Numer Algor (2012) 60:193–204
DOI 10.1007/s11075-012-9537-2

ORIGINAL PAPER

The computation of isotropic vectors

Gérard Meurant

Received: 16 October 2011 / Accepted: 15 January 2012 /
Published online: 14 March 2012

Abstract We describe algorithms to compute isotropic vectors for matrices with real or complex entries. These are unit vectors b satisfying $b^*Ab = 0$. For real matrices the algorithm uses only the eigenvectors of the symmetric part corresponding to the extreme eigenvalues. For complex matrices, we first use the eigenvalues and eigenvectors of the Hermitian matrix $K = (A - A^*)/2i$. This works in many cases. In case of failure we use the Hermitian part H or a combination of eigenvectors of H and K. We give some numerical experiments comparing our algorithms with those proposed by R. Carden and C. Chorianopoulos, P. Psarrakos and F. Uhlig.

Keywords Isotropic vectors · Eigenvectors

1 Introduction

Given a nonsingular square matrix A of order n with real or complex coefficients, we are interested in computing non-trivial vectors b with real or complex entries and unit norm such that

$$b^*Ab = 0, \tag{1}$$

where the * denotes the conjugate transpose. In the literature on quadratic forms, such a vector b is said to be isotropic. Note that this is a special instance

This paper is dedicated to Claude Brezinski on the occasion of his 70th birthday.

G. Meurant (✉)
30 rue du sergent Bauchat, 75012, Paris, France
e-mail: gerard.meurant@gmail.com

of a more general problem stated by F. Uhlig [10]: how to compute a vector b, of unit norm, such that

$$b^*Ab = \mu, \tag{2}$$

where μ is a given complex number. In [10] a heuristic geometric algorithm for computing such a vector was described. It is straightforward to see that the problem (2) reduces to (1) for a different matrix since (2) is equivalent to

$$b^*(A - \mu I)b = 0.$$

If μ is an eigenvalue of A, then a corresponding eigenvector of A gives a solution. If not, $A - \mu I$ is nonsingular and we have to compute an isotropic vector for that matrix. Note that even if A is real, we have to deal with a complex matrix when the shift μ is complex. Hence, from now on, we will assume that the target value is zero. While we were working on this problem in 2009, Carden [1] published an algorithm that is both simpler and faster than Uhlig's first algorithm. More recently, Chorianopoulos, Psarrakos and Uhlig [2] proposed a new algorithm that is allegedly faster than Carden's. In these algorithms, most of the computing time is spent computing eigenvalues and eigenvectors of the Hermitian or skew-Hermitian parts of A. Hence, what is important is the number of eigenanalyses that have to be done to obtain a solution. The minimum number of eigenanalyses in [1] and [2] is two. We will see that, when the matrix is real and in some cases for complex matrices, one can obtain solutions with only one eigenanalysis. For real matrices we will describe a very simple algorithm needing only the extreme eigenvalues and corresponding eigenvectors of the symmetric part of A. For complex matrices the situation is more difficult but, in many cases, a solution can also be obtained with only one eigenanalysis.

Our interest in computing isotropic vectors is related to the study of partial stagnation of the GMRES algorithm for solving linear systems with real matrices; see Saad and Schultz [9]. When starting from $x^0 = 0$, the conditions for complete stagnation are

$$b^*A^jb = 0, \quad j = 1, \ldots, n-1.$$

If we only have $b^*Ab = 0$, then we have stagnation for the first iteration, that is, $\|r^1\| = \|r^0\| = \|b\|$ where r^k is the residual vector at iteration k.

Note that the set of solutions of (1) is not a vector space; a linear combination of two solutions is not necessary a solution. The set

$$W(A) = \{x^*Ax \,|\, x \in \mathbb{C}^n, x^*x = 1\}, \tag{3}$$

is known as the field of values of A (or numerical range). Therefore, the problems (1) and (2) are often called the inverse field of values problem or the inverse numerical range problem. The field of values has been used to study convergence of some iterative methods for solving linear systems; see, for instance, Eiermann [4]. In order for (1) to have at least one solution, we need the origin to be in the field of values of A. The solution of (1) can be reduced to considering problems with Hermitian and skew-Hermitian matrices.

Theorem 1 *Let A and b have complex entries. We have the equivalence*

$$b^*Ab = 0 \Leftrightarrow b^*(A + A^*)b = 0 \text{ and } b^*(A - A^*)b = 0.$$

Proof Clearly, if $b^*Ab = 0$, we have $b^*A^*b = 0$, therefore the two conditions on the right are satisfied. Conversely, if we assume the conditions on the right, then by taking the sum, we obtain $b^*Ab = 0$. □

Note that this does not give polynomial equations for the components of b because of the conjugacy. If only the condition $b^*(A + A^*)b = 0$ is satisfied, remarking that $(b^*Ab)^* = b^*A^*b$, we just obtain that the real part of b^*Ab is zero. Similarly, if we have $b^*(A - A^*)b = 0$, then only the imaginary part of b^*Ab is zero. We will use these facts for computing a solution for complex matrices. When b and A are both real, the problem is much easier, since we only have to consider the symmetric part of A. We have the equivalence

$$b^TAb = 0 \Leftrightarrow b^T(A + A^T)b = 0.$$

We will denote respectively by $H = (A + A^*)/2$ and $\tilde{K} = (A - A^*)/2 = iK$ the Hermitian and skew-Hermitian parts of A.

In Section 2 we describe how to compute as many isotropic vectors as we wish for real matrices. This can be done using some eigenvectors of H. Section 3 considers complex matrices. In some cases we can compute solutions with only one eigenanalysis for the matrix K improving on the results in [1] and [2]. However, this does not always work. A remedy that may work is to use the eigenvectors of H. When all this fails, we rely on some techniques developed in [2]. Numerical experiments are described in Section 4. They show that in many cases, our algorithms are faster than those in [1, 2]. Finally, we give some conclusions.

2 Real matrices

When A is real, we are concerned with computing solutions of

$$b^*Hb = 0, \tag{4}$$

where H is real and symmetric (i.e. $H = H^T$). It is known (see, for instance, [6, 7]) that the field of values $W(A)$ is symmetric with respect to the real axis and 0 is in $W(A)$ if and only if $\omega_1 \leq 0 \leq \omega_n$ where ω_1 and ω_n are respectively the smallest and largest eigenvalues of H. Let X_1 and X_2 be the real eigenvectors corresponding to ω_1 and ω_2. Then, $X_1^TAX_1 = X_1^THX_1 = \omega_1$ and $X_n^TAX_n = X_n^THX_n = \omega_n$ are real and they are the left-most and right-most points of $W(A)$ on the real axis.

Computing real solutions of (4) can be easily done by using the eigenvectors of H. Let us assume that we are looking for vectors b of norm 1. The matrix H can be written as

$$H = X\Omega X^T,$$

where Ω is the diagonal matrix of the eigenvalues ω_i that are real numbers, and X is the orthonormal matrix of the eigenvectors such that $X^T X = I$. Then we use this spectral decomposition into (4),

$$b^* Hb = b^* X \Omega X^T b = 0.$$

Let $c = X^T b$ be the vector of the projections of b on the eigenvectors of H. We have the following result.

Theorem 2 *Let b be a solution of (4), the vector $c = X^T b$ with components c_i satisfy the equations*

$$\sum_{i=1}^{n} \omega_i |c_i|^2 = 0, \quad \sum_{i=1}^{n} |c_i|^2 = 1. \tag{5}$$

Proof Inserting the definition $c = X^T b$ into (4), we have

$$b^* Hb = b^* X \Omega X^T b = c^* \Omega c = 0$$

and since Ω is diagonal, $c^* \Omega c$ can be written as a sum of the components $\bar{c}_i \omega_i c_i = \omega_i |c_i|^2$. Moreover, $\|b\| = \|X^T c\| = \|c\| = 1$ which gives the other condition. □

Note that (5) involves only real numbers. From Theorem 2, 0 must be a convex combination of the eigenvalues ω_i. As we have already seen, this means that if A or $-A$ are positive real, then (4) does not have a non-trivial solution. Otherwise, 0 is in $W(A)$ and we can always find a real solution. In fact, when $n > 2$ there is an infinite number of solutions. Let us now show how to construct solutions. We first consider using two eigenvectors and then we will see how to compute more solutions using three eigenvectors.

Since not all the eigenvalues of H have the same sign, the smallest one ω_1 has to be such that $\omega_1 < 0$. Let $k > 1$, be such that $\omega_k > 0$ and t be a positive real number smaller than 1. Typically, we will use $k = n$. We set $|c_1|^2 = t$, $|c_k|^2 = 1 - t$ and $c_i = 0, i \neq 1, k$. From (5) we must have

$$\omega_1 t + \omega_k (1 - t) = 0,$$

whose solution is

$$t_s = \frac{\omega_k}{\omega_k - \omega_1}.$$

Note that since $\omega_1 < 0$, the denominator is positive as well as t_s and $t_s < 1$. The modulus of c_1 (resp. c_k) is the square root of t_s (resp. $1 - t_s$). Since $b = Xc$ and we can eventually change the signs, two real solutions are

$$b = \sqrt{t_s} X_1 + \sqrt{1 - t_s} X_k, \quad b = -\sqrt{t_s} X_1 + \sqrt{1 - t_s} X_k,$$

where X_1 and X_k are the eigenvectors corresponding to ω_1 and ω_k. The other possibilities for the signs give solutions that are in the same directions as these

two ones. Since the two terms in these solutions have the same denominator we can, in fact, write the solutions as

$$b = \sqrt{\omega_k}X_1 + \sqrt{|\omega_1|}X_k, \quad b = -\sqrt{\omega_k}X_1 + \sqrt{|\omega_1|}X_k,$$

Then the solution vectors have to be normalized to be of unit norm. It gives

$$b = \sqrt{\frac{\omega_k}{\omega_k + |\omega_1|}}X_1 + \sqrt{\frac{|\omega_1|}{\omega_k + |\omega_1|}}X_k, \quad b = -\sqrt{\frac{\omega_k}{\omega_k + |\omega_1|}}X_1 + \sqrt{\frac{|\omega_1|}{\omega_k + |\omega_1|}}X_k.$$

In principle, these b's can be multiplied by $e^{i\theta}$ but this gives solutions in the same directions. In fact, we can use every pair of positive and negative eigenvalues. This process can give as many real solutions as two times the number of pairs of eigenvalues of H with opposite signs if all the eigenvalues are different. The two solutions constructed before are independent and moreover, (as remarked in [3]), orthogonal if $\omega_k = -\omega_1$. We denote this algorithm as `Alg.R`.

When A and b are real we have proved the following result.

Theorem 3 *If A is real and such that A or $-A$ are not positive real, there exist at least two independent real isotropic vectors.*

After having derived `Alg.R`, we found a paper (probably unknown to most people in the linear algebra community) authored by two mechanical engineers [3] who proposed the same construction. However, we can go further on by using more eigenvectors to construct more solutions.

To prove that we have an infinite number of real solutions and compute some of them, we have to consider at least three distinct eigenvalues with different signs (when they exist). Assume that we have $\omega_1 < 0 < \omega_2 < \omega_3$ and let $t_1 = |c_1|^2, t_2 = |c_2|^2$. The equation to satisfy is

$$\omega_1 t_1 + \omega_2 t_2 + \omega_3(1 - t_1 - t_2) = (\omega_1 - \omega_3)t_1 + (\omega_2 - \omega_3)t_2 + \omega_3 = 0, \quad (6)$$

with the constraints $t_i \geq 0$, $i = 1, 2$ and $t_1 + t_2 \leq 1$. Hence we have

$$t_2 = \frac{\omega_3}{\omega_3 - \omega_2} - \frac{\omega_3 - \omega_1}{\omega_3 - \omega_2}t_1.$$

This defines a line in the (t_1, t_2) plane and we have to see if this line intersects the triangular region defined by the constraints on t_1 and t_2. The line intersect the t_1 axis at $\omega_3/(\omega_3 - \omega_1)$ which is positive and smaller than 1 since ω_1 is negative. The intersection with the t_2 axis is $\omega_3/(\omega_3 - \omega_2)$ which is larger than 1. This line has a negative slope. All the admissible values for t_1 and t_2 are given by a segment inside the triangle. Therefore, there is an infinite number of feasible positive pairs (t_1, t_2). This is illustrated in Fig. 1 for $\omega_1 = -1, \omega_2 = 1$ and $\omega_3 = 2$. The triangle is the region where the constraints are satisfied and the feasible values are on the bold segment. The case $\omega_1 < \omega_2 < 0 < \omega_3$ is similar except that the line crosses the t_2 axis below 1. Then we obtain the solutions b by combining the three corresponding eigenvectors. Here the

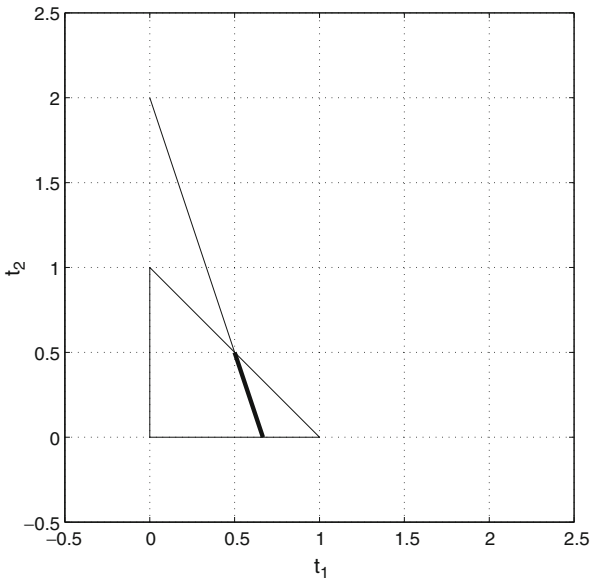

Fig. 1 $\omega_1 = -1, \omega_2 = 1$ and $\omega_3 = 2$

problem for the coefficients is in three dimensions and the feasible solutions for t live in a space of dimension one.

The previous construction proves the following result.

Theorem 4 *If $n > 2$ and A is real, such that A or $-A$ are not positive real and H has at least three distinct eigenvalues of different signs, there exist an infinite number of real isotropic vectors.*

We note that a similar result was proved in [3], but the proofs are not mathematically rigorous.

Of course, we can go on with the same idea. If we use four distinct eigenvalues of different signs, then we have to look at a problem in three dimensions. The region where the constraints are satisfied is a tetrahedron. We have to consider the intersection of a given plane with this tetrahedron. More generally, the equations that define the problem using k distinct (renamed) eigenvalues are

$$\sum_{i=1}^{k-1}(\omega_i - \omega_k)t_i + \omega_k = 0, \quad t_i \geq 0,\ i = 1, \ldots, k-1, \quad \sum_{i=1}^{k-1} t_i \leq 1.$$

The first equation defines an hyperplane and we have to look at the intersection of this hyperplane with the volume defined by the constraints.

If A is real we are finished, because we have just shown that we can compute as many real solutions as we wish. In particular we can eventually compute n independent solutions, even though this does not define an isotropic subspace. In [3] the question of the existence of an orthogonal basis was raised for

the real case. It was shown that such a basis can only exist if $trace(A) = 0$. However, it would be more interesting to look for vectors that are mutually "A-orthogonal" since this would lead to a matrix B such that $B^*AB = 0$ and the columns of B would generate an isotropic subspace.

3 Complex matrices

Unfortunately, if A has complex elements the previous constructions give only vectors for which $Re(b^*Ab) = 0$. In this section we consider two different algorithms for computing isotropic vectors when the matrix A is complex.

First, we remark that, in some cases, using a similar algorithm as in the previous section, we can find a set of solutions for the Hermitian matrix $H = (A + A^*)/2$ with zero real parts having positive and negative imaginary parts. When using three eigenvectors of H, there exist an infinite number of solutions that are obtained along the segment inside the triangle of constraints; see Fig. 1. When varying continuously the point along this segment, the imaginary part of the solution varies continuously. If the imaginary parts corresponding to the two ends of the segment are of different signs, then by the mean value theorem, there is a point on the segment giving a zero imaginary part. This point can be computed by dichotomy. Note that this only involves computation of the quadratic form x^*Ax. We do not need any new eigenanalysis. However, the change of sign in the values x^*Ax do not happen for any triplet of eigenvalues. Unfortunately, we were not able to characterize the set of matrices for which this is true. We denote this algorithm by `Alg.1`.

The second algorithm first use the eigenvalues and eigenvectors of the matrix $K = (A - A^*)/(2i)$ which is Hermitian. The algorithm of Section 2 gives vectors b such that $Im(b^*Ab) = 0$. Combining the eigenvectors of K corresponding to positive and negative eigenvalues, we can (in some cases) obtain two vectors b_1 and b_2 such that $\alpha_1 = Re(b_1^*Ab_1) < 0$ and $\alpha_2 = Re(b_2^*Ab_2) > 0$. Then we use the following result in [6].

Lemma 5 [6] *Let b_1 and b_2 two unit vectors with $Im(b_i^*Ab_i) = 0$, $i = 1, 2$ and $\alpha_1 = Re(b_1^*Ab_1) < 0$, $\alpha_2 = Re(b_2^*Ab_2) > 0$. Let $b(t,\theta) = e^{-i\theta}b_1 + tb_2$, $t, \theta \in \mathbb{R}$, $\alpha(\theta) = e^{i\theta}b_1^*Ab_2 + e^{-i\theta}b_2^*Ab_1$. Then*

$$b(t,\theta)^*Ab(t,\theta) = \alpha_2 t^2 + \alpha(\theta)t + \alpha_1,$$

$\alpha(\theta) \in \mathbb{R}$ when $\theta = arg(b_2^*Ab_1 - b_1^T\bar{A}\bar{b}_2)$. For $t_1 = (-\alpha(\theta) + \sqrt{\alpha(\theta)^2 - 4\alpha_1\alpha_2})/(2\alpha_2)$, we have

$$b(t_1,\theta) \neq 0, \quad \frac{b(t_1,\theta)^*}{\|b(t_1,\theta)\|} A \frac{b(t_1,\theta)}{\|b(t_1,\theta)\|} = 0.$$

Lemma 5 shows how to compute a solution from b_1 and b_2. If we have b_1 and b_2 such that $\alpha_1 = Re(b_1^*Ab_1) < 0$ and $\alpha_2 = Re(b_2^*Ab_2) > 0$ we are

finished. Note that when A is real and taking $b_1 = X_1, b_2 = X_2$, the eigenvectors of H, then $\theta = 0$ and Lemma 5 shows how to compute one isotropic vector.

However, in the complex case, we cannot always find suitable vectors b_1 and b_2. In particular, Y_i being the eigenvectors of K, if all the values $Re(Y_i^* H Y_j)$ have the same sign, the algorithm fails. An extreme example is a Jordan block with a complex value α on the diagonal and elements 1 on the diagonal above. Then we have $Re(Y_i^* H Y_j) = 0$, $i \neq j$ and $Y_i^* H Y_i = -Re(\alpha)$. Therefore, the real parts of $b^* A b$ for all the vectors b that can be generated are the same.

When we are not able to obtain the vectors b_1 and b_2 needed in Lemma 5, we compute the eigenvectors of H and we apply the same technique to the matrix iA. In case of failure, now that we have at hand the eigenvectors of K and H, we use the following method which was suggested to us by what is done in [2]. We combine eigenvectors of H and K. Let x (resp. y) be an eigenvector of K (resp. H), we consider the vectors $X_\theta = \cos(\theta)x + \sin(\theta)y$, $0 \leq \theta \leq \pi$. When θ goes from 0 to π, $X_\theta^* A X_\theta$ describes an ellipse within the field of values. For a given pair of eigenvectors x, y we look for intersections of the ellipse with the real axis. Noticing that $A = H + iK$, we have

$$\begin{aligned} X_\theta^* A X_\theta = {} & \cos^2(\theta)(x^* H x + i x^* K x) \\ & + \sin^2(\theta)(y^* H y + i y^* K y) \\ & + \sin(\theta)\cos(\theta)(x^* H y + y^* H x + i[x^* K y + y^* K x]). \end{aligned}$$

Let $\alpha = Im(x^* H x + i x^* K x)$, $\beta = Im(y^* H y + i y^* K y)$ and $\gamma = Im(x^* H y + y^* H x + i[x^* K y + y^* K x])$. Asking for the imaginary part of $X_\theta^* A X_\theta$ to be zero, we obtain the equation

$$\alpha \cos^2(\theta) + \beta \sin^2(\theta) + \gamma \sin(\theta)\cos(\theta) = 0.$$

Assuming $\cos(\theta) \neq 0$ and dividing, we obtain a quadratic equation for $t = \tan(\theta)$,

$$\beta t^2 + \gamma t + \alpha = 0.$$

If this equation has real solutions, then we obtain values of θ which give vectors X_θ such that $Im(X_\theta^* A X_\theta) = 0$. When the size of the problem is large, we do not apply these techniques to all pairs of eigenvectors since this may be too costly. We just use the eigenvectors corresponding to a few of the smallest and largest eigenvalues.

If this third stage also fails, we switch to the algorithms described in [2]. They first use the eigenvectors of H and K that are already computed. However, this does not always give a suitable vector. The remedy is then to rotate the matrix, multiplying by $e^{i\theta}$ for well-chosen angles θ, but this requires other computations of eigenvectors; see [2]. We denote this multi-stage algorithm by `Alg.2`. Of course, when it works, the first stage of our algorithm must be faster since it uses only one eigenanalysis contrary to what is done in [2]. When the first stage or the second stage fail, then this is, of course, a waste of time and then, the algorithms of [2] or [1] are faster. Another point to note is that

the computing times depend on the order by which the eigenvectors of H and K are considered.

4 Numerical examples

The computations were done with Matlab 7 (R14). To have fair comparisons with the algorithms in [2] and [1], we will always use the QR algorithm for computing all the eigenvalues and eigenvectors we need in all algorithms. Let us start with a real matrix. The first example is a random real matrix of order 100 obtained with the Matlab function `randn`. Table 1 gives the computing times and the value of $|b^T Ab|$ for the algorithm of Section 2 and the algorithms in [2] (we used the March 2011 version of the code) and [1] for computing one isotropic vector. `Alg.R` is faster because it uses only one computation of eigenvalues and eigenvectors for H. The absolute value of the Rayleigh quotient is larger by one or two orders of magnitude but still acceptable. With the algorithm of Section 2 we were able to compute 100 different isotropic vectors in 0.022 seconds with only one eigenanalysis. However, the rank of the matrix of the solutions was only 50.

Table 2 displays the results for the same matrix and a complex shift $\mu = 1 + 8i$. This makes the problem complex and we cannot use `Alg.R` any longer. However, for this value of the shift μ, `Alg.2` is working using only the eigenvectors of K and therefore, faster than the other algorithms. The letter K within parenthesis indicates that only the eigenvectors of K were involved. When we have to use the other stages of the algorithms, we use letters H and E. Note that for random matrices the eigenvalues are usually spread into the field of values. We see in Tables 1 and 2 that `Alg.R` and `Alg.2` are faster. This is because, in these examples, they only use one computation of eigenvalues and eigenvectors.

The second example comes from Liesen and Strakoš in [8]; see also [5]. They discretized

$$-\nu\Delta u + w \cdot \nabla u = 0,$$

Table 1 Random matrix of order 100, $\mu = 0$

Algorithm	Time (s)	$\lvert b^T Ab \rvert$
`Alg.R`	0.017	$6.2617\ 10^{-14}$
Alg. [2]	0.057	$1.6012\ 10^{-15}$
Alg. [1]	0.077	$3.4417\ 10^{-15}$

Table 2 Random matrix of order 100, $\mu = 1 + 8i$

Algorithm	Time (s)	$\lvert b^T Ab - \mu \rvert$
`Alg.2` (K)	0.037	$4.5989\ 10^{-15}$
Alg. [2]	0.060	$1.5424\ 10^{-15}$
Alg. [1]	0.070	$7.1089\ 10^{-15}$

Table 3 Example 2, $\mu = 0.02$

Algorithm	Time (s)	$\lvert b^T Ab \rvert$
`Alg.R`	0.14	$4.8833\ 10^{-17}$
Alg. [2]	0.43	$1.2081\ 10^{-17}$
Alg. [1]	0.60	$1.0971\ 10^{-17}$

Table 4 Example 2, $\mu = 0.055 + 0.02\,i$

Algorithm	Time (s)	$\lvert b^T Ab - \mu \rvert$
`Alg.1`	0.49	$8.2506\ 10^{-15}$
`Alg.2` (K)	0.29	$4.9874\ 10^{-17}$
Alg. [2]	0.44	$3.4964\ 10^{-18}$
Alg. [1]	0.51	$3.1273\ 10^{-17}$

with $w = [0, 1]^T$ in $\Omega = (0, 1)^2$ with Dirichlet boundary conditions $u = g$ on $\partial\Omega$ using a stabilized Petrov–Galerkin SUPG method with bilinear finite elements on a regular Cartesian mesh. The matrix is

$$A = \nu N \otimes M + M \otimes ((\nu + \delta h)N + C),$$

where δ is the stabilization parameter, h is the mesh size and

$$M = \frac{h}{6}\text{tridiag}(1, 4, 1), \quad N = \frac{1}{h}\text{tridiag}(-1, 2, -1), \quad C = \frac{1}{2}\text{tridiag}(-1, 0, -1),$$

are tridiagonal matrices with constant diagonals. We use $h = 1/16$, $\nu = 0.01$ and $\delta = 0.34$. This gives a real matrix of order 225. However, the origin is not in the field of values. Hence we shift the matrix by -0.02, but it is still real and we can use `Alg.R`. Results are given in Table 3. For this example, `Alg.R` is about three times faster than the algorithm of [2]. Table 4 displays results for a complex shift $\mu = 0.055 + 0.02\,i$ which is inside the convex hull of the eigenvalues of A. `Alg.2` is faster than `Alg.1` and the algorithms from [2] and [1]. In Table 5 we use another shift $\mu = 0.055 + 0.04\,i$ which is outside the convex hull of the eigenvalues of A. Only using the eigenvectors of K still works in this case and we are faster than [2] and [1].

Table 5 Example 2, $\mu = 0.055 + 0.04\,i$

Algorithm	Time (s)	$\lvert b^T Ab - \mu \rvert$
`Alg.1`	1.52	$3.5382\ 10^{-17}$
`Alg.2` (K)	0.32	$2.9219\ 10^{-18}$
Alg. [2]	0.44	$1.9516\ 10^{-18}$
Alg. [1]	0.60	$2.0835\ 10^{-17}$

Table 6 Example 3, $\mu = 5000 + 10000\,i$

Algorithm	Time (s)	$\lvert b^T Ab - \mu \rvert$
`Alg.1`	0.20	$9.7386\ 10^{-11}$
`Alg.2` (K)	0.10	$1.8645\ 10^{-11}$
Alg. [2]	0.18	$2.1398\ 10^{-12}$
Alg. [1]	0.16	$1.3690\ 10^{-12}$

Table 7 Example 3, $\mu = 10000 + 10000\,i$

Algorithm	Time (s)	$\lvert b^T Ab - \mu \rvert$
`Alg.1`	0.22	$3.5156\ 10^{-12}$
`Alg.2` (K + H)	0.175	$3.5156\ 10^{-12}$
Alg. [2]	0.18	$8.1981\ 10^{-13}$
Alg. [1]	0.17	$6.9563\ 10^{-12}$

Table 8 Example 3, $\mu = 12000 + 10000\,i$

Algorithm	Time (s)	$\lvert b^T Ab - \mu \rvert$
`Alg.2` (K + H + E)	0.20	$1.0785\ 10^{-12}$
Alg. [2]	0.18	$3.4106\ 10^{-13}$
Alg. [1]	0.31	$4.5702\ 10^{-12}$

Table 9 Example 3, $\mu = 12500 + 10000\,i$

Algorithm	Time (s)	$\lvert b^T Ab - \mu \rvert$
`Alg.2` (K + H + E)	0.20	$2.2801\ 10^{-12}$
Alg. [2]	0.25	$5.7001\ 10^{-13}$
Alg. [1]	0.38	$4.3884\ 10^{-12}$

The third example is a variant of an example from [2]. The matrix of order 200 is constructed with the Fiedler and the Moler matrices, F and M. Then, let $B = F + iM$. Finally the matrix A is (in Matlab notation)

```
A=B+(-3+5i)*ones(200)-(200+500i)*eye(200).
```

The first shift, $\mu = 5000 + 10000\,i$, is well inside the field of values. The results are in Table 6. `Alg.2` works using only the eigenvectors of K and is twice as fast as the algorithms from [2] and [1]. The second shift, $\mu = 10000 + 10000\,i$, for which the results are in Table 7, is closer to the boundary. Then, the first stage of `Alg.2` fails and we use also the eigenvalues of H. If we come closer to the boundary, for $\mu = 12000 + 10000\,i$, the first two stages fail and we have to use the intersections of ellipses with the real axis. Of course, the computing time is larger, see Tables 8 and 9. Finally, the shift $\mu = 12500 + 10000\,i$ is very close to the boundary and the algorithms from [2] and [1] need more than two eigenvector computations. Then, the computing time of `Alg.2` is comparable to what is obtained with the algorithm of [2].

5 Conclusions

In this paper we have shown how to compute isotropic vectors for matrices with real or complex entries. For real matrices the algorithm we propose is quite fast, using only the eigenvectors corresponding to the extreme eigenvalues of the symmetric part. In some cases for complex matrices, our algorithm is faster than the algorithms of [2] or [1] because it uses only one computation of eigenvalues and eigenvectors of an Hermitian matrix. However, this very simple algorithm does not always work, depending on the value of the shift μ. In this case, we use the Hermitian part of A. This needs a second eigenvector

computation. If this also fails, we resort to a combination of the eigenvectors of K and H. In this case the computing times of the three algorithms are almost the same.

Acknowledgements The writing of this paper was started in 2010 during a visit to the Nečas Center of Charles University in Prague supported by a grant Jindrich Nečas Center for mathematical modeling, project LC06052 financed by MSMT. The author thanks particularly Zdeněk Strakoš and Miroslav Rozložník for their kind hospitality.

The author thanks the referee for interesting comments.

References

1. Carden, R.: A simple algorithm for the inverse field of values problem. Inverse Probl. **25**, 1–9 (2009)
2. Chorianopoulos, C., Psarrakos, P., Uhlig, F.: A method for the inverse numerical range problem. Electron. J. Linear Algebra **20**, 198–206 (2010)
3. Ciblak, N., Lipkin, H.: Orthonormal isotropic vector bases. In: Proceedings of DETC'98, 1998 ASME Design Engineering Technical Conferences (1998)
4. Eiermann, M.: Field of values and iterative methods. Linear Algebra Appl. **180**, 167–197 (1993)
5. Fischer, B., Ramage, A., Silvester, D.J., Wathen, A.J.: On parameter choice and iterative convergence for stabilised discretisations of advection-diffusion problems. Comput. Methods Appl. Mech. Eng. **179**, 179–195 (1999)
6. Horn, R.A., Johnson, C.R.: Topics in Matrix Analysis. Cambridge University Press, Cambridge (1991)
7. Johnson, C.R.: Numerical determination of the field of values of a complex matrix. SIAM J. Numer. Anal. **15**, 595–602 (1978)
8. Liesen, J., Strakoš, Z.: GMRES convergence analysis for a convection–diffusion model problem. SIAM J. Sci. Comput. **26**(6), 1989–2009 (2005)
9. Saad, Y., Schultz, M.H.: GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Statist. Comput. **7**(3), 856–869 (1986)
10. Uhlig, F.: An inverse field of values problem. Inverse Problems **24**, 1–19 (2008)