The computation of isotropic vectors

Gérard MEURANT

October 2011

Given a nonsingular square matrix A and a real or complex number μ, we would like to compute (when it exists) a vector b of unit norm such that

$$b^* A b = \mu$$

This is equivalent to

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

If μ is an eigenvalue, take b as the corresponding eigenvector. If not the problem reduces to $b^* \tilde{A} b = 0$ with $\tilde{A} = A - \mu I$

Even if A is real, $\tilde{A}$ may be complex

Isotropic vectors

Vectors b such that

$$b^* A b = 0$$

are called isotropic vectors

Let

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

be the field of values of A (or the numerical range)

The existence of an isotropic vector is equivalent to $0 \in W(A)$

Computing b is known as the inverse field of values problem

Motivation

Assume that we solve the linear system $Ax = b$ with GMRES starting from $x_0 = 0$

If $b^* A^j b = 0, j = 1, \ldots, m$ GMRES stagnates for m iterations

Therefore, if b is an isotropic vector, we have
$\|r_1\| = \|r_0\| = \|b\| = 1$

Literature

F. Uhlig, *An inverse field of values problem*, Inverse Problems, v 24 (2008), pp. 1–19

R. Carden, *A simple algorithm for the inverse field of values problem*, Inverse Problems, v 25 (2009), pp. 1–9

C. Chorianopoulos, P. Psarrakos and F. Uhlig, *A method for the inverse numerical range problem*, Elec. J. Linear Alg., v 20 (2010), pp. 198–206

The algorithms proposed in these papers require (at least) two computations of the eigensystem of the symmetric (or antisymmetric) part of A

New algorithm

When A is real we propose an algorithm that needs only one eigenvector computation

When A is complex using only one eigenanalysis works in many cases, but not always. We may have to use 2 or more...

A real

We are looking for a real isotropic vector b

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

Let $H = (A + A^T)/2$ be the symmetric part of A

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

with Ω diagonal (ω_i), X matrix of the eigenvectors

If H is definite, 0 is not in the FOV and there is no solution

Using two eigenvectors

Assume H is indefinite

Let $c = X^T b$. Then

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

Let $\omega_1 < 0$ be the smallest eigenvalue. There exists $k > 1$ such that $\omega_k > 0$ (for instance $k = n$)

Let $< 0 < t < 1$ and $|c_1|^2 = t$, $|c_k|^2 = 1 - t$, $c_i = 0, \forall i \neq 1, k$
It yields

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

and

$$t_s = \frac{\omega_k}{\omega_k - \omega_1} > 0$$

$|c_1|^2 = t_s, |c_k|^2 = 1 - t_s$

Two normalized real solutions are

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

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

We can eventually combine other eigenvectors to obtain more solutions. However, we can do better with 3 eigenvectors (when $n > 2$)

Using 3 eigenvectors

Assume $\omega_1 < 0 < \omega_2 < \omega_3$. The equation is

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

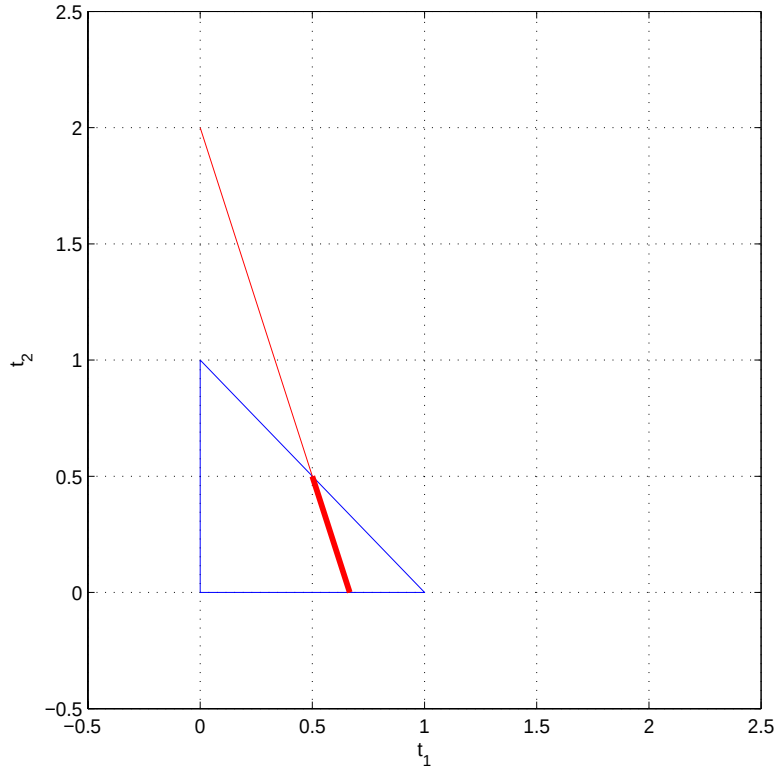
We need to satisfy

$$t_1 > 0, t_2 > 0, t_1 + t_2 \leq 1$$

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

The constraints define a triangle and the equality defines a line in the (t_1, t_2) plane

Then we combine the 3 eigenvectors corresponding to $\omega_1, \omega_2, \omega_3$



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

feasible values of (t_1, t_2) = red bold segment

The case $\omega_1 < \omega_2 < 0 < \omega_3$ is almost similar

A complex

If we apply the algorithm for real matrices with $K = (A - A^*)/2i$ we can obtain vectors b such that $Im(b^* Ab) = 0$

Assume we can find two such vectors b_1 and b_2 such that

$$Re(b_1^* Ab_1) < 0, \quad Re(b_2^* Ab_2) > 0$$

Then we apply the following result from Horn and Johnson (Topics in matrix analysis, Cambridge Univ. Press (1991)) which combines the two vectors to obtain a solution

Lemma

Let b_1 and b_2 two unit vectors with $Im(b_i^* A b_i) = 0,\ i = 1, 2$ and

$$\alpha_1 = Re(b_1^* A b_1) < 0, \quad \alpha_2 = Re(b_2^* A b_2) > 0$$

Let

$$b(t, \theta) = e^{-i\theta} b_1 + t b_2,\ t, \theta \in \mathbb{R},$$

$$\alpha(\theta) = e^{i\theta} b_1^* A b_2 + e^{-i\theta} b_2^* A b_1$$

Then

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

$\alpha(\theta) \in \mathbb{R}$ when $\theta = arg(b_2^* A b_1 - b_1^T \bar{A}\, \bar{b}_2)$

For $t_1 = (-\alpha(\theta) + \sqrt{\alpha(\theta)^2 - 4\alpha_1\alpha_2})/(2\alpha_2)$, we have

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

In case of failure

Unfortunately we cannot always find b_1 and b_2 such that

$$Re(b_1^* A b_1) < 0, \quad Re(b_2^* A b_2) > 0$$

The first remedy is to apply the same algorithm to iA with the eigenvectors of H

Note that this requires a second eigenanalysis, but now we have the eigenvectors of K and H

In case of failure (2)

If using iA fails too, we consider vectors

$$x_\theta = \cos(\theta)x + \sin(\theta)y$$

where x (resp. y) is an eigenvector of K (resp. H)

When $\theta : 0 \longrightarrow \pi$, $x_\theta^* A x_\theta$ describes an ellipse within the field of values

We look for θ such that $Im(x_\theta^* A x_\theta) = 0$. This may give the vectors we need

In case of failure (3)

If all this fail, we use the algorithm of Chorianopoulos, Psarrakos and Uhlig which is more robust, but more costly

Numerical examples

Example 1 : random real matrix of order 100 with 0 in the field of values

We compare our algorithms with those of Chorianopoulos, Psarrakos and Uhlig and Carden

The three algorithms are coded in Matlab using QR for the eigenvector computations

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

Algo	time (s)	$\lvert b^T A b \rvert$
GM (real)	0.017	$6.2617\ 10^{-14}$
CPU	0.057	$1.6012\ 10^{-15}$
Carden	0.077	$3.4417\ 10^{-15}$

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

Algo	time (s)	$\lvert b^T A b \rvert$
GM (K)	0.037	$4.5989\ 10^{-15}$
CPU	0.060	$1.5424\ 10^{-15}$
Carden	0.070	$7.1089\ 10^{-15}$

With a complex shift μ the matrix is complex and we cannot use the real algorithm

Example 2

The second example comes from Liesen and Strakoš (2005) ; see also Fischer, Ramage, Silvester and Wathen (1999)

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

with $w = [0, 1]^T$ in $\Omega = (0, 1)^2$

$$u = g \text{ on } \partial\Omega$$

stabilized Petrov–Galerkin SUPG method with bilinear finite elements on a regular Cartesian mesh

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

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

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

$$C = \frac{1}{2}\mathrm{tridiag}(-1, 0, -1)$$

We use $h = 1/16$, $n = 225$, $\nu = 0.01$ and $\delta = 0.34$

0 is not in the field of values of A. We have to shift the matrix

Example 2 $n = 225$, $\mu = 0.02$

Algo	time (s)	$\lvert b^T A b \rvert$
GM (real)	0.14	$4.8833\ 10^{-17}$
CPU	0.43	$1.2081\ 10^{-17}$
Carden	0.60	$1.0971\ 10^{-17}$

The real shift μ is inside the convex hull of the eigenvalues

Example 2 $n = 225$, $\mu = 0.055 + 0.02i$

Algo	time (s)	$\lvert b^T A b \rvert$
GM (K)	0.29	$4.9874\ 10^{-17}$
CPU	0.44	$3.4964\ 10^{-18}$
Carden	0.51	$3.1273\ 10^{-17}$

The shift μ is outside the convex hull of the eigenvalues

The computing times of GM depends on how many pairs of eigenvectors we have to consider before succeeding

Example 2 $n = 225$, $\mu = 0.055 + 0.04i$

Algo	time (s)	$\lvert b^T A b \rvert$
GM (K)	0.32	$2.9219\ 10^{-18}$
CPU	0.44	$1.9516\ 10^{-18}$
Carden	0.60	$2.0835\ 10^{-17}$

The shift μ is outside the convex hull of the eigenvalues

Example 3

This example comes from CPU

The matrix of order 200 is constructed with the Fiedler and the Moler matrices (from Matlab), F and M

Then, let $B = F + iM$. Finally the matrix A is

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

Example 3 $n = 225$, $\mu = 5000 + 10000i$

Algo	time (s)	$\lvert b^T A b \rvert$
GM (K)	0.10	$1.8645\ 10^{-11}$
CPU	0.18	$2.1398\ 10^{-12}$
Carden	0.16	$1.3690\ 10^{-12}$

The shift μ is inside the convex hull of the eigenvalues

Example 3 $n = 225$, $\mu = 10000 + 10000i$

Algo	time (s)	$\lvert b^T A b \rvert$
GM (K + H)	0.175	$3.5156\ 10^{-12}$
CPU	0.18	$8.1981\ 10^{-13}$
Carden	0.17	$6.9563\ 10^{-12}$

The shift μ is outside the convex hull of the eigenvalues

Example 3 $n = 225$, $\mu = 12000 + 10000i$

Algo	time (s)	$\lvert b^T A b \rvert$
GM (K + H + E)	0.20	$1.0785\ 10^{-12}$
CPU	0.18	$3.4106\ 10^{-13}$
Carden	0.31	$4.5702\ 10^{-12}$

The shift μ is closer to the boundary of the FOV

Example 3 $n = 225$, $\mu = 12500 + 10000i$

Algo	time (s)	$\lvert b^T A b \rvert$
GM (K + H + E)	0.20	$2.2801\ 10^{-12}$
CPU	0.25	$5.7001\ 10^{-13}$
Carden	0.38	$4.3884\ 10^{-12}$

The shift μ is very close to the boundary of the FOV

CPU's and Carden's algorithms use more than 2 eigenvalue-eigenvector computations

Conclusion

We have proposed algorithms that are (in many cases) faster than previously known algorithms

However, they need (at least) one eigenvector computation

It would be interesting to find methods that do not need eigenvector computations

Possibilities : minimization, Krylov, ...