Revisiting the inverse field of values problem

N. Bebiano[*], J. da Providência[†] A. Nata[‡], and J.P. da Providência[§]

November 24, 2013

Abstract. The field of values of a linear operator is the convex set in the complex plane comprising all Rayleigh quotients. For a given complex matrix, Uhlig proposed the inverse field of values problem: given a point inside the field of values determine a unit vector for which this point is the corresponding Rayleigh quotient. In the present note we propose an alternative method of solution to those that have appeared in the literature. Our approach builds on the fact that the field of values can be seen as a union of ellipses under a compression to the bidimensional case, in which case the problem has an exact solution. Refining an idea of Marcus and Pesce, we provide alternative algorithms to plot the field of values of a general complex matrix, that perform faster and more accurately than the existing ones.

AMS classification: 15A60; 47B35

Keywords: Field of values; Numerical range; Inverse problem; Generating vector; Compression

1 Introduction

Let $(\mathcal{H}, \langle\cdot,\cdot\rangle)$ be a Hilbert space and $\mathcal{B}(\mathcal{H})$ denote the set of bounded linear operators $\mathcal{H} \to \mathcal{H}$. The *field of values* (also known as the *numerical range*) of a linear operator $T : \mathcal{H} \to \mathcal{H}$ is the set of complex numbers

$$F(T) = \left\{ \frac{\langle T\,w, w\rangle}{\langle w, w\rangle} : w \in \mathcal{H},\ \ \langle w, w\rangle \neq 0 \right\}.$$

Thus, the field of values is the set of all Rayleigh quotients of T. The numerical range is an useful tool in the study of matrices and operators (e.g., see [5] and [7] and references therein), that has applications in stability analysis of dynamical systems and convergence theory of matrix iterations, among others.

[*]CMUC, University of Coimbra, Department of Mathematics, P 3001-454 Coimbra, Portugal (bebiano@mat.uc.pt)

[†]University of Coimbra, Department of Physics, P 3004-516 Coimbra, Portugal (providencia@teor.fis.uc.pt)

[‡]CMUC, Polytechnic Institute of Tomar, Department of Mathematics, P 2300-313 Tomar, Portugal (anata@ipt.pt)

[§]Depatamento de Física, Univ. of Beira Interior, P-6201-001 Covilhã, Portugal (joaodaprovidencia@daad-alumni.de)

The field of values is a convex and compact set. The compactness follows readily from the fact that $F(T)$ is the image of the unit compact sphere in $\mathcal{H}$ under the continuous mapping $x \to \langle Tx, x\rangle$. The convexity of $F(T)$ was stated by Toeplitz [11] and Hausdorff [6] in the first decades of the last century.

Given a point $z \in F(T)$, the *inverse field of values problem* seeks to determine a unit vector $w_z \in \mathcal{H}$ such that $z = \langle Tw_z, w_z\rangle$, referred to as a *Ritz value*. Such a vector is called a *generating vector* for z. The inverse field of values problem was firstly proposed by Uhlig in [12], motivated by the importance of the condition $0 \in F(T)$ on the stability of continuous or discrete systems $\dot{x} = Tx$ or $x_{k+1} = Tx_k$. The problem searches for the solution of the two coupled-equations

$$w_z^* T w_z = z, \;\; w_z^* w_z = 1,$$

for $w_z \in \mathcal{H}$. So this is an algebraic problem consisting on a system of two complex quadratic equations in the complex components of w_z. Finding an algebraic solution can be performed with computer algebra systems such as MATHEMATICA, but this works only for moderate dimensions. Also an analytic approach using the Lagrange multipliers formalism makes sense, however this is only feasible for low dimensions. We are interested in finding solution vectors for larger dimensions than algebraic or analytic methods can provide, such as n of the order of hundreds or thousands.

Following Uhlig, we shall use the acronym "FOV" for "field of values". The inverse FOV (iFOV) problem deserved the attention of some authors, e.g., Carden [2], Meurant [10] and different methods of solution have been proposed. A new method was given in [3] that is simpler and faster than the existing ones, and provides accurate numerical results where the previous ones often fail, namely for very close points to the field of values boundary. In these algorithms, most of the computing time is spent computing eigenvalues and eigenvectors of the Hermitian part of the matrix, and in [2] and [3] the minimum number of eigenanalyzes is two.

This paper has two main goals. The first one is to provide algorithms to plot quickly and accurately the field of values. The second goal is to revisit the iFOV problem, and to propose an alternative and simpler approach. Our method is conceptually straightforward, since it is essentially based on the reduction to the 2×2 case. It requires a small number of eigenanalyzes, sometimes only one, and compares well in execution time and in error with the existing ones in the literature.

This paper is organized as follows. In Section 2, the main ideas used in our method are presented. In Section 3, two alternative algorithms for plotting the field of values of a general complex matrix are proposed. We emphasize that these algorithms play a crucial role in the solution of the iFOV problem, and remarkably improve the approximation of the FOV's boundary. Section 4 provides an overview of the approaches to the iFOV problem. In Section 5, this problem is solved for a matrix of

arbitrary size by a reduction to the 2-dimensional case. In Section 6, some numerical examples that illustrate the theory are given, and the different approaches are compared. The images here included were numerically computed using MATLAB 7.8.0.347 (R2009a).

2 Main ideas

2.1 General properties of F

From now on we shall consider $\mathcal{H} = \mathbb{C}^n$. Let M_n be the algebra of $n \times n$ complex matrices. Writing the *cartesian decomposition* of $T \in M_n$, that is, $T = H(T) + i\,K(T)$, where

$$H(T) = \frac{T + T^*}{2} \quad \text{and} \quad K(T) = \frac{T - T^*}{2i},$$

are Hermitian matrices, we can easily conclude that $F(H(T))$ is the orthogonal projection of $F(T)$ on the real axis and $F(K(T))$ is the orthogonal projection of $F(T)$ on the imaginary axis.

We recall that a *supporting line* of a convex set $S \subset \mathbb{C}$ is a line containing a boundary point of S and defining two half planes, such that one of them does not contain S. For $\theta \in [0, 2\pi)$, if we consider a supporting line of $F(T)$ perpendicular to the direction of slope θ, the orthogonal projection of the numerical range on this direction is given by $F(H(\mathrm{e}^{-i\theta}T))$.

The following well known properties of F are directly related with the subject of this note. For their proofs we refer the interested reader to [7].

(1) $\sigma(T) \subset F(T)$, where $\sigma(T)$ denotes the spectrum (set of eigenvalues) of T.

(2) For any unitary matrix U, $F(U^*TU) = F(T)$.

(3) For any complex number z, $F(T + zI) = F(T) + z$.

(4) $F(T)$ is a real line segment if and only if T is Hermitian.

(5) If T is normal, then $F(T) = \mathrm{Co}\{\sigma(T)\}$, where $\mathrm{Co}\{\cdot\}$ denotes the convex hull of $\{.\}$.

(6) Let x_θ be a unit eigenvector associated with the largest, or with the smallest, eigenvalue of $\mathrm{H}(\mathrm{e}^{-i\theta}T)$. Then, $z_\theta = x_\theta^* T x_\theta \in \partial F(T)$, the boundary of $F(T)$. Furthermore, $z_\theta \in L_\theta \cap F(T)$ where L_θ is the tangent line at z_θ.

(7) The boundary of $F(T)$ is a piecewise algebraic curve, and each of its non-differentiable boundary points is an eigenvalue of T.

2.2 F as the union of elliptical discs

For P a 2-dimensional orthogonal projection, the restriction of PTP to the range of P is called a 2-dimensional *compression* of T. For completeness, we recall the following important result, usually known as the *Elliptical Range Theorem* (for a proof see, e.g., [7] or [13]).

Theorem 2.1 *Let $T \in M_2$. Then $F(T)$ is a (possibly degenerate) closed elliptical disc, whose foci are the eigenvalues of T, λ_1 and λ_2. The Cartesian equation of the boundary of this elliptical disk is*

$$\frac{X^2}{M^2} + \frac{Y^2}{N^2} = \frac{1}{4}$$

where

$$X = (x - \mathrm{Re}(c)) \cos\gamma + (y - \mathrm{Im}(c)) \sin\ \gamma,\ \ Y = (x - \mathrm{Re}(c)) \sin\gamma + (y - \mathrm{Im}(c)) \cos\gamma,$$

$c = (\lambda_1 + \lambda_2)/2$ is the center of the ellipse, and γ is the slope of the line segment with endpoints λ_1 and λ_2. The lengths of the major and minor axis of the ellipse are

$$M = \sqrt{\mathrm{Tr}\ (T^*T) - 2\mathrm{H}\left(\overline{\lambda_1}\lambda_2\right)},\ \ N = \sqrt{\mathrm{Tr}\ (T^*T) - |\lambda_1|^2 - |\lambda_2|^2},$$

respectively.

The following result is fundamental in our approach to the iFOV problem.

Theorem 2.2 *(Marcus and Pesce [9]) For any $T \in M_n$,*

$$F(T) = \bigcup_{\tilde{u},\tilde{v}} F(T_{\tilde{u}\tilde{v}}),$$

where

$$T_{uv} = \begin{bmatrix} \langle T\tilde{u}, \tilde{u}\rangle & \langle T\tilde{v}, \tilde{u}\rangle \\ \langle T\tilde{u}, \tilde{v}\rangle & \langle T\tilde{v}, \tilde{v}\rangle \end{bmatrix}, \tag{1}$$

$\tilde{u}$ and $\tilde{v}$ varying over all pairs of real orthonormal vectors.

The Marcus-Pesce Theorem is applicable to the problem of devising an effective procedure for constructing the field of values of an arbitrary $n \times n$ complex matrix. The idea is to generate a reasonable complete set of orthonormal pairs $\tilde{u}, \tilde{v}$, not necessarily real, and actually compute the union of the elliptical discs $F(T_{\tilde{u}\tilde{v}})$. This is the so-called *inside-out* approach, an alternative to the *outside-in* approach (e.g., [8]), which consists of determining the smallest and the largest eigenvalues of $\mathrm{H}\left(\mathrm{e}^{-i\theta}T\right)$ for various values of $\theta \in [0, 2\pi)$. These eigenvalues provide sharp approximations for the projection of the field of values on the direction θ. The collection of these sharp bounds gives an outer polygonal approximation to $F(T)$, whose sides are tangent to $\partial F(T)$ (see Property (6) of Subsection 2.1). This procedure may be extended to the infinite dimensional setting, as well as for linear unbounded operators.

3 Algorithms for plotting F of a general complex matrix

The main purpose of this section is to provide two alternative algorithms for numerically determining $F(T)$ within some prescribed tolerance tol. These algorithms are used in our solution method for the iFOV problem. In [9], a MS-BASIC program for plotting $F(T)$ was presented based on formula (1), in the case T is a real matrix. Our algorithms improve Marcus-Pesce algorithm, as they work efficiently for complex matrices and large dimensions. Instead of using randomly generated vectors u and v, we make use of suitably chosen vectors u and v which generate boundary points of the numerical range. Throughout, we denote by $\text{span}\{u, v\}$ the subspace generated by the linearly independent vectors u, v, and by $\tilde{u}, \tilde{v}$, two orthonormal vectors in $\text{span}\{u, v\}$. The respective MATLAB programs are available at the following website:

http://www.mat.uc.pt/∼bebiano

3.1 Algorithm A

I. Check if T is Hermitian. If so, compute its smallest and largest eigenvalues and the line joining them is $F(T)$. If not, proceed to the next step.

II. Check T for normality ($TT^* = T^*T$). If T is normal, compute its eigenvalues and their convex hull is $F(T)$. If T is not normal, continue.

III. Set $\theta_k = (k-1)\pi/m$, $k = 1, \ldots, m$ for some positive integer $m \geq 3$.

IV. Starting with $k = 1$, up to $k = m$, take the following steps:

(i) Compute eigenvectors u_k and v_k associated, respectively, to the largest and smallest eigenvalue of the Hermitian matrix $\mathrm{H}\left(\mathrm{e}^{-i\theta_k}T\right)$.

(ii) Compute the compression of T to $\text{span}\{u_k, v_k\}$, $T_{\tilde{u}_k\tilde{v}_k}$.

(iii) Compute the boundary Γ_k of $F(T_{\tilde{u}_k\tilde{v}_k})$.

(iv) If $k < m$, take the next k value and return to (i).

V. Plot the convex-hull of the collection of curves $\Gamma_1, \ldots, \Gamma_m$ as an approximation for $F(T)$.

Algorithm A may not be efficient in the case T is unitarily similar to a direct sum of matrices, because in this case the field of values of the compressed matrices may degenerate into line segments.In this event, the next modified algorithm, which essentially differs in the choice of generating vectors for boundary points, is more convenient.

3.2 Algorithm B

Steps **I**, **II** and **III** are as in algorithm A.

IV. Compute eigenvectors u_1 and v_1 associated, respectively, to the largest and smallest eigenvalue of $\mathrm{H}\left(\mathrm{e}^{-i\theta_1}T\right)$.

V. Starting with $k = 2$ and up to $k = m$, take the following steps:

(i) Compute eigenvectors u_k and v_k associated, respectively, to the largest and smallest eigenvalue of $\mathrm{H}\left(\mathrm{e}^{-i\theta_k}T\right)$.

(ii) Compute the compressions of T to $\mathrm{span}\{u_k, u_{k-1}\}$ and $\mathrm{span}\{v_k, v_{k-1}\}$, $T_{\tilde{u}_k\tilde{u}_{k-1}}$ and $T_{\tilde{v}_k\tilde{v}_{k-1}}$, respectively.

(iii) Compute and draw the boundaries of $F(T_{\tilde{u}_k\tilde{u}_{k-1}})$ and $F(T_{\tilde{v}_k\tilde{v}_{k-1}})$, Γ_k and Λ_k, respectively.

(iv) If $k < m$, take next k value and return to (i). Otherwise, continue.

VI. Take the following steps.

(i) Compute the compressions of T to $\mathrm{span}\{u_1, v_m\}$ and $\mathrm{span}\{v_1, u_m\}$, $T_{\tilde{u}_1\tilde{v}_m}$ and $T_{\tilde{v}_1\tilde{u}_m}$, respectively.

(ii) Compute the boundaries of $F(T_{\tilde{v}_1\tilde{u}_m})$ and $F(T_{\tilde{u}_1\tilde{v}_m})$, Γ_1 and Λ_1 respectively.

VII. Take the convex-hull of the collection of curves $\Gamma_1, \ldots, \Gamma_m, \Lambda_1, \ldots, \Lambda_m$ as an approximation for $F(T)$.

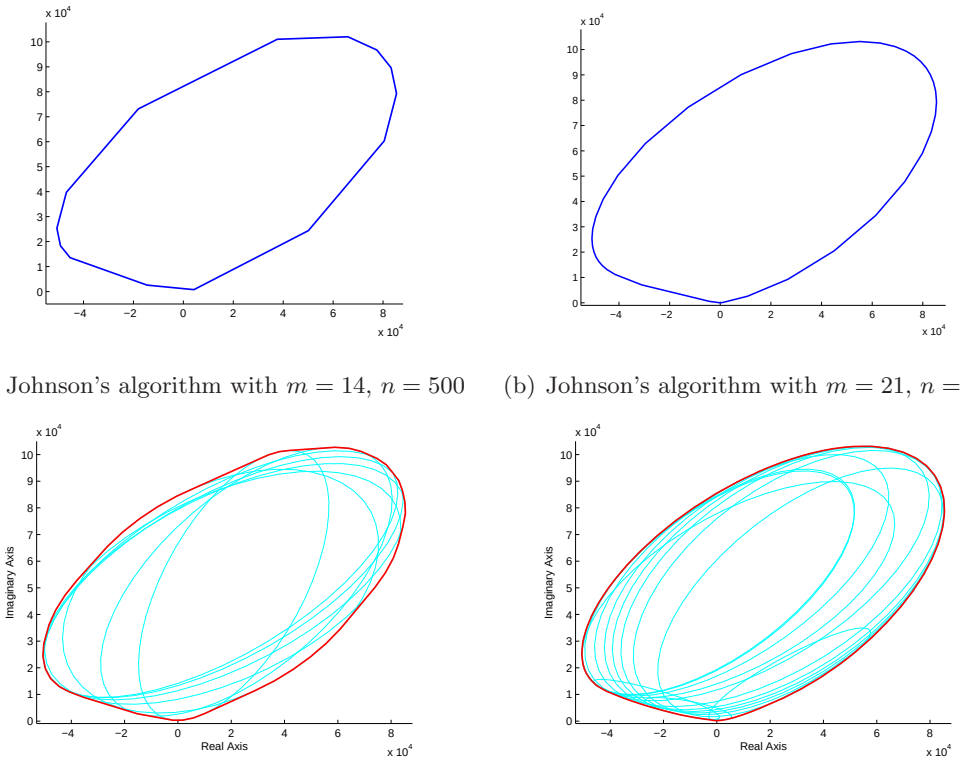
3.3 Accuracy and examples

Algorithm B provides a much better accuracy than algorithm A in the approximation of $F(T)$. Both algorithms A and B behave especially well when compared with the "outside-in" approach, which merely provides a polygonal approximation of $F(T)$, and so it requires a much finer mesh to reach convenient accuracy. The amount of effort necessary to plot $F(T)$ by algorithms A and B to a desired accuracy depends essentially on the shape. For a given difficulty in shape, the effort is of the same order as that of solving the Hermitian eigenvalue-eigenvector problem necessary to generate a boundary point. Thus, the necessary effort is $\mathcal{O}(n^3)$.

The same arguments in [8] suggest that the accuracy of algorithm A increases with m as m^2. Since in algorithm B, the boundary of F is approximated by arcs of ellipses which are tangent to the boundary at two of its points consecutively determined, one expects that the respective accuracy increases with m as m^4. In Johnson's method the boundary is piecewise approximated by line segments,

which depend linearly on some convenient coordinate, so the error involved is of the order of $1/m^2$. In algorithm B, the boundary is piecewise approximated by arcs which have well defined slopes at the end points. These arcs must be at least cubic curves, so the error involved is of the order of $1/m^4$. Since the boundary points produced by a given mesh distribute themselves on the boundary with a density which is inversely proportional to the local radius of curvature of the boundary, it follows that the local accuracy is also inversely proportional to the radius of curvature. It is in order to observe that, with respect to computational effort, the most convenient interpolation between two successive computed points on the boundary of $F(T)$ is by the unique cubic which has the right slopes at the interpolated points.

In a recent paper, Uhlig [14] addresses the accurate computation of the boundary of $F(T)$ by a method which is similar to algorithm B. His careful numerical analysis supports the expected fast convergence of this algorithm.



(a) Johnson's algorithm with $m = 14$, $n = 500$

(b) Johnson's algorithm with $m = 21$, $n = 500$

(c) Agorithm A with $m = 7$, $n = 500$

(d) Algorithm B with $m = 7$, $n = 500$

Figure 1: Comparing the performance of Johnson's algorithm, algorithm A and algorithm B.

Example 3.1 *In Figure 1, the* 500×500 *complex matrix A in [3, p. 202] is considered. This matrix is constructed from the complex matrix* $B = F + iM$*, where* F *is the Fiedler matrix* $F = (|i - j|)$ *and*

	m	area	acc. digits	seconds
Johnson's algorithm	7	8.6260×10^9	0	5.082322
	14	9.0890×10^9	1	7.229055
	28	9.2140×10^9	2	14.252341
	56	9.2439×10^9	2	24.169221
	112	9.2513×10^9	3	47.456900
Agorithm A	7	9.1064×10^9	1	3.363152
	14	9.2060×10^9	2	6.152282
	28	9.2383×10^9	2	11.847686
	56	9.2487×10^9	2	23.233351
	112	9.2514×10^9	3	46.187566
Agorithm B	7	9.2406×10^9	2	3.424524
	14	9.2509×10^9	3	6.330416
	28	9.2528×10^9	3	12.108766
	56	9.2533×10^9	4	23.853617
	112	9.2534×10^9	4	47.086406

Table 1: Performance of Johnson's algorithm, algorithm A and algorithm B for the 500×500 matrix of the Example 3.1.

M is the Moler matrix $M = U^T U$ for the upper triangular matrix U with $u_{i,j} = -1$ for $j > i$. By adding $3 + 5i$ multiplied by the 500×500 matrix whose entries are ones, we obtain the matrix A, or in MATLAB notation $A = B + (-3 + 5i)ones(500)$. The execution time of the different algorithms for this matrix is compared in Table 1. The subroutine fv.m *of the MTLAB "Matrix Computation Toolbox" has been used to implement Johnson's algorithm.*

The accuracy of algorithms A and B may be improved by choosing a finer mesh.

4 The iFOV problem: the state of the art

Uhlig [12] provided a complicated solution for the inverse field of values problem, that initially generates points that surround the given point z, by using the fact that points on the boundary of $F(T)$ and their generating vectors can be computed by Johnson's method [8]. Uhlig's method proceeds with a randomized approach to surround z tighter and tighter, by successive attempts, and iteratively refines the generating vector approximation.

In the continuation, Carden [2] pointed out the connection between the iFOV problem and iterative eigensolvers, and presented a simpler method of solution based on the following procedure:

(1) Solution of the 2×2 case by reduction to a form that is useful for proving the FOV convexity.

(2) Using Property (6), the given point z is surrounded by points on the FOV boundary and associated generating vectors are determined.

(3) From the triangle defined by three points on the FOV boundary that surround z and convexity, generating vectors for the endpoints of a line containing z are determined. By compression to the subspace spanned by these vectors, iFOV is solved by reduction to the 2×2 case.

A more efficient method for the iFOV problem than either those in [12] or [2], has been presented in [3] along the following lines:

(1) Direct solution of the 2×2 case.

(2) Generation of points on the FOV boundary according to Property (6). Surrounding of z by ellipses determined by compressions defined by the generating vectors of all possible pairs of vertices of the inner approximation. For each ellipse, its intersection with the real axis is determined. If in any ellipse there are points on both sides of z, then corresponding generating vectors are used to reduce to the 2×2 case and to solve the iFOV problem.

(3) Bisection on the angles to refine inner approximation in the direction of z.

A relevant aspect of [2] and [3] is surrounding the desired point by four points on the FOV boundary, determined by considering the largest and the smallest eigenvalues of the Hermitian matrices $\mathrm{H}\left(\mathrm{e}^{-i\theta'}T\right)$ and $\mathrm{H}\left(\mathrm{e}^{-i\theta''}T\right)$, for $\theta'' - \theta' = \pi/2$. In our approach, boundary points similarly determined are also considered, which, however may not surround the desired point, it only being required that this point belongs to one of the elliptical discs. Moreover, we find it advantageous to relax the restriction $\theta'' - \theta' = \pi/2$.

4.1 Analytic solution of the iFOV problem in the 2×2 case

Following Carden [2] given $T \in M_2$, and $z \in \mathbb{C}$, we exactly determine a unit vector $w_z \in \mathbb{C}^2$ such that $z = w_z^* T w_z$. Without loss of generality, we may assume that the trace of T is zero. (If the trace was nonzero, we would subtract $\mathrm{Tr}\,(T/2)$ from T and from z.) Under this assumption, the eigenvalues sum to zero, and may be denoted as λ and $-\lambda$. Let us assume that the eigenvalues of T are non-zero. Further, we may assume that the eigenvalues of T are real. (To accomplish this, we need to multiply both T and z by $\mathrm{e}^{-i\psi}$ where $\lambda = a\mathrm{e}^{i\psi},\ a > 0$.) A unitary matrix U can be found such that T is carried out into the Schur form

$$U^*TU = \hat{T} = \begin{pmatrix} a & c \\ 0 & -a \end{pmatrix},$$

where c is real. Since the field of values is invariant under unitary similarity transformations, z need not to be altered. Having in mind that any 2×2 matrix can be shifted, scaled, and unitarily transformed into this form, we solve the inverse problem for $\hat{T}$.

Let $z = x + iy \in F(\hat{T})$. Without loss of generality, we may consider the unit vector $w_z \in \mathbb{C}^2$ as $w_z = (\cos u, e^{i\phi} \sin u)^T$. We find

$$w_z^* \hat{T} w_z = a \cos 2u + \frac{c}{2} \sin 2u \ \cos\phi + i \frac{c}{2} \sin 2u \ \sin\phi,$$

which easily gives

$$\cos\ 2u = \frac{4ax \pm \sqrt{c^4 + 4a^2c^2 - 4c^2x^2 - (4c^2 + 16a^2)y^2}}{4a^2 + c^2}. \tag{2}$$

This relation determines u, and the relation

$$\sin\ \phi = \frac{2y}{c \sin 2u} \tag{3}$$

determines ϕ. Hence, given $z \in F(\hat{T})$ the u and ϕ that specify the generating vector w_z must satisfy the above relations, and in terms of the matrix T the generating vector is Uw_z.

If $c^4 + 4a^2c^2 - 4c^2x^2 - (4c^2 + 16a^2)y^2 \geq 0$ (the inequality corresponding to $x + iy$ being inside the ellipse), it is always possible to determine w_z such that $x + iy = w_z^* T w_z / w_z^* w_z$, and conversely. The sought after solution is not necessarily unique [2], because for $x + iy$ in the interior of $F(T)$, there exist two linearly independent vectors determined by u and $u - \pi/4$, for $u \neq \pi/4$. If $u = \pi/4$, then w_z is a point on the boundary, and the generating vector is unique provided that the ellipse is nondegenerate. If $c = 0$, then $F(T)$ is the line segment defined by the two eigenvalues, and the solution is unique only for the end points. If $T = 0$, then $F(T)$ is a singleton and any unit vector in $\mathbb{C}^2$ is a generating vector.

The above ideas are now used to develop an algorithm to solve the iFOV in the 2×2 case, which will be crucial in solving the general case. Let $\tilde{u}, \tilde{v}$ be two orthonormal vectors belonging to $\text{span}\{u, v\}$. Given the 2-dimensional compression $T_{\tilde{u}\tilde{v}}$ of T to $\text{span}\{u, v\}$ (cf. (1)) and for $z \in F(T_{\tilde{u}\tilde{v}})$, take the following steps:

I. Determine the eigenvalues of $T_{\tilde{u}\tilde{v}}$, λ_1, λ_2. Construct a unitary matrix U that takes $T_{\tilde{u}\tilde{v}}$ to the Schur form

$$T_{\tilde{u}\tilde{v}} = U T_{\tilde{u}\tilde{v}}^{(0)} U^* = U \begin{bmatrix} \lambda_1 & \mathrm{e}^{i\theta} d \\ 0 & \lambda_2 \end{bmatrix} U^*, \quad d \geq 0, \quad \theta = \arg(\lambda_1 - \lambda_2).$$

II. Let $z_0 = (\lambda_1 + \lambda_2)/2$, and $a = |\lambda_1 - \lambda_2|/2$. Thus,

$$T_{\tilde{u}\tilde{v}}^{(00)} = \begin{bmatrix} a & d \\ 0 & -a \end{bmatrix} = \mathrm{e}^{-i\theta}(T_{\tilde{u}\tilde{v}}^{(0)} - z_0 I_2),$$

and $\omega_0 = \mathrm{e}^{-i\theta}(z - z_0) \in F(T_{\tilde{u}\tilde{v}}^{(00)})$.

III. A generating vector $\zeta^{(0)} = (\zeta_1^{(0)}, \zeta_2^{(0)})^T$ of ω_0 is easily found using (2) and (3). Hence $\zeta = (\zeta_1, \zeta_2)^T = U\zeta^{(0)}$ is a generating vector of z.

5 Algorithms for the inverse FOV problem

Given an $n \times n$ complex matrix T and $z = x + iy \in \mathbb{C}$, let $\epsilon > 0$ denote some tolerance (for example, $\epsilon = 10^{-16}||T||$ for a double precision computation). Our approach to the iFOV problem is based on Marcus-Pesce Theorem, having in mind that we can exactly generate any point in the interior of the ellipses, as well as in its boundary, as described in Section 4.1.

5.1 Algorithm A′

I. Discretization of the interval $[\mathbf{0}, \pi]$: Choose a m-mesh $\theta_k = \dfrac{\pi(k-1)}{m}$, $k = 1, \ldots, m$, for some positive integer $m \geq 2$.

II. For $k \in \{1, \ldots, m\}$, starting with $k = 1$, take the following sub-steps.

(i) Construct the matrix $T_k = \mathrm{H}\left(\mathrm{e}^{-i\theta_k}T\right)$, and compute its largest and smallest eigenvalues. Determine a pair of associated eigenvectors u_k and v_k. In each step this is a well defined Hermitian eigenvalue-eigenvector problem.

(ii) If $|\lambda_{\min}(T_k) - \lambda_{\max}(T_k)| < \epsilon$, then $F(T)$ is approximately a line segment, so that T is approximately either Hermitian or skew-Hermitian (or a complex shift of one of these). If the line segment degenerates into a point, then T is a scalar matrix. In either of these cases, it can be easily decided if z belongs to $F(T)$, and if so, a generating vector may be determined. Otherwise, continue.

(iii) Check whether the given point $z = x + iy$ belongs to the intersection of the following half-planes

$$x\cos\,\theta_k + y\sin\,\theta_k \leq \lambda_{\max}(T_k)$$

$$x\cos\,\theta_k + y\sin\,\theta_k \geq \lambda_{\min}(T_k).$$

If z is not inside the above half-planes intersection, then $z \notin F(T)$. Otherwise, continue.

(iv) Take the compression $T_{u_k v_k}$ of T to span$\{u_k, v_k\}$. If $z \in F(T_{u_k v_k})$, let $T_{\tilde{u}\tilde{v}} = T_{u_k v_k}$ and continue to III. Otherwise, take the next value of k and go to (i).

III. The generating vector of $z \in F(T)$ is given by $w_z = \tilde{u}\zeta_1 + \tilde{v}\zeta_2$, where $\tilde{u}, \tilde{v}$ are orthonormal vectors such that (1) holds, and ζ_1, ζ_2 are as defined in **III** Section 4.1.

Next we introduce a slight modification to algorithm A' changing the form of generating the compressions. This modification allows the treatment of some deficiencies revealed by algorithm A' with the degeneracy of the elliptical discs into line segments.

5.2 Algorithm B′

I. As Step I of algorithm A′.

II. As sub-steps (i), (ii), and (iii) of Step II with $k = 1$ of algorithm A′.

III. For $k \in \{2, \ldots, m\}$, starting with $k = 2$, take the sub-steps (i), (ii) and (iii) of Step II of algorithm A′.

(iv) Take the compression T_{u_{k-1},u_k} of T to span$\{u_{k-1}, u_k\}$. If $z \in F(T_{u_{k-1},u_k})$, let $T_{\tilde{u}\tilde{v}} = T_{\tilde{u}_{k-1},\tilde{u}_k}$ and continue to IV. Otherwise, take the compression T_{v_{k-1},v_k} of T to span$\{v_{k-1}, v_k\}$. If $z \in F(T_{v_{k-1},v_k})$, let $T_{\tilde{u}\tilde{v}} = T_{\tilde{v}_{k-1},\tilde{v}_k}$ and continue to IV. Otherwise and if $k < m$, take the next value of k and go to (i) of this step.

IV. Compute the compressions of T to span$\{u_1, v_m\}$ and span$\{v_1, u_m\}$, $T_{\tilde{u}_1\tilde{v}_m}$ and $T_{\tilde{v}_1\tilde{u}_m}$, respectively.

V. As Step III of algorithm A′.

Algorithms A' and B' may be refined by replacing the interval $[0, \pi]$ by a smaller one around the perpendicular direction to the slope of the boundary at the closest point to the point under investigation, and by choosing a fine mesh in that interval.

6 Discussion and examples

We observe that algorithms A′ and B′ are particular forms of a super-algorithm in which the compression of T to the space spanned by eigenvectors associated with the largest and smallest eigenvalues of H $\left(e^{-i\theta_k}T\right)$ and H $\left(e^{-i(\theta_k-\alpha)}T\right)$ are considered for some convenient α. In algorithm A′, we have $\alpha = \pi$, and in algorithm B′, $\alpha = \theta_2$, as the largest eigenvalue of H $\left(e^{-i\theta_k}T\right)$ is the smallest

eigenvalue of $\mathrm{H}\left(\mathrm{e}^{-i(\theta_k-\pi)}T\right)$ and $\theta_{k+1}=\theta_k+\theta_2$. However, other values of α may be more appropriate according with the particular case taken into consideration.

Our algorithms are deterministic, in the sense that they determine a solution provided the considered discretization of the interval $[0,\pi]$ is fine enough. The lack of determinism occurs if the boundary of $F(T)$ contains line segments (also called *flat portions*), which occurs when the extreme eigenvalues of $\mathrm{H}(\mathrm{e}^{-i\theta_k}T)$ for a certain k have multiplicities greater than one. For a given point in $F(T)$ the algorithms will always determine a generating vector which may only depend on the chosen mesh.

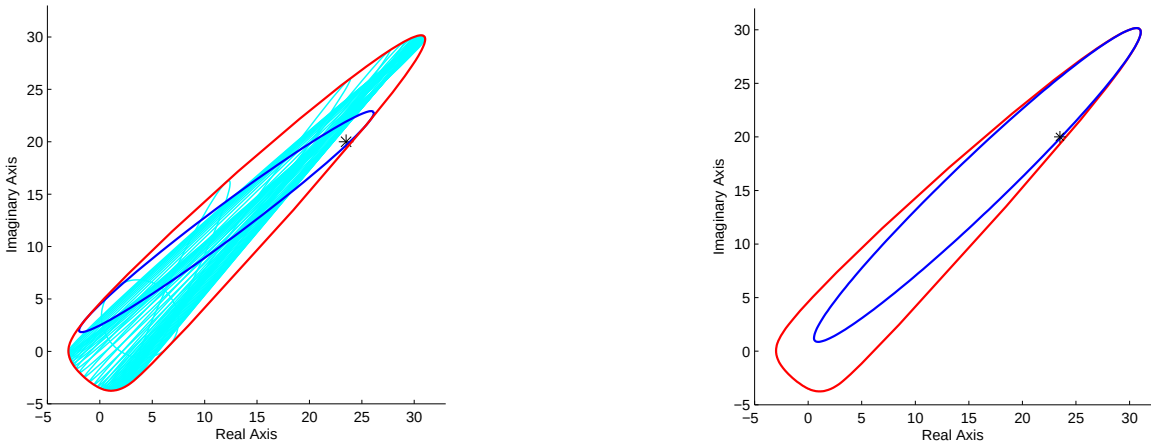
	algorithm	m	seconds	eigenanalyses	error
$\mu=23.5+i20$	`invnrp` from [3]	–	0.027332	11	3.5527×10^{-15}
	A'	41	0.043476	33	7.9441×10^{-15}
	B'	2	0.036121	2	1.4648×10^{-14}
$\mu=23.7+i20$	`invnrp` from [3]	–	0.026650	12	3.5527×10^{-15}
	A'	41	0.042243	33	1.0658×10^{-14}
	B'	5	0.050956	5	5.0243×10^{-15}
$\mu=23.7289639701427+i20$	`invnrp` from [3]	–	0.026945	12	7.1054×10^{-15}
	A'	41	0.043584	33	7.1054×10^{-15}
	B'	5	0.050975	5	3.5527×10^{-15}

Table 2: Performance of algorithms A', B' and `invnrp` from [3], for Example 6.1.

Example 6.1 *Our first example features a* 10×10 *matrix generated by the MATLAB command* $T=randn(10)+(3+3i)ones(10)$. *The obtained results are summarized in Table 1 and in Figure 2, illustrating the advantage of algorithm* B', *which requires a smaller value of* m *to solve the problem, than algorithm* A' *does, without implementation of the refinement procedure.*

Example 6.2 *The* 45×45 *complex matrix whose field of values is depicted in Figure 3 is defined as* $A=B+(-3+5i)ones(45)$, *where* $B=F+iM$ *(cf. Example 3.1). Our results, for the test point* $\mu=-200+500i$, *are summarized in Table 2 and compared with those for the algorithms of [2] and [3]. In Figure 3, the advantage of algorithms* A' *and* B', *which solve the problem with a smaller number of eigen-analyzes as compared with the approaches in [2] and [3], is illustrated.*

We have also replaced the interval $[0,\pi]$ *of Step 1 in algorithm* B' *by* $[\alpha,\alpha+\pi]$, *for* $\alpha=\pm\pi/4,\pm\pi/2,\pm3\pi/4$. *We found that the performance of the algorithm is only slightly affected by this replacement, confirming that the good quality of the performance is not accidental.*



(a) Algorithm A′ (b) Algorithm B′

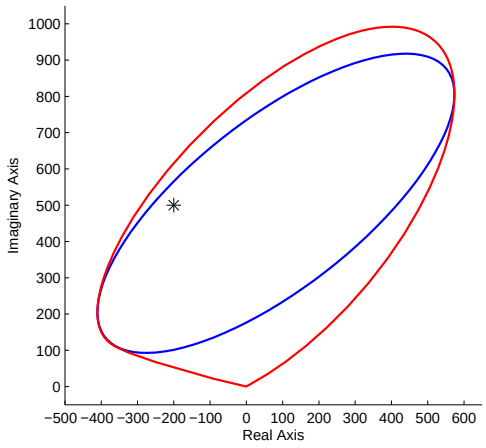
Figure 2: Solution of the iFOV problem for the matrix $A = \text{randn}(10) + (3 + 3i)\text{ones}(10)$ of Example 6.1, using algorithms A′ and B′, being $\mu = 23.5 + 20i$ the point indicated.

algorithm	m	seconds	eigenanalyses	error
`inversefov` from [2]	–	0.204922	3	2.2737×10^{-13}
`invnrp` from [3]	–	0.028962	3	1.6078×10^{-13}
A'	2	0.028287	1	1.1369×10^{-13}
B'	2	0.037529	2	3.4224×10^{-13}

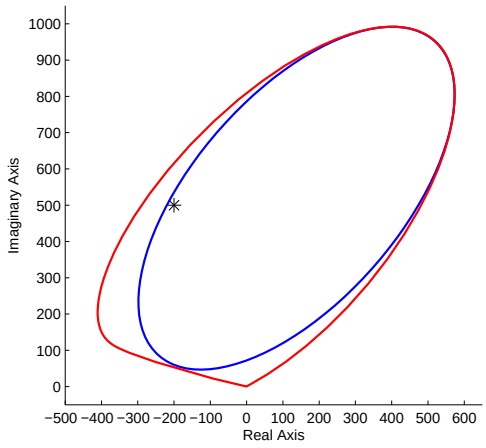
Table 3: Performance of algorithms A', B', inversefov [2] and invnrp [3], for Example 6.2.

Example 6.3 *Our last example considers the* 188×188 *Jordan block* J *for the eigenvalue* $1 + 3i$ *(with all co-diagonals equal to* 1*) and the test point* $\mu = 1.707 + i3.707$*, that lies inside* $F(J)$ *within* 10^{-5} *from the boundary. As it is well known, in this case* $F(J)$ *is a circular disc centered at* $1 + 3i$*. Since the fields of values of the* 2*-dimensional compressions of this high dimensional Jordan block are elliptical discs with a huge eccentricity, the solution of the iFOV problem requires a very careful consideration of the relevant range of* θ_k*, in order to find an elliptical disc which contains the point* μ*. We have considered the refinement of algorithms* A' *with* $m = 2$ *and* B' *with* $m = 3$ *and the interval* $\left[arc\tan(\frac{3.707-3}{1.707-1}), arc\tan(\frac{3.707-3}{1.707-1}) + 0.05\pi\right]$*. The results are summarized in Table 3. The performance of algorithms* A' *and* B' *compares very well with the ones of the approaches in [2] or [3].*

Our codes have been tested in a PC with an Intel Core 2 Duo T5550 (1.83 GHz) processor and 3



(a) Algorithm A′



(b) Algorithm B′

Figure 3: Solution of the iFOV problem for the matrix $A = (B + (-3 + 5i)\text{ones}(45))$ and the point $\mu = (-200 + 500i)$ considered in Example 6.2, using algorithms A' and B′.

GB random-access memory (RAM).

algorithm	m	seconds	eigenanalyses	error
`inversefov` from [2]	–	0.478268	3	2.0948×10^{-15}
`invnrp` from [3]	–	0.261283	3	4.9651×10^{-16}
A′	2	0.276974	1	1.6012×10^{-15}
B′	3	0.395581	2	2.2204×10^{-16}

Table 4: Performance of algorithms A', B', inversefov [2] and invnrp [3], for Example 6.3.

Acknowledgments

Valuable suggestions of the Editor Prof. Froilán M. Dopico and of the anonymous Referees are gratefully acknowledged. In particular, we wish to thank Prof. Dopico for bringing [14] to our attention and one of the Referees for providing a useful code.

References

[1] N. Bebiano, I. Spitkovsky, Numerical ranges of Toeplitz operators with matrix symbols, *Linear Algebra Appl.* **436** (2012) 1721-1726.

[2] R. Carden, A simple algorithm for the inverse field of values problem. *Inverse Problems* **25** (2009) 115019.

[3] C. Crorianopoulos, P. Psarrakos and F. Uhlig, A method for the inverse numerical range problem. *Linear Algebra Appl.* **24** (2010) 055019.

[4] C. Davis, The Toeplitz-Hausdorff Theorem explained. *Canad. Math. Bull.,* **14** (1971) 245-246.

[5] K.E. Gustafson and D.K.M. Rao, Numerical range, the field of values of linear operators and matrices. *Springer-Verlag, New York* (1997).

[6] F. Hausdorff, Der Wertvorrat einer Bilinearform. *Math. Z.* **3** (1919) 314-316.

[7] R.A.Horn and C.R.Johnson, Topics in matrix analysis. *Cambridge University Press, Cambridge* (1991).

[8] C.R. Johnson, Numerical determination of the field of values of a general complex matrix *SIAM J. Numer. Anal.* **15** (1978) 595-602.

[9] M. Marcus and C. Pesce, Computer generated numerical ranges and some resulting theorems. *Linear and Multilinear Algebra*, **20** (1987) 121-157.

[10] G. Meurant, The computation of isotropic vectors. *Numerical Algorithms* **60** (2012) 193-204.

[11] O.Toeplitz, Das algebraische Analogon zu einern satze von Fejr. *Math. Z.* **2** (1918) 187-197.

[12] F. Uhlig, An inverse field of values problem. *Inverse Problems* **24** (2008) 055019.

[13] F. Uhlig, The field of values of a complex matrix, an explicit descriotion of the 2×2 case. *SIAM Alg. Disc. Math.* **6** (1985) 541-545.

[14] F. Uhlig, Faster and more accurate computation of the field of values boundary for n by n matrices. *Linear and Multilinear Algebra* (2013) DOI: 10.1080/03081087.2013.779269.