned. We made that choice to better document the consistency of the behavior of the observed errors with our theoretical results. As a matter of fact, the ill-conditioning of a selected eigenvalue complicates the interpretation of the outcome, since a classical Error Analysis result [3, Section 12, at the very beginning] implies, in our context, that the algorithmic error can even reach, approximately, the size $\kappa(\lambda)\,\mathbf{eps}\,\|A\|$. As a consequence of the large value of $\kappa(\lambda)$, sometimes the effects of the perturbation on A become observable only for large values of μ.

Moreover, we actually annihilate (numerically) the coefficient $r(0){=}y^H E\,x/y^H x$ of the linear term in the expansion in powers of α, at $\alpha = 0$, of the inherent error on λ. The more α grows, the farther the direction E we found is, in general, from the direction which would yield $r(\alpha) = 0$. This means that the more α grows, the less the linear term in the expansion of the inherent error on the eigenvalue λ_α of the matrix $A + \alpha E = A + 10^\mu \mathbf{eps}\, E$ is negligible. A usually very large upper bound for $|r(\alpha)|$ can be estimated equal to $\kappa(\lambda_\alpha)$, which, in general, has to be supposed less than or equal to $\kappa(\lambda)$ (volcano effect, lava flow rate). Sometimes, as a consequence of the large value of $\kappa(\lambda)$ in case of ill-conditioning, as soon as the inherent error becomes the principal part of the error, $|r(\alpha)|$ assumes values so large that one observes a linear behavior of the induced perturbation on λ that dominates the quadratic term.

– Test 9. Wilkinson matrix

This is one of the famous Wilkinson matrices. Its dimension is 10. The zero-structure we considered generalizes the one discussed in an example in [23].

Algorithm 2. A=**diag**(10:-1:1)+**diag**(10***ones** (9,1),1). The worst conditioned eigenvalues (5th, 6th). ‘lhess’. $p = n$. The diagonal entries. Real.

$\kappa(\lambda^*(5)){=}\kappa(\lambda^*(6))$ =4.2810e+5. The 2-norm condition number of the 2×2 coefficient matrix of the nonhomogeneous linear system in Section 4, point 2 (c), is 2.40584e+1. The moduli of the ratios in (2.2) respectively are 8.e-12 and 1.e-11.

Table 9

μ/λ^*	$0\nearrow 9$	10	11	12	13
5	-11,-10	3.e-8	3.e-6	5.e-4	2.e-2
6	-11,-10	3.e-8	3.e-6	5.e-4	1.e-2

Algorithm 1. As before. ‘mds’.

The value of the ratio in (3.1), reached in 62 iterations (10099 function evaluations), is 3.e-15.

Table 9a

μ/λ^*	$0\nearrow 9$	10	11	12	13	14
5	-11,-10	3.e-8	3.e-6	2.e-4	7.e-3	9.e-2
6	-11,-10	3.e-8	3.e-6	2.e-4	6.e-3	7.e-2

Comparison. In both the cases, the inherent error $|\lambda(A)-\lambda(A+10^{\mu}\mathbf{eps}\,E)|$ is negligible while μ ranges from 0 to 9. Then it can be appreciated, becoming the principal part of the observed error, which grows quadratically in Table 9. In Table 9a, the observed error behaves quadratically only for a while. Then, apparently due to the reasons stated above, it goes linearly.

Some of the uncontrolled eigenvalues have a condition number slightly less than that of the eigenvalues we selected. Sometimes, in similar situations, it is very difficult to observe the behavior of the uncontrolled eigenvalues, due to their high sensitivities. This is what happened in this case. Thus, we do not report the row "oths".

– **Test 10. Frank matrix**

As above, we treated it using both the algorithms.

Algorithm 2. A=**gallery**('frank',12). The worst conditioned eigenvalue (9th). 'uhess'. $p = 0$. Real.
$\kappa(\lambda^*(9))$=3.8774e+7. The modulus of the ratio in (2.2) is 4.e-9.

Table 10

μ/λ^*	$1\nearrow 9$	10	11	12	13	14	15	16
9	-10,-9,-8	4.6e-7	4.7e-6	4.7e-5	4.7e-4	4.7e-3	4.8e-2	5.e-1
oths	$-16\to-9\nearrow-9\to-2$	$-8\to-2$	$-7\to-1$	$-6\to-1$	$-5\to-1$	$-4\to0$	$-4\to0$	$-2\to0$

Here the induced perturbation on the selected (strongly ill-conditioned) eigenvalue becomes observable only for large values of μ, and the quadratic behavior does not appear at all. We can note the moderate size of the induced perturbation.

Even in this case uncontrolled eigenvalues are ill-conditioned. However, the checks we carried out on them gave reliable results. So, in this case, we decided to report even the row "oths".

Algorithm 1. As before. 'mds'.

The value of the ratio in (3.1), reached in 27 iterations (9168 function evaluations), is 0.

In spite of the annihilation of the ratio in (3.1), the observed results, even if qualitatively analogous to those reported above, are less satisfactory. So we do not report the table.

– Test 11. Bessel matrix

As above, the Bessel matrix of dimension 25 is treated using both the algorithms.

Algorithm 2. A=**full**(**gallery**('tridiag',1./**sqrt** (4*(1:24).^2-1),[-1 **zeros** (1,24)],-1./**sqrt** (4*(1:24).^2-1))). The worst conditioned eigenvalue (25th). 'sparse'. [1 1 2 1 1 2]. Real.

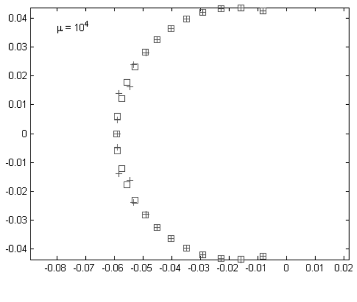
$\kappa(\lambda^*(25))$=3.9411e+12. The modulus of the ratio in (2.2) is 0.

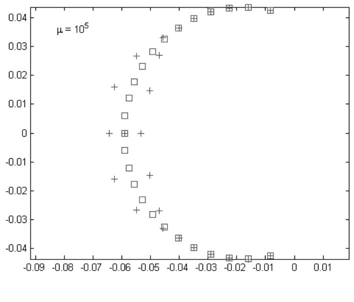
In this case, the best thing to do to illustrate the outcome is to report some spectral plots. We selected eight plots. They concern the cases of $\mu = 4 : 1 : 11$. The eigenvalues of A are marked with squares while the eigenvalues of $A + 10^{\mu}\mathbf{eps}\,E$ are marked with plus signs. The plots highlight how large the perturbation on the uncontrolled eigenvalues is with respect to that on the selected eigenvalue, which remains essentially unaltered.

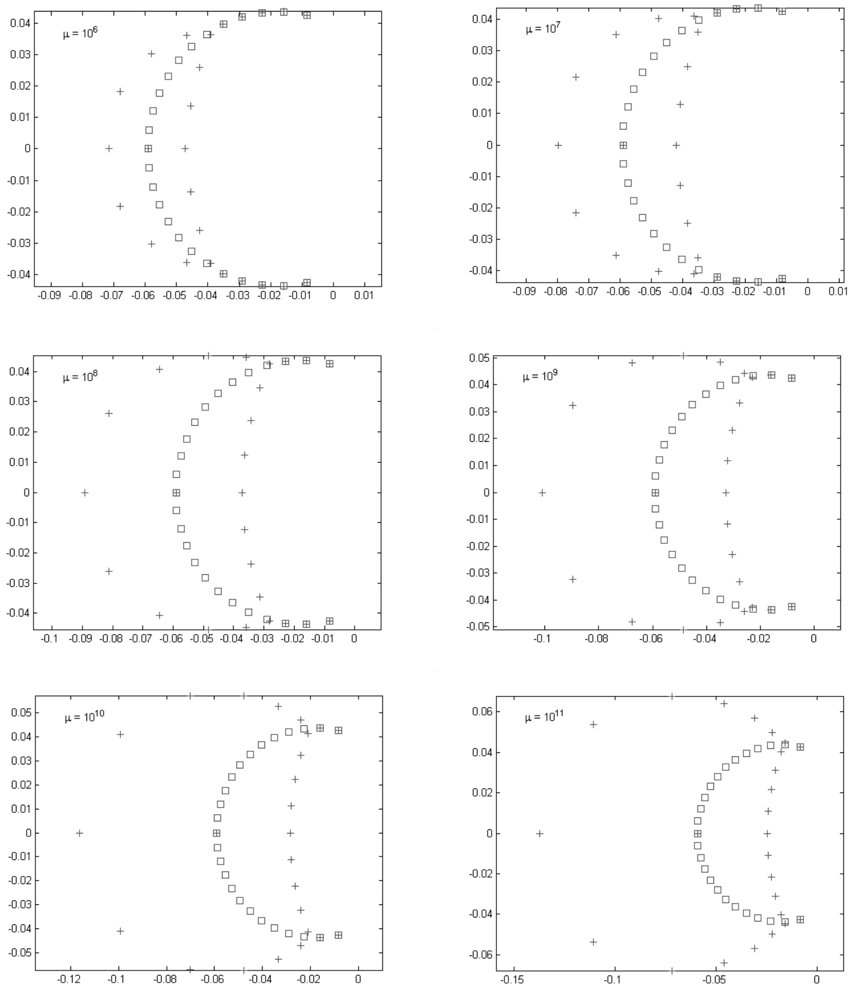
Algorithm 1. As before. 'nms'.

The value of the ratio in (3.1) is 0, reached in 161 (321 function evaluations) iterations.

This time, results (and plots) are totally in line with those given by Algorithm 2. Note that **mdsmax** fails in this case.







6 – Conclusions

Given a matrix $A \in \mathbb{C}^{n\times n}$, we numerically construct zero-structured unit-norm matrices E in such a way that $A + \alpha\, E$ keeps essentially unaltered a set of ν simple eigenvalues of A for reasonably large values of α.

The ν eigenvalues and the zero-structure can be arbitrarily selected. Theorem 2.1 states that such matrices E exist if and only if condition (2.11) holds.

It is shown, however, that we may construct satisfactory matrices E even if, actually, no solution exists.

We present improved versions of the two algorithms in [14]. They have complementary qualities and drawbacks.

Algorithm 2 is quite cheap. It works efficiently every time solution matrices E exist and even in case of large systems (2.9).

Algorithm 1 can be useful even when no solution exists (see, for an interesting example, Test 8). It is theoretically able to carry out any test, provided that, to keep the computational cost reasonable, n, m and ν are not too large.

The MATLAB codes of both the algorithms are available upon request.

Selected tests illustrate the behavior of the inherent and observed errors.

Our results agree with the theory outlined in Section 2, even in the case of ill-conditioned eigenvalues. With regard to this last case, a significant example is given by the spectral plots of the perturbed Bessel matrix (Test 11).

REFERENCES

[1] E. Anderson – Z. Bai – C. H. Bischof – S. Blackford – J. W. Demmel – J. J. Dongarra – J. J. Du Croz – A. Greenbaum – S. J. Hammarling – A. McKenney – D. C. Sorensen: *LAPACK Users' Guide*, third edition, SIAM, (1999), http://www.netlib.org//lapack/lug/.

[2] G. H. Golub – C. F. Van Loan: *Matrix Computations*, third edition, The Johns Hopkins University Press, Baltimore and London, 1996.

[3] G. H. Golub – J. H. Wilkinson: *Ill–conditioned eigensystems and the computation of the Jordan canonical form*, SIAM Review, **18** (1976), 578-619.

[4] E. Grosswald: *Bessel Polynomials*, Lecture Notes in Mathematics, vol. 698, Springer–Verlag, New York, 1978.

[5] D. J. Higham – N. J. Higham: *Backward Error and Condition of Structured Linear Systems*, SIAM J. Matrix Anal. Appl., **13** (1) (1992), 162-175.

[6] N. J. Higham: *The Matrix Computation Toolbox*, http://www.ma.man.ac.uk/~higham/mctool-box.

[7] N. J. Higham: *The Accuracy of Solutions to Triangular Systems*, SIAM J. Numer. Anal., **26** 5 (1989), 1252-1265.

[8] N. J. Higham: *Optimization by direct search in matrix computations*, SIAM J. Matrix Anal. Appl., **14** (2) (1993), 317-333.

[9] N. J. Higham: *Accuracy and Stability of Numerical Algorithms*, second edition, SIAM, Philadelphia, 2002.

[10] M. Karow – D. Kressner – F. Tisseur: *Structured Eigenvalue Condition Numbers*, SIAM J. Matrix Anal. Appl. **28**(4) (2006), 1052-1068.

[11] C. T. Kelley: *Iterative Methods for Optimization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.

[12] H. W. J. Lenferink – M. N. Spijker: *On the use of stability regions in the numerical analysis of initial value problems*, Math. Comp., **57** (1991), 221-237.

[13] S. Noschese – L. Pasquini: *Eigenvalue Patterned Condition Numbers: Toeplitz and Hankel Cases*, to appear in J. Comput. Appl. Math.

[14] S. Noschese – L. Pasquini: *How to Find Matrix Modifications Keeping Essentially Unaltered a Selected Set of Eigenvalues*, in "Linear Algebra Proceedings", SIAM Conference on Applied Linear Algebra, Williamsburg, VA, July, 15-19, 2003; CP5 in http://www.siam.org/meetings/la03/proceedings/.

[15] S. Noschese – L. Pasquini: *Eigenvalue Condition Numbers: Zero-Structured versus Traditional*, J. Comput. Appl. Math., **185** (2006), 174-189.

[16] B. N. Parlett – C. Reinsch: *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numer. Math., **13** (1969), 292-304; see also: Handbook for Automatic Computation, Linear Algebra, J. H. Wilkinson and C. Reinsch, (eds.), 1971, Springer–Verlag, New York, pp. 315-326.

[17] L. Pasquini: *Accurate Computation of the Zeros of the Generalized Bessel Polynomials*, Numer. Math., **86** (3) (2000), 507-538.

[18] A. Ruhe: *Closest normal matrix finally found!*, BIT, **27** (1987), 585-598.

[19] G. W. Stewart: *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[20] F. Tisseur: *A Chart of Backward Errors for Singly and Doubly Structured Eigenvalue Problems*, SIAM J. Matrix Anal. Appl., **24** (3) (2003), 877-897.

[21] V. J. Torczon: *On the convergence of the multidirectional search algorithm*, SIAM J. Optimization, **1** (1991), 123-145.

[22] L. N. Trefethen: *Pseudospectra of matrices*, in "Numerical Analysis 1991", D. F. Griffiths and G. A. Watson (eds.), vol. 260, pp. 234-266, Longman, 1992.

[23] D. S. Watkins: *Fundamentals of Matrix Computations*, John Wiley and Sons, New York, 1991.

[24] J. H. Wilkinson: *Rounding errors in algebraic processes*, Her Majesty's Stationery Office, (1963).

[25] J. H. Wilkinson: *The algebraic eigenvalue problem*, Clarendon Press, Oxford, England, 1965.

[26] J. H. Wilkinson: *Sensitivity of eigenvalues*, II, Utilitas Mathematica, **30** (1986), 243-286.

Lavoro pervenuto alla redazione il 13 novembre 2006
ed accettato per la pubblicazione il 4 dicembre 2006.
Bozze licenziate il 14 maggio 2007

Indirizzo degli Autori:

S. Noschese – L. Pasquini – Dipartimento di Matematica "Guido Castelnuovo" – Università di Roma "La Sapienza" – P.le A. Moro, 2 – 00185 Roma, Italy
E-mail: noschese@mat.uniroma1.it lionello.pasquini@uniroma1.it

This work has been performed within the activities of the National GNCS-INDAM Research Project: "Algebra Lineare Numerica per matrici con struttura" and supported by the related funds.