

UNIVERZA V LJUBLJANI  
FAKULTETA ZA MATEMATIKO IN FIZIKO

Finančna matematika – 1. stopnja

Mirjam Pergar

**Računanje izotropnih vektorjev**

Delo diplomskega seminarja

Mentor: prof. dr. Bor Plestenjak

Ljubljana, 2017

## KAZALO

1. Uvod	4
1.1. Problem	4
1.2. Numerični zaklad	5
1.3. Uporaba	8
2. Realne matrike	8
2.1. Iskanje izotropnih vektorjev	10
3. Kompleksne matrike	12
3.1. Iskanje izotropnih vektorjev	12
4. Numerična analiza	17
4.1. Opis algoritma	17
4.2. Primerjava	18
4.3. Izboljšava	24
5. Zaključek	27
Literatura	28
6. Priloga	29

## Računanje izotropnih vektorjev

### POVZETEK

Delo diplomskega seminarja se ukvarja z računanjem izotropnih vektorjev  $b \in \mathbb{R}^n$  ( $\mathbb{C}^n$ ), ki so rešitev enačbe  $b^*Ab = 0$  za dano nesingularno matriko  $A \in \mathbb{R}^{n \times n}$  ( $\mathbb{C}^{n \times n}$ ). Pomemben pojem pri iskanju teh vektorjev je numerični zaklad. Računanje izotropnih vektorjev ločimo na dva primera, in sicer, ko je  $A$  realna in ko je kompleksna matrika. Ko imamo realno matriko, nas v resnici zanima problem  $b^*Hb = 0$  za hermitsko matriko  $H = (A + A^*)/2$ . Pokažemo, kako izračunamo dva izotropna vektorja iz dveh lastnih vektorjev, če je  $A$  nedefinitna. Če ima  $H$  vsaj tri različne lastne vrednosti, ki niso istega predznaka, pokažemo, da obstaja neskončno izotropnih vektorjev. Ko imamo kompleksno matriko, je problem težji in postopkov, kako pridemo do rešitve, je več. Predstavimo tri teoretične postopke Meuranta, Cardna ter Chorianopouloso, Psarrakosa in Uhliga. Meurantov algoritem najprej uporabi lastne vrednosti in vektorje poševno-hermitske matrike  $K = (A - A^*)/2i$ . Če to ne deluje, uporabi lastne vektorje hermitske matrike  $H$ , če pa še to ne deluje, potem uporabi kombinacijo lastnih vektorjev  $K$  in  $H$ . Ta postopek je implementiran v Matlabu. Naš algoritem primerjamo z drugima algoritmoma na nekaj primerih in ugotovimo, da deluje najhitreje, če najde rešitev samo z lastnimi vektorji matrike  $K$ . Na koncu opišemo še postopek bisekcije kota, ki bi izboljšal naš algoritem.

## The computation of isotropic vectors

### ABSTRACT

The bachelor thesis deals with the computation of isotropic vectors  $b \in \mathbb{R}^n$  ( $\mathbb{C}^n$ ), which are solutions of the equation  $b^*Ab = 0$  for a given nonsingular matrix  $A \in \mathbb{R}^{n \times n}$  ( $\mathbb{C}^{n \times n}$ ). An important concept when searching for these vectors is the field of values (or numerical range). We divide the computation of isotropic vectors into two parts, that is when  $A$  is a real or a complex matrix. When we have a real matrix, we are in fact interested in problem  $b^*Hb = 0$  for a hermitian matrix  $H = (A + A^*)/2$ . We show how to compute two isotropic vectors from two eigenvectors if  $A$  is non-definitive. Furthermore, if  $H$  has at least three distinct eigenvalues of different signs, then there exists an infinite number of isotropic vectors. When we are dealing with a complex matrix the problem is more difficult and more methods to find a solution exist. We present three theoretical procedures by Meurant, Carden, and Chorianopoulos, Psarrakos, Uhlig. Meurant's algorithm first uses eigenvalues and eigenvectors of the skew-Hermitian matrix  $K = (A - A^*)/2i$ . If this does not work it uses eigenvectors of the Hermitian matrix  $H$ , and if this also does not work it uses a combination of eigenvectors from  $K$  and  $H$ . We implemented this algorithm in Matlab. Further, we compare our algorithm with the algorithms from the other two papers on several examples and we conclude that our algorithm is faster if we can find a solution with only one eigenanalysis of matrix  $K$ . Lastly we describe the procedure of angle bisection, which is a possible improvement of our algorithm.

**Math. Subj. Class. (2010):** 15A60, 47A12

**Ključne besede:** izotropni vektor, numerični zaklad

**Keywords:** isotropic vector, field of values, numerical range

## 1. UVOD

V uvodnem poglavju bomo predstavili problem iskanja izotropnih vektorjev ter vse pojme, ki nam bodo v nadaljevanju pomagali pri računanju, kot je numerični zaklad. V ostalih poglavjih bomo problem razdelili na realne in kompleksne matrike ter pojasnili, kako izračunamo izotropne vektorje v vsakem primeru. Na koncu bomo en algoritem implementirali v Matlabu, ga primerjali s še dvema algoritmoma ter opisali kako bi ga lahko izboljšali. Sedaj pa opišimo glavni problem tega dela.

**1.1. Problem.** Naj bo  $A \in \mathbb{R}^{n \times n}(\mathbb{C}^{n \times n})$ ,  $\det(A) \neq 0$ . Iščemo enotski vektor  $b \in \mathbb{R}^n(\mathbb{C}^n)$ , da je

$$(1) \quad b^*Ab = 0,$$

pravimo mu *izotropni vektor*.

Bolj splošen je problem *inverznega numeričnega zaklada*, kjer iščemo enotski vektor  $b$ , za katerega velja:

$$(2) \quad b^*Ab = \mu,$$

kjer je  $\mu$  dano kompleksno število.

Očitno je, da je problem (2) možno prevesti na problem (1) za drugo matriko, saj je (2) ekvivalentno

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

Če je  $\mu$  lastna vrednost matrike  $A$ , torej velja  $Av = \mu v$ , kjer je  $v \neq 0$  lastni vektor, ki pripada  $\mu$ , potem je rešitev problema inverznega numeričnega zaklada vektor  $v$ . Če pa  $\mu$  ni lastna vrednost matrike  $A$ , je  $A - \mu I$  nesingularna in moramo lastne in izotropne vektorje izračunati. Tudi če matrika  $A$  ni realna imamo opravka s kompleksno matriko, ko je  $\mu$  kompleksno število.

Zato bomo od sedaj naprej brez škode za splošnost vse vrednosti  $\mu$  enačili z 0.

**Izrek 1.1.** ([7], izrek 1) *Naj imata  $A$  in  $b$  realne ali kompleksne elemente. Potem velja:*

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

*Dokaz.* ( $\Rightarrow$ ) Če velja  $b^*Ab = 0$ , je tudi  $(b^*Ab)^* = b^*A^*b = 0$ . Če preoblikujemo prvo enačbo na desni v  $b^*Ab + b^*A^*b$  dobimo 0. Drugo enačbo dokažemo na podoben način.

( $\Leftarrow$ ) S seštevkoma enačb na desni dobimo enačbo na levi:

$$\begin{aligned} b^*(A + A^*)b + b^*(A - A^*)b &= 0, \\ b^*(A + A^* + A - A^*)b &= 0, \\ b^*(2A)b &= 0, \\ b^*Ab &= 0. \end{aligned}$$

□

Če velja le  $b^*(A + A^*)b = 0$ , ugotovimo, da je  $\Re(b^*Ab) = 0$ . Podobno, če velja samo  $b^*(A - A^*)b = 0$ , potem je  $\Im(b^*Ab) = 0$ . Ta dejstva bomo uporabili pri računanju rešitev za kompleksne matrike. Ko sta  $b$  in  $A$  realna, je problem mnogo enostavnejši, saj moramo upoštevati le simetričen del matrike  $A$ .

Hermitski in poševno-hermitski del matrike  $A$  bomo označili s

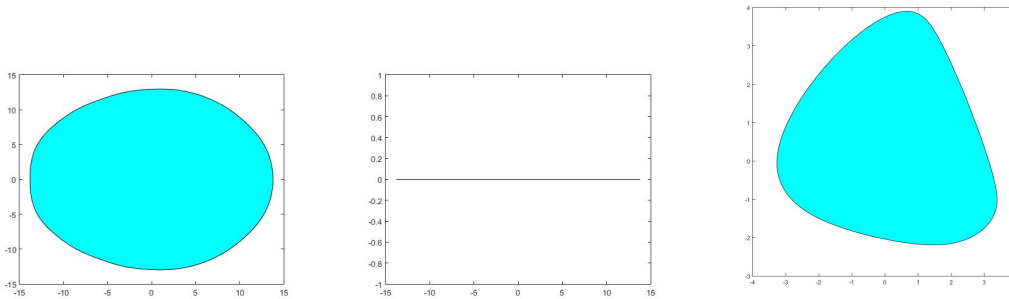
$$H = \frac{A + A^*}{2} \quad \text{in} \quad K = \frac{A - A^*}{2i}.$$

## 1.2. Numerični zaklad.

**Definicija 1.2.** *Numerični zaklad* matrike  $A \in \mathbb{C}^{n \times n}$  je podmnožica kompleksne ravnine, definirana kot

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

Očitno je numerični zaklad  $W(A)$  množica vseh Rayleighovih kvocientov matrike  $A$ . Podobno kot spekter (množica vseh lastnih vrednosti matrike) je numerični zaklad množica iz katere lahko razberemo informacije o matriki in pogosto da več informacij kot spekter sam. Lastne vrednosti hermitskih in normalnih matrik imajo uporabne lastnosti, s katerimi si pomagamo pri izračunu numeričnega zaklada. Če vzamemo hermitski del matrike, se numerični zaklad preslika na realno os in ostane le daljica, kar je podobno kot če bi vzeli realni del kompleksnega števila. Poleg tega ima ta daljica za robova najmanjšo in največjo lastno vrednost matrike. To prikazuje slika 1.



SLIKA 1. Numerični zaklad realne, hermitske in kompleksne matrike, narisano z algoritmom opisanem v [8], ki je dostopen v [9].

Če hočemo, da ima (1) vsaj eno rešitev, mora biti izhodišče vsebovano v  $W(A)$ .

Nekatere lastnosti numeričnega zaklada so:

**Izrek 1.3.** ([6], [8])

- (i)  $W(A)$  je konveksna in kompaktna podmnožica  $\mathbb{C}$ .
- (ii)  $\sigma(A) \subseteq W(A)$ , kjer  $\sigma(A)$  označuje spekter.
- (iii) Za vsako unitarno matriko  $U$  je  $W(U^*AU) = W(A)$ .
- (iv) Seštevanje s skalarjem:  $W(A + zI) = W(A) + z$  za vsak skalar  $z \in \mathbb{C}$ .
- (v) Množenje s skalarjem:  $W(zA) = zW(A)$  za vsak skalar  $z \in \mathbb{C}$ .
- (vi) Subaditivnost: Za vsaki matriki  $A, B \in \mathbb{C}^{n \times n}$  velja  $W(A + B) \subseteq W(A) + W(B)$ .
- (vii) Projekcija:  $W(H) = \Re(W(A))$  za vsako matriko  $A \in \mathbb{C}^{n \times n}$ , kjer s  $H$  označimo hermitski del matrike  $A$ .
- (viii) Če je  $A$  normalna, potem  $W(A) = \text{Co}(\sigma(A))$ , kjer s  $\text{Co}$  označimo zaprto konveksno ogrinjačo množice.
- (ix)  $W(A)$  je daljica na realni osi, če in samo če je  $A$  hermitska.

*Dokaz.* (i) Konveksnost bomo dokazali v naslednjem poglavju. Kompaktnost: Množica  $W(A)$  je zaloga vrednosti zvezne funkcije  $x \rightarrow x^*Ax$  na definicijskem območju  $\{x : x \in \mathbb{C}^n, x^*x = 1\}$  (površina enotske sfere), ki je kompaktna

množica. Ker je  $W(A)$  slika kompaktne množice pod zvezno funkcijo, sledi, da je  $W(A)$  kompaktna.

- (ii) Predpostavimo, da imamo  $\lambda \in \sigma(A)$ . Potem obstaja neničelen vektor  $x \in \mathbb{C}^n$  (za katerega lahko predpostavimo, da je enotski), za katerega velja  $Ax = \lambda x$  in zato je

$$\lambda = \lambda x^* x = x^*(\lambda x) = x^* Ax \in W(A).$$

- (iii) Naj bo  $U$  unitarna matrika. Ker unitarna transformacija preslika enotsko sfero nazaj v enotsko sfero, so kompleksna števila, ki sestavljajo množici  $W(U^*AU)$  in  $W(A)$ , enaka. Če je  $x \in \mathbb{C}^n$  in  $x^*x = 1$ , imamo

$$x^*(U^*AU)x = y^*Ay \in W(A),$$

kjer je  $y = Ux$ , torej

$$y^*y = x^*U^*Ux = x^*x = 1.$$

Torej je  $W(U^*AU) \subseteq W(A)$ .

- (iv) Imamo

$$\begin{aligned} W(A + zI) &= \{x^*(A + zI)x : x^*x = 1\} \\ &= \{x^*Ax + zx^*x : x^*x = 1\} \\ &= \{x^*Ax + z : x^*x = 1\} \\ &= \{x^*Ax : x^*x = 1\} + z \\ &= W(A) + z. \end{aligned}$$

- (v) Imamo

$$\begin{aligned} W(zA) &= \{x^*(zA)x : x^*x = 1\} \\ &= \{zx^*Ax : x^*x = 1\} \\ &= z\{x^*Ax : x^*x = 1\} \\ &= zW(A). \end{aligned}$$

- (vi) Imamo

$$\begin{aligned} W(A + B) &= \{x^*(A + B)x : x \in \mathbb{C}^n, x^*x = 1\} \\ &= \{x^*Ax + x^*Bx : x \in \mathbb{C}^n, x^*x = 1\} \\ &\subseteq \{x^*Ax : x \in \mathbb{C}^n, x^*x = 1\} + \{y^*By : y \in \mathbb{C}^n, y^*y = 1\} \\ &= W(A) + W(B). \end{aligned}$$

- (vii) Računamo

$$\begin{aligned} x^*Hx &= x^*\frac{1}{2}(A + A^*)x \\ &= \frac{1}{2}(x^*Ax + x^*A^*x) \\ &= \frac{1}{2}(x^*Ax + (x^*Ax)^*) \\ &= \frac{1}{2}(x^*Ax + \overline{x^*Ax}) \\ &= \Re(x^*Ax). \end{aligned}$$

(viii) Če je  $A$  normalna matrika, potem je  $A = U^* \Lambda U$ , kjer je  $\Lambda = \text{diag}\{\lambda_1, \dots, \lambda_n\}$  diagonalna in  $U$  unitarna matrika. Po lastnosti (iii) je  $W(A) = W(\Lambda)$  in, ker je

$$x^* \Lambda x = \sum_{i=1}^n \bar{x}_i x_i \lambda_i = \sum_{i=1}^n |x_i|^2 \lambda_i,$$

je  $W(\Lambda)$  množica vseh konveksnih kombinacij diagonalnih elementov  $\Lambda$  ( $x^* x = 1$  implicira  $\sum_i |x_i|^2 = 1$  in  $|x_i|^2 \geq 0$ ). Ker so diagonalni elementi  $\Lambda$  lastne vrednosti  $A$ , to pomeni da je  $W(A) = \text{Co}(\sigma(A))$ .

(ix) Ker je hermitska matrika tudi normalna, velja

$$W(H) = \text{Co}(\sigma(H))$$

zaradi lastnosti (viii). Vemo, da se numerični zaklad hermitske matrike projicira na realno os ter, da je  $\sigma(H)$  množica vseh lastnih vrednosti matrike  $H$ . Konveksna ogrinjača  $\sigma(H)$  je množica vseh daljic med lastnimi vrednostmi matrike  $H$  in, ker je numerični zaklad hermitske matrike daljica na realni osi, je to daljica, ki ima za krajišči najmanjšo in največjo lastno vrednostjo  $H$ . □

1.2.1. *Konveksnost.* V tem poglavju bomo dokazali, da je numerični zaklad konveksen, kar je znano tudi kot Toeplitz-Hausdorffov izrek. Načinov kako dokazati konveksnost je več, npr. v [6], vendar bomo mi izrek dokazali tako kot je dokazan v [8], za to pa potrebujemo dodatno lemo.

**Lema 1.4.** ([8], lema 2) Naj bo  $H \in \mathbb{C}^{n \times n}$  hermitska matrika in  $\mu \in W(H)$ . Potem je množica  $\mathcal{L}_H(\mu) = \{x \in \mathbb{C}^n : x^* x = 1, x^* H x = \mu\}$  povezana s potmi (t.j. obstaja pot med poljubnima dvema točkama iz te množice).

*Dokaz.* Zaradi lastnosti (iii) in (iv) lahko brez škode za splošnost predpostavimo, da je  $\mu = 0$  in  $H = \text{diag}\{d_1, d_2, \dots, d_n\}$  realna diagonalna matrika. Potem je

$$W(H) = \left\{ \sum_{j=1}^n d_j |x_j|^2 : x_1, x_2, \dots, x_n \in \mathbb{C}, \sum_{j=1}^n |x_j|^2 = 1 \right\}.$$

Naj bosta  $x = (x_j), y = (y_j) \in \mathcal{L}_H(0)$  t.j.  $x, y$  sta enotska vektorja, za katera velja

$$\sum_{j=1}^n d_j |x_j|^2 = \sum_{j=1}^n d_j |y_j|^2 = 0.$$

Pokazati želimo, da v  $\mathcal{L}_H(0)$  obstaja zvezna pot, ki povezuje  $x$  in  $y$ . Ker je vsak vektor

$$\begin{bmatrix} r_1 e^{i\theta_1} & r_2 e^{i\theta_2} & \dots & r_n e^{i\theta_n} \end{bmatrix}^T \in \mathcal{L}_H(0)$$

( $r_j \geq 0, \theta_j \in [0, 2\pi); j = 1, 2, \dots, n$ ) povezan z realnim vektorjem

$$\begin{bmatrix} r_1 & r_2 & \dots & r_n \end{bmatrix}^T \in \mathcal{L}_H(0)$$

z zvezno krivuljo

$$\begin{bmatrix} r_1 e^{i\theta_1(1-t)} & r_2 e^{i\theta_2(1-t)} & \dots & r_n e^{i\theta_n(1-t)} \end{bmatrix}^T; \quad t \in [0, 1],$$

ki je v  $\mathcal{L}_H(0)$ , sledi, da sta  $x$  in  $y$  realna in nenegativna. Potem zvezna krivulja

$$u(t) = (u_j(t)) = \left( \sqrt{(1-t)x_j^2 - ty_j^2} \right) \in \mathcal{L}_H(0) \cap \mathbb{R}^n; \quad t \in [0, 1],$$

zadošča pogojema  $u(0) = x$  in  $u(1) = y$ , s čimer je lema dokazana.  $\square$

Seveda zgornja lema drži tudi, če  $H$  pomnožimo s skalarjem. Še več, za realno diagonalno matriko  $H = \text{diag}\{d_1, d_2, \dots, d_n\}$ ,  $d_1 \geq d_2 \geq \dots \geq d_n$ , vrednost  $\sum_{j=1}^n d_j |x_j|^2$  (kjer je  $x_1, x_2, \dots, x_n \in \mathbb{C}$ ,  $\sum_{j=1}^n |x_j|^2 = 1$ ) doseže maksimum  $d_1$  pri  $|x_1| = 1$  in  $x_2 = x_3 = \dots = x_n = 0$  ter doseže minimum  $d_n$  pri  $|x_n| = 1$  in  $x_1 = x_2 = \dots = x_{n-1} = 0$ . Velja naslednja posledica.

**Posledica 1.5.** ([8], posledica 3) Naj bo  $H \in \mathbb{C}^{n \times n}$  hermitska matrika in  $\mu$  njena najmanjša ali največja lastna vrednost. Potem množica  $\mathcal{L}_H(\mu) = \{x \in \mathbb{C}^n : x^*x = 1, x^*Hx = \mu\}$  vsebuje vse enotske lastne vektorje matrike  $H$ , ki pripadajo  $\mu$ .

**Izrek 1.6** (Toeplitz-Hausdorffov izrek). ([8], izrek 4) Za vsako matriko  $A \in \mathbb{C}^{n \times n}$  je  $W(A)$  konveksna.

*Dokaz.* Dokazati moramo, da če vzamemo poljubni točki  $\mu, \nu \in W(A)$ , potem daljica s krajiščema  $\mu$  in  $\nu$  leži v  $W(A)$ . Zaradi lastnosti (iv) lahko brez škode za splošnost predpostavimo, da sta  $\mu = 0$  in  $\nu = 1$ . Naj bosta  $x, y \in \mathbb{C}^n$  taka enotska vektorja, da velja  $x^*Ax = 0$  in  $y^*Ay = 1$ . Naj bo  $H$  hermitski del matrike  $A$  in  $K$  poševno-hermitski del matrike  $A$ , kot smo ju definirali v prejšnjem razdelku. Množica  $\mathcal{L}_K(0) = \{x \in \mathbb{C}^n : x^*x = 1, x^*Kx = 0\}$  je zaradi prejšnje leme povezana s potmi. Ker sta  $x, y \in \mathcal{L}_K(0)$ , obstaja taka zvezna vektorska funkcija  $z(t) : [0, 1] \rightarrow \mathcal{L}_K(0)$ , da velja  $z(0) = x$  in  $z(1) = y$ . Sledi, da je funkcija

$$\begin{aligned} f(t) &:= z(t)^*Az(t) = z(t)^*Hz(t) + z(t)^*Kz(t) \\ &= z(t)^*Hz(t) \end{aligned}$$

realna, zvezna glede na spremenljivko  $t$  ter zadošča pogojema

$$f(0) = z(0)^*Az(0) = x^*Ax = 0$$

in

$$f(1) = z(1)^*Az(1) = y^*Ay = 1.$$

Sledi, da je daljica s krajiščema 0 in 1 vsebovana v  $W(A)$ .  $\square$

**1.3. Uporaba.** Zanimanje za izračun izotropnih vektorjev je povezano s preučevanjem delne stagnacije algoritma GMRES za reševanje linearnih sistemov z realnimi matrikami. Numerični zaklad se uporablja za preučevanje konvergence nekaterih iterativnih metod za reševanje linearnih sistemov in ima mnogo aplikacij v numerični analizi, diferencialnih enačbah, teoriji sistemov itd.

## 2. REALNE MATRIKE

V tem razdelku bomo opisali, kako izračunamo željeno število izotropnih vektorjev za realno matriko, kot je predstavljeno v članku [7]. To storimo z uporabo lastnih vektorjev matrike

$$H = \frac{A + A^*}{2}.$$

Ko je  $A$  realna matrika, nas zanima, kako izračunati rešitev naslednje enačbe:

$$(3) \quad b^*Hb = 0,$$

kjer je  $H$  realna in simetrična matrika (t.j.  $H = H^T$ ).



**Lema 2.1.** ([4], lema 3) *Izotropni vektorji realne matrike  $A$  so identični izotropnim vektorjem njenega simetričnega dela.*

*Dokaz.* To sledi iz  $b^T A b = b^T A_{sim} b + b^T A_{psim} b = b^T A_{sim} b$ , kjer je z  $A_{sim} = \frac{A+A^T}{2}$  označen simetrični del matrike  $A$  in z  $A_{psim} = \frac{A-A^T}{2}$  poševno-simetrični del matrike  $A$ .  $\square$

Velja enakost:

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

Vemo, da je  $W(A)$  simetrična glede na realno os in, da je  $0 \in W(A)$ , če in samo če  $\lambda_n \leq 0 \leq \lambda_1$ , kjer sta  $\lambda_n$  najmanjša in  $\lambda_1$  največja lastna vrednost matrike  $H$ . Naj bosta  $x_1$  in  $x_n$  realna lastna vektorja, pripadajoča  $\lambda_1$  in  $\lambda_n$ . Potem sta  $x_1^T A x_1 = x_1^T H x_1 = \lambda_1$  in  $x_n^T A x_n = x_n^T H x_n = \lambda_n$  realni točki na skrajni levi in skrajni desni zaloge vrednosti  $W(A)$  na realni osi.

Realne rešitve (3) izračunamo z uporabo lastnih vektorjev matrike  $H$ . Predpostavimo, da iščemo vektorje  $b$  z normo 1. Matriko  $H$  lahko zapišemo kot

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

kjer je  $\Lambda$  matrika, ki ima na diagonali lastne vrednosti  $\lambda_i$ , ki so realna števila.  $X$  je ortogonalna matrika lastnih vektorjev, tako da  $X^T X = I$ . Potem uporabimo ta spektralni razcep v (3):

$$b^* H b = b^* X \Lambda X^T b = 0.$$

Označimo s  $c = X^T b$  vektor projekcije  $b$  na lastne vektorje matrike  $H$ . Dobimo naslednji izrek.

**Izrek 2.2.** ([7], izrek 2) *Naj bo  $b$  rešitev problema (3). Potem vektor  $c = X^T b$  s komponentami  $c_i$  zadošča naslednjima enačbama:*

$$(4) \quad \sum_{i=1}^n \lambda_i |c_i|^2 = 0,$$

$$(5) \quad \sum_{i=1}^n |c_i|^2 = 1.$$

*Dokaz.* Enačbo (4) dokažemo tako, da  $c = X^T b$  oz.  $c^* = b^* X$  vstavimo v (3) in dobimo

$$b^* H b = b^* X \Lambda X^T b = c^* \Lambda c = 0.$$

Ker je  $\Lambda$  diagonalna matrika, lahko  $c^* \Lambda c$  zapišemo kot vsoto komponent  $\bar{c}_i \lambda_i c_i = \lambda_i |c_i|^2$  za  $i = 1, 2, \dots, n$ . Za enačbo (5) vemo, da je  $\|b\|_2 = 1$ . Če normo zapišemo s  $c$ , dobimo

$$\|b\|_2 = \|Xc\|_2 = \|c\|_2 = 1,$$

saj je  $X$  ortogonalna matrika.  $\square$

**Opomba 2.3.** Enačbi veljata samo za realna števila. Zaradi izreka 2.2 mora biti 0 konveksna kombinacija lastnih vrednosti  $\lambda_i$ . Kot smo že videli, to pomeni, da če je  $A$  definitna matrika (pozitivno ali negativno), potem (3) nima netrivialne rešitve. Drugače je  $0 \in W(A)$  in lahko vedno najdemo realno rešitev. V bistvu, kadar je  $n > 2$ , lahko najdemo neskončno izotropnih vektorjev.

**2.1. Iskanje izotropnih vektorjev.** Najprej bomo pokazali kako izračunamo izotropne vektorje, če imamo dve lastni vrednosti, kasneje pa za tri lastne vrednosti in nedefinitno matriko  $A$ .

Če niso vse lastne vrednosti  $H$  enako predznačene, potem mora za najmanjšo lastno vrednost  $\lambda_n$  veljati  $\lambda_n < 0$ . Naj bo  $k < n$  tak, da je  $\lambda_k > 0$  in naj bo  $t$  pozitivno realno število manjše od 1. Izberemo taka  $c_n$  in  $c_k$ , da velja  $|c_n|^2 = t$ ,  $|c_k|^2 = 1 - t$  in  $c_i = 0, i \neq n, k$ , ker velja enačba (5),  $t + (1 - t) = 1$ . Iz (4) mora veljati enačba:

$$\lambda_n t + \lambda_k (1 - t) = 0,$$

katere rešitev je:

$$(6) \quad t_s = \frac{\lambda_k}{\lambda_k - \lambda_n}.$$

Ker je  $\lambda_n < 0$ , je imenovalec pozitiven in  $t_s$  pozitiven ter  $t_s < 1$ . Absolutna vrednost  $c_n$  (oz.  $c_k$ ) je kvadratni koren  $t_s$  (oz.  $1 - t_s$ ). Ker je  $b = Xc$ , sta realni rešitvi:

$$b_1 = \sqrt{t_s}x_n + \sqrt{1 - t_s}x_k, \quad b_2 = -\sqrt{t_s}x_n + \sqrt{1 - t_s}x_k,$$

kjer sta  $x_n$  in  $x_k$  lastna vektorja, ki pripadata lastnima vrednostima  $\lambda_n$  in  $\lambda_k$ . Druge možnosti za predznak dajo rešitve, ki so v isti smeri kot ti dve. Ker imata izraza v rešitvah enaka imenovalca, lahko rešitvi zapišemo kot:

$$b_1 = \sqrt{\lambda_k}x_n + \sqrt{|\lambda_n|}x_k, \quad b_2 = -\sqrt{\lambda_k}x_n + \sqrt{|\lambda_n|}x_k$$

(sledi iz [4]). Vektor mora biti normiran, zato

$$b_1 = \sqrt{\frac{\lambda_k}{\lambda_k + |\lambda_n|}}x_n + \sqrt{\frac{|\lambda_n|}{\lambda_k + |\lambda_n|}}x_k, \quad b_2 = -\sqrt{\frac{\lambda_k}{\lambda_k + |\lambda_n|}}x_n + \sqrt{\frac{|\lambda_n|}{\lambda_k + |\lambda_n|}}x_k.$$

Predpostavimo, da je  $b$  realen.

**Posledica 2.4.** ([4], posledica 9) Dobljena izotropna vektorja sta ortogonalna ( $b_1^T b_2 = 0$ ), če in samo če  $\lambda_k = -\lambda_n$ .

*Dokaz.*

$$b_1^T b_2 = (\sqrt{\lambda_k}x_n + \sqrt{|\lambda_n|}x_k)^T (-\sqrt{\lambda_k}x_n + \sqrt{|\lambda_n|}x_k) = -(\lambda_n + \lambda_k).$$

□

Konstruirani rešitvi sta neodvisni in še več, ortogonalni, če  $\lambda_k = -\lambda_n$ .

Ko sta  $A$  in  $b$  realna smo dokazali naslednji izrek:

**Izrek 2.5.** ([7], izrek 3) Če je  $A$  realna in nedefinitna (t.j. ni pozitivno ali negativno definitna), potem obstajata najmanj dva neodvisna realna izotropna vektorja.

Za ta postopek lahko uporabimo vsak par pozitivne in negativne lastne vrednosti, uporaba najmanjše lastne vrednosti  $\lambda_n$  torej ni nujna. Tako lahko postopek vrne toliko rešitev kot je dvakratno število parov lastnih vrednosti matrike  $H$  z nasprotnimi predznaki, če so vse lastne vrednosti različne.

Da bi pokazali, da imamo neskončno število realnih rešitev in, da bi jih nekaj izračunali, moramo vzeti vsaj tri različne lastne vrednosti, ki ne smejo biti istega

predznaka (ko obstajajo). Predpostavimo, da imamo  $\lambda_1 < 0 < \lambda_2 < \lambda_3$  in naj bo  $t_1 = |c_1|^2$ ,  $t_2 = |c_2|^2$ . Veljati mora enačba (5)

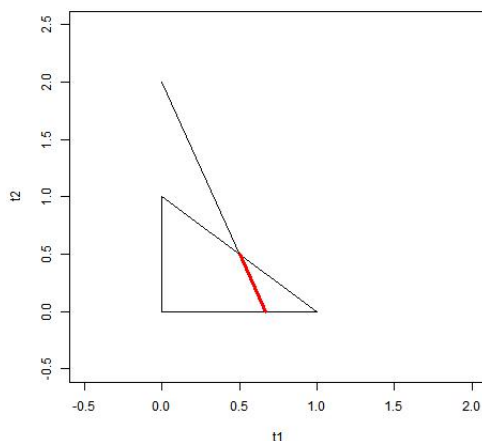
$$(7) \quad \lambda_1 t_1 + \lambda_2 t_2 + \lambda_3(1 - t_1 - t_2) = (\lambda_1 - \lambda_3)t_1 + (\lambda_2 - \lambda_3)t_2 + \lambda_3 = 0$$

s pogoji:  $t_i \geq 0, i = 1, 2$ , in  $t_1 + t_2 \leq 1$ . Torej velja zveza

$$t_2 = \frac{\lambda_3}{\lambda_3 - \lambda_2} - \frac{\lambda_3 - \lambda_1}{\lambda_3 - \lambda_2} t_1.$$

S tem je definirana premica v  $(t_1, t_2)$ -ravnini in preveriti moramo, če ta premica seka trikotnik, definiran s pogoji za  $t_1, t_2$ . Premica seka  $t_1$ -os pri  $\lambda_3/(\lambda_3 - \lambda_1)$ , kar je več kot 1, saj je  $\lambda_1 < 0$ ,  $t_2$ -os pa pri  $\lambda_3/(\lambda_3 - \lambda_2)$ , kar je tudi več kot 1. Ta premica ima negativen naklon. Vse dopustne vrednosti za  $t_1$  in  $t_2$  so dane z daljico v trikotniku. Zato obstaja neskončno število možnih pozitivnih parov  $(t_1, t_2)$ .

**Primer 2.6.** Poglejmo si enostaven zgled za  $3 \times 3$  matriko z lastnimi vrednostmi  $\lambda_1 = -1$ ,  $\lambda_2 = 1$  in  $\lambda_3 = 2$ . Matrika lastnih vektorjev  $X$  je enaka identiteti  $I$ . Enačba premice je  $t_2 = 2 - 3t_1$  s pogoji  $t_1, t_2 \geq 0$  in  $t_1 + t_2 \leq 1$ . Dopustne rešitve za  $t_1$  in  $t_2$  so dane z daljico v trikotniku, slika 2.



SLIKA 2. Dopustne rešitve so na rdeči daljici v trikotniku omejitvev.

Iz daljice lahko izberemo katerikoli par točk  $(t_1, t_2)$ , npr.  $(0.5, 0.5)$ , saj potem vemo kako izgleda vektor

$$c = \begin{bmatrix} \sqrt{0.5} \\ \sqrt{0.5} \\ 0 \end{bmatrix}.$$

Iz enačbe  $c = X^T b$  dobimo rešitev

$$b = Xc = c = \begin{bmatrix} \sqrt{0.5} \\ \sqrt{0.5} \\ 0 \end{bmatrix}.$$

Seveda je rešitev tudi  $b = -c$ . Tako lahko izračunamo neskončno izotropnih vektorjev  $b$ .

Primer  $\lambda_1 < \lambda_2 < 0 < \lambda_3$  je podoben zgornjemu, le da premica seka  $t_2$ -os pod 1. Potem dobimo rešitve  $b$  s kombiniranjem pripadajočih treh lastnih vektorjev. Takšna konstrukcija pripelje do naslednjega izreka:

**Izrek 2.7.** ([7], izrek 4) Če je  $n > 2$  in je  $A$  realna in nedefinitna, potem ima matrika  $H$  vsaj tri različne lastne vrednosti z različnimi predznaki. Potem obstaja neskončno število realnih izotropnih vektorjev.

Seveda lahko nadaljujemo z večanjem števila lastnih vrednosti. Če uporabimo štiri različne lastne vrednosti z različnimi predznaki, potem moramo na problem gledati v treh dimenzijah. Prostor, kjer je omejitvam zadoščeno, je tetraeder, torej moramo poiskati presek dane ravnine s tem tetraedrom. V splošnem, če imamo  $k$  različnih lastnih vrednosti z različnimi predznaki, definira naš problem naslednja enačba:

$$(8) \quad \sum_{i=1}^{k-1} (\lambda_i - \lambda_k) t_i + \lambda_k = 0, \quad t_i \geq 0, \quad i = 1, \dots, k-1, \quad \sum_{i=1}^{k-1} t_i \leq 1.$$

Prva enačba opisuje hiperravnino v kateri moramo poiskati presečišča te hiperravnine s telesom, definiranim s pogoji. Če je  $A$  realna matrika smo končali, saj smo pokazali, da lahko poračunamo toliko realnih rešitev kot hočemo.

### 3. KOMPLEKSNE MATRIKE

V tem razdelku si bomo pogledali, kako se računa izotropne vektorje, ko je  $A$  kompleksna matrika. Predstavljeni bodo trije teoretični postopki iz [7],[1] in [3], kako priti do izotropnih vektorjev.

#### 3.1. Iskanje izotropnih vektorjev.

3.1.1. *Meurant 1.* V nekaterih primerih lahko izračunamo rešitve s samo enim računanjem lastnih vrednosti in vektorjev matrike  $K$ , vendar to ne deluje vedno. Sredstvo, ki lahko pomaga, je, da uporabimo lastne vektorje matrike  $H$ . Če ima matrika  $A$  kompleksne elemente, nam prejšnja konstrukcija za realne matrike vrne le vektorje za katere je  $\Re(b^*Ab) = 0$ . Najprej opazimo, da lahko v nekaterih primerih uporabimo podobno konstrukcijo kot v prejšnjem razdelku, ki najde množico rešitev za hermitsko matriko  $H$  z ničelnim realnim delom ter pozitivnim in negativnim imaginarnim delom. Z uporabo treh lastnih vektorjev  $H$  obstaja neskončno rešitev dobljenih na daljici v trikotniku omejitev. Ko poljubno točko iz daljice nenehno sprehajamo po tej daljici, se tudi imaginarni del rešitve nenehno spreminja. Če sta imaginarna dela, ki ustrezata robnima točkama daljice, različnih predznakov, potem iz izreka o povprečni vrednosti sledi, da obstaja točka na daljici, ki ima ničeln imaginarni del.

3.1.2. *Meurant 2.* Druga konstrukcija algoritma iz [7] uporabi lastne vrednosti in lastne vektorje matrike  $K = (A - A^*)/(2i)$ , ki je hermitska. S konstrukcijo iz 2. razdelka lahko poiščemo tak vektor  $b$ , da je  $\Im(b^*Ab) = 0$ . S kombiniranjem lastnih vektorjev matrike  $K$  pripadajočim  $k$  pozitivnim in negativnim lastnim vrednostim lahko (v nekaterih primerih) izračunamo taka vektorja  $b_1$  in  $b_2$ , da  $\alpha_1 = \Re(b_1^*Ab_1) < 0$  in  $\alpha_2 = \Re(b_2^*Ab_2) > 0$ .

**Lema 3.1.** ([7], lema 5, [6]) Naj bosta  $b_1$  in  $b_2$  enotska vektorja z  $\Im(b_i^*Ab_i) = 0$ ,  $i = 1, 2$ , in  $\alpha_1 = \Re(b_1^*Ab_1) < 0$ ,  $\alpha_2 = \Re(b_2^*Ab_2) > 0$ . Naj bo

$$b(t, \theta) = e^{-i\theta}b_1 + tb_2, \quad t, \theta \in \mathbb{R},$$

$$\alpha(\theta) = e^{i\theta}b_1^*Ab_2 + e^{-i\theta}b_2^*Ab_1.$$

Potem je

$$b(t, \theta)^*Ab(t, \theta) = \alpha_2 t^2 + \alpha(\theta)t + \alpha_1, \quad \alpha(\theta) \in \mathbb{R},$$

ko

$$\theta = \arg(b_2^*Ab_1 - b_1^T \bar{A} \bar{b}_2).$$

Za

$$t_1 = \frac{-\alpha(\theta) + \sqrt{\alpha(\theta)^2 - 4\alpha_1\alpha_2}}{2\alpha_2}$$

imamo

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

Lema 3.1 prikazuje, kako se izračuna rešitev iz  $b_1$  in  $b_2$ . Če imamo  $b_1$  in  $b_2$ , taka da  $\alpha_1 = \Re(b_1^*Ab_1) < 0$  in  $\alpha_2 = \Re(b_2^*Ab_2) > 0$ , smo končali.

*Dokaz.* Imamo enotska vektorja  $b_1, b_2$  za katera velja  $\Im(b_i^*Ab_i) = 0$  za  $i = 1, 2$  in  $\alpha_1 = \Re(b_1^*Ab_1) < 0$ ,  $\alpha_2 = \Re(b_2^*Ab_2) > 0$ . Radi bi, da velja

$$\Re(b(t, \theta)^*Ab(t, \theta)) = 0, \quad t, \theta \in \mathbb{R}.$$

Računamo

$$\begin{aligned} b(t, \theta)^*Ab(t, \theta) &= (e^{-i\theta}b_1 + tb_2)^*A(e^{-i\theta}b_1 + tb_2) \\ &= (b_1^*(e^{-i\theta})^T + b_2^*t)A(e^{-i\theta}b_1 + tb_2) \\ &= (b_1^*(e^{i\theta})^T A + b_2^*tA)(e^{-i\theta}b_1 + tb_2) \\ &= b_1^*e^{i\theta}Ae^{-i\theta}b_1 + b_1^*e^{i\theta}Atb_2 + b_2^*tAe^{-i\theta}b_1 + b_2^*tAtb_2 \\ &= b_1^*Ab_1 + e^{i\theta}b_1^*Ab_2t + e^{-i\theta}b_2^*Ab_1t + t^2b_2^*Ab_2. \end{aligned}$$

Če gledamo samo realni del zadnje vrstice, dobimo naslednjo kvadratno enačbo:

$$\alpha_1 + \alpha(\theta)t + \alpha_2 t^2 = 0.$$

Ena rešitev te enačbe je

$$t_1 = \frac{-\alpha(\theta) + \sqrt{\alpha(\theta)^2 - 4\alpha_1\alpha_2}}{2\alpha_2}.$$

Preveriti je potrebno le še kdaj bo

$$\alpha(\theta) = e^{i\theta}b_1^*Ab_2 + e^{-i\theta}b_2^*Ab_1 \in \mathbb{R}.$$

To bo držalo, ko bo

$$\theta = \arg(b_2^*Ab_1 - \overline{b_1^*Ab_2}) = \arg(b_2^*Ab_1 - b_1^T \bar{A} \bar{b}_2).$$

□

Algoritem:

1. S kombiniranjem lastnih vektorjev  $K$ , pripadajočim pozitivnim in negativnim lastnim vrednostim, izračunamo vektorja  $b_1$  in  $b_2$ , taka da  $\alpha_1 = \Re(b_1^*Ab_1) < 0$  in  $\alpha_2 = \Re(b_2^*Ab_2) > 0$ . Uporabimo lemo 3.1 in končamo.
2. Če ne najdemo  $b_1, b_2$  potrebna za lemo 3.1, izračunamo še lastne vektorje matrike  $H$ . Ponovimo korak 1. za matriko  $\iota A$ .
3. Če postopek ne deluje niti za  $\iota A$ , uporabimo kombinacijo lastnih vektorjev  $K$  in  $H$ , kjer z  $x$  označimo lastni vektor  $K$  in z  $y$  lastni vektor  $H$ .
4. Upoštevamo vektorje  $X_\theta = \cos(\theta)x + \sin(\theta)y$ ,  $0 \leq \theta \leq \pi$ .  $X_\theta^*AX_\theta$  opiše elipso znotraj numeričnega zaklada.
5. Za dan par  $(x, y)$  iščemo presečišča elipse  $X_\theta^*AX_\theta$  z realno osjo. Z upoštevanjem, da je  $A = H + iK$ , računamo:

$$X_\theta^*AX_\theta = \cos^2(\theta)(x^*Hx + ix^*Kx) + \sin^2(\theta)(y^*Hy + iy^*Ky) + \sin(\theta)\cos(\theta)(x^*Hy + y^*Hx + i[x^*Ky + y^*Kx]).$$

Naj bo  $\alpha = \Im(x^*Hx + ix^*Kx)$ ,  $\beta = \Im(y^*Hy + iy^*Ky)$  in  $\gamma = \Im(x^*Hy + y^*Hx + i[x^*Ky + y^*Kx])$ . Ko izenačimo imaginarni del  $X_\theta^*AX_\theta$  z 0, dobimo enačbo:

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

Predpostavimo  $\cos(\theta) \neq 0$  in delimo, dobimo kvadratno enačbo za  $t = \tan(\theta)$ ,

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

6. Če ima ta enačba realne rešitve, potem dobimo vrednosti  $\theta$ , ki nam vrnejo take vektorje  $X_\theta$ , da  $\Im(X_\theta^*AX_\theta) = 0$ .
7. Če tudi ta konstrukcija ne deluje, uporabimo algoritem iz [3], opisan čez dva razdelka.

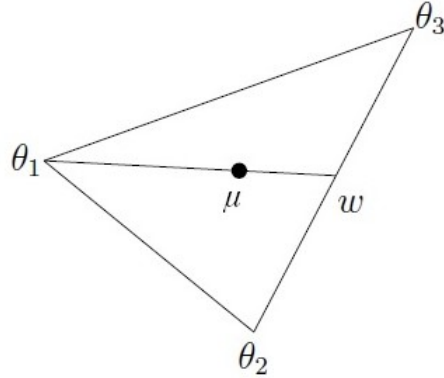
**Opomba 3.2.** Ko je  $A$  realna in imamo  $b_1 = x_1, b_2 = x_2$  za lastne vektorje  $H$ , potem je  $\theta = 0$  in lema 3.1 pove, kako se izračuna en izotropni vektor. Vendar v kompleksnem primeru ne moremo vedno najti primernih vektorjev  $b_1$  in  $b_2$ . Konstrukcija ne deluje, če imajo vse vrednosti  $\Re(y_i^*Hy_j)$  enak predznak, kjer so  $y_i$  lastni vektorji  $K$ . Ekstremen primer je Jordanov blok s kompleksno vrednostjo  $\alpha$  na diagonali in elementi 1 na naddiagonali. Potem je  $\Re(y_i^*Hy_j) = 0$ , ko  $i \neq j$ , in  $y_i^*Hy_i = -\Re(\alpha)$ . Zato so realni deli  $b^*Ab$  za vse vektorje  $b$ , ki se lahko konstruirajo, enaki.

**Opomba 3.3.** Ko je velikost problema velika, ne uporabimo zadnje konstrukcije za vse pare lastnih vektorjev, saj nas to lahko preveč stane. Uporabimo samo lastne vektorje, ki pripadajo nekaj najmanjšim in največjim lastnim vrednostim.

3.1.3. *Carden.* V tem razdelku opišemo idejo za Cardenov algoritem [1] za dano matriko  $A \in \mathbb{C}^{n \times n}$  in  $\mu \in \mathbb{C}$ . Kot smo omenili v prvem poglavju, je vseeno, če rešujemo problem (2) ali problem (1) za matriko  $A - \mu I$ . Preden se lotimo algoritma, je potrebno opisati postopek iskanja izotropnega vektorja, ki ga bomo uporabili v algoritmu. Predpostavimo, da je  $\mu$  v konveksni ogrinjači treh točk  $\theta_i \in W(A)$ , za katere smo lahko izračunali izotropne vektorje  $b_i$ . Konveksna ogrinjača teh treh točk  $\theta_i$  je trikotnik (lahko je izrojen). Radi bi, da je  $\mu$  na daljici, ki ima take robne točke, da za njih vemo ali lahko izračunamo izotropne vektorje. Zato lahko brez škode za splošnost predpostavimo, da je  $\theta_1$  ena od robnih točk te daljice. Za drugo robno točko vzamemo  $w$ , ki je presečišče daljice med  $\theta_2$  in  $\theta_3$  s premico, ki teče skozi  $\theta_1$  in  $\mu$ .

Ker je  $w$  konveksna kombinacija  $\theta_2$  in  $\theta_3$ , mu lahko določimo pripadajoč izotropni vektor. Ker pa je  $\mu$  konveksna kombinacija  $w$  in  $\theta_1$ , lahko tudi njemu določimo izotropni vektor, slika 3.

Naj bo  $\varepsilon > 0$  toleranca (npr.  $\varepsilon = 10^{-16} \|A\|$  za dvojno natančnost).



SLIKA 3. Ilustracija postopka določanja izotropnega vektorja za točko v konveksni ogrinjači treh točk iz  $W(A)$ .

Algoritem:

1. Poiščemo zunanjo aproksimacijo  $W(A)$  (pravokotnik, katerega stranice so vzporedne z realno in imaginarno osjo), tako da izračunamo najbolj levo in najbolj desno lastno vrednost  $H_\theta = (e^{i\theta} A + e^{-i\theta} A^*)/2$  za  $\theta = 0, \pi/2$ . Če  $\mu$  ni v zunanji aproksimaciji, potem  $\mu \notin W(A)$  in ustavimo algoritem, drugače nadaljujemo.
2. Če je višina ali širina zunanje aproksimacije manj kot  $\varepsilon$ , potem je  $W(A)$  približno hermitska ali poševno-hermitska (ali kompleksen premik katere od teh). Če sta višina in širina zunanje aproksimacije manj kot  $\varepsilon$ , potem je  $W(A)$  približno točka. V obeh primerih lahko ugotovimo ali je  $\mu \in W(A)$ . Če je, poiščemo pripadajoč izotropni vektor, drugače nadaljujemo.
3. Nadaljujemo s konstrukcijo notranje aproksimacije  $W(A)$  (štirikotnik z oglišči v robnih točkah  $\partial W(A)$ ) z uporabo lastnih vektorjev najbolj leve in desne lastne vrednosti  $H_\theta$ .
4. Če  $\mu$  leži v notranji aproksimaciji, lahko poiščemo izotropni vektor, ki generira  $\mu$ . Uporabimo postopek, ki je bil opisan v začetku tega razdelka. Če  $\mu$  ne leži v notranji aproksimaciji, določimo katera stranica notranje aproksimacije mu leži najbližje.
5. Izračunamo  $\hat{\mu}$ , ki je najbližja točka do  $\mu$ , ki leži na notranji aproksimaciji. Če je  $|\hat{\mu} - \mu| < \varepsilon$ , izračunamo izotropni vektor za  $\hat{\mu}$  in ga sprejmemo kot izotropni vektor za  $\mu$  ter ustavimo algoritem.
6. Posodobimo notranjo in zunanjo aproksimacijo z izračunom največje lastne vrednosti in pripadajočega lastnega vektorja  $H_\theta$ , kjer je smer  $\theta$  pravokotna na stranico notranje aproksimacije, ki je najbližja  $\mu$ . Če ne dobimo nove robne točke, ki se ni dotikala notranje aproksimacije, potem  $\mu \notin W(A)$ .
7. Ponovno preverimo, če je  $\mu$  v novi zunanji aproksimaciji. Če je, se vrnemo na četrti korak, drugače  $\mu \notin W(A)$ .

**Opomba 3.4.** Korake 4.-7. ponavljamo, dokler ni notranja aproksimacija  $\varepsilon$  blizu  $\mu$  ali dokler zunanja aproksimacija ne vsebuje  $\mu$ . Carden trdi, da se v večini primerov postopek konča v koraku 4. po le nekaj ponavljanjih.

3.1.4. *Chorianopoulos, Psarrakos in Uhlig.* V tem razdelku opišemo algoritem Chorianopoulosa, Psarrakosa in Uhliga (označimo s CPU) za inverzen problem numeričnega zaklada. Algoritem je hitrejši in daje natančne numerične rezultate tam, kjer se zgornja algoritma mnogokrat ustavita. Tak primer je ko  $\mu$  ( $\mu \in W(A)$  ali  $\mu \notin W(A)$ ) leži zelo blizu roba numeričnega zaklada. Ta algoritem uporabi za iskanje izotropnih vektorjev le nekaj najbolj osnovnih lastnosti numeričnega zaklada poleg konveksnosti, kot sta (iv) in (v).

Algoritem:

1. Izračunamo do štiri robne točke numeričnega zaklada  $p_i$  in njihove izotropne vektorje  $b_i$  za  $i = 1, 2, 3, 4$  tako, da izračunamo ekstremne lastne vrednosti, ki pripadajo enotskim lastnim vektorjem  $x_i$  matrike  $H$  in  $K$ .
2. Nastavimo  $p_i = b_i^* A b_i$ . Dobimo štiri točke  $p_i$ , ki označujejo ekstremne vrednosti  $W(A)$ , t.j. najmanjši in največji horizontalni in vertikalni razteg. Označimo jih z  $rM$  in  $rm$  za maksimalen in minimalen horizontalni razteg  $W(A)$  in z  $iM$  in  $im$  za maksimalen in minimalen vertikalni razteg  $W(A)$ . Če je  $|p_i| < 10^{-13}$  za  $i = 1, 2, 3, 4$ , potem je naš izotropni vektor kar pripadajoč enotski vektor.
3. Če med računanjem lastnih vektorjev in lastnih vrednosti ugotovimo, da je ena izmed matrik  $H$  in  $K$  definitna, t.j. da imajo njene lastne vrednosti vse enak predznak, potem vemo, da  $\mu \notin W(A)$ , in algoritem ustavimo.
4. Narišemo elipse, ki so preslikave velikega kroga kompleksne sfere  $\mathbb{C}^n$ , ki gredo skozi vse možne pare točk  $rm, rM, im$  in  $iM$ , ki imajo nasprotno predznačene imaginarne dele. Nato izračunamo presečišča vsake dobljene elipse z realno osjo.
5. Če so izračunana presečišča na obeh straneh 0, potem izračunamo izotropni vektor z lemo 3.1.
6. Če presečišča niso na obeh straneh 0, potem moramo rešiti kvadratno enačbo

$$(9) \quad (tx + (1-t)y)^* A (tx + (1-t)y) = (x^* A x + y^* A y - (x^* A y + y^* A x))t^2 + (-2y^* A y + (x^* A y + y^* A x))t + y^* A y.$$

Njene ničle določajo koordinatne osi  $W(A)$  točk na elipsah skozi točke  $x^* A x$ ,  $y^* A y \in \partial W(A)$  in so generirane s točkami v  $\mathbb{C}^n$  na velikem krogu skozi  $x$  in  $y$ .

7. To je kvadratna enačba s kompleksnimi števili. Nas zanimajo samo rešitve, ki imajo imaginaren del enak 0, saj želimo uporabiti lemo 3.1. Če imaginarni del enačbe (9) enačimo z 0, dobimo naslednjo polinomske enačbo z realnimi koeficienti:

$$(10) \quad t^2 + gt + \frac{p}{f} = 0$$

za  $q = \Im(x^* A x)$ ,  $p = \Im(y^* A y)$  in  $r = \Im(x^* A y + y^* A x)$ . Označimo  $f = p+q-r$  in  $g = (r - 2p)/f$ .



8. Enačba (10) ima realni rešitvi  $t_i$ ,  $i = 1, 2$ , ki vrneto generirajoča vektorja  $b_i = t_i x + (1 - t_i)y$  ( $i = 1, 2$ ) za realni točki. Z normalizacijo dobimo izotropne vektorje.
9. Če nobena od možnih elips ne seka realne osi na vsaki strani 0, potem preverimo, če to stori njihova skupna množica in ponovimo isti postopek.
10. Če ne najdemo take elipse niti za skupno množico, potem izračunamo še več lastnih vrednosti in lastnih vektorjev za  $A(\theta) = \cos(\theta)H + \sin(\theta)K$  za kote  $\theta \neq 0, \pi/2$  in delamo bisekcijo med točkami  $rm, rM, im, iM$ .
11. Končamo, ko najdemo definitno matriko  $A(\theta)$  ali elipso, ki seka realno os na obeh straneh 0, nakar lahko uporabimo lemo 3.1.

#### 4. NUMERIČNA ANALIZA

V tem poglavju bomo na kratko opisali naš algoritem, ki je povzet po algoritmu poglavja 3.1.2 iz [7], a mu ni identičen, in ga primerjali z algoritmoma Cardna iz [2] in CPU iz [10] na različnih primerih problema iskanja izotropnih vektorjev.

**4.1. Opis algoritma.** Najprej na kratko opišimo kako naš algoritem, opisan v poglavju 6, deluje.

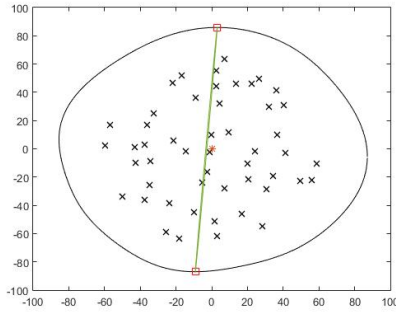
Prva stvar, ki jo algoritem naredi, je da problem (2) preobrne v začetni problem (1). Nato preveri, če je matrika  $A$  realna in če je, pokliče funkcijo `izotropniMeurantR`, ki vrne dva izotropna vektorja, izračunana kot v poglavju 2. Nadalje algoritem uporabi isti postopek kot je opisan v poglavju 3.1.2, tako da poskusi rešiti problem z lastnimi vektorji matrike  $K$ , če to ne deluje poskusi z lastnimi vektorji matrike  $H$  in na koncu še s kombinacijo lastnih vektorjev matrik  $K$  in  $H$ . V članku [7] ni napisano koliko lastnih vektorjev je potrebno oz. se jih splača izračunati, zato smo to določili sami in sicer naš algoritem računa z lastnimi vektorji, ki pripadajo trem največjim in trem najmanjšim lastnim vrednostim. Edina razlika med našim algoritmom in algoritmom iz 3.1.2 nastane zaradi premalo informacij v članku [7] o tem, kako najti vektorja  $b_1$  in  $b_2$ , da lahko uporabimo lemo 3.1. Naš algoritem najde  $b_1$  in  $b_2$  tako, kot je opisano v koraku 4. in 5. algoritma, ko kombiniramo lastne vektorje matrike  $K$  in  $H$ , kar naredi funkcija `xtheta`. Torej upoštevamo vektorje  $X_\theta = \cos(\theta)x + \sin(\theta)y$ ,  $0 \leq \theta \leq \pi$ , kjer  $X_\theta^* A X_\theta$  opiše elipso znotraj numeričnega zaklada. Za dan par  $(x, y)$  iščemo presečišča elipse  $X_\theta^* A X_\theta$  z realno osjo, kjer sta  $x$  in  $y$  lastna vektorja matrike  $K$  ali matrike  $H$  in ti presečišči vzamemo kot kandidata za  $b_1$  in  $b_2$ . Vendar naš algoritem ta korak še izboljša, saj maksimizira elipso  $X_\theta^* A X_\theta$  tako, da  $X_\theta$  definiramo kot

$$X_\theta(\varphi) = \cos(\theta)x e^{\varphi x} + \sin(\theta)y,$$

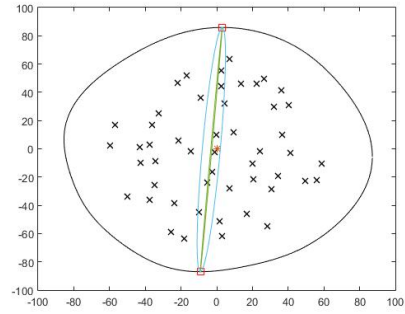
kjer je

$$\varphi = \frac{1}{2i} \ln(x^* A y (y^* A x)^{-1}).$$

Maksimiziranje elipse tukaj pomeni, da za fiksni točki  $x^* A x$  in  $y^* A y$ , ki sta krajišči velike osi, poskušamo maksimalno podaljšati malo os. Optimalno vrednost  $\varphi$  smo dobili kot odvod  $X_\theta^*(\varphi) A X_\theta(\varphi)$  po  $\varphi$ , ki mora biti enak 0. Če maksimiziramo elipso, je večja verjetnost, da bo vsebovala izhodišče in s tem je večja verjetnost, da najdemo rešitev v manj korakih, slika 4B.



(A) Z rdečo \* označimo izhodišče. Elipsa je izračunana brez  $\varphi$  in ne vsebuje izhodišča.



(B) Narišemo elipso  $X_{\theta}^*(\varphi)AX_{\theta}(\varphi)$ . Nova elipsa vsebuje izhodišče.

SLIKA 4. Numerični zaklad naključne kompleksne matrike  $A$  ter kako nam maksimiziranje elipse lahko koristi.

Če najdemo primerna  $b_1$  in  $b_2$ , da lahko uporabimo lemo 3.1, ki je definirana s funkcijo `lema31`, smo končali.

**4.2. Primerjava.** Označimo algoritme z `AlgM`, `AlgC` in `AlgCPU`. Numerične zaklade bomo narisali s funkcijo iz [5], ki s križci nariše tudi lastne vrednosti matrike  $A$ . Za vsak algoritem bomo preverili v kolikšnem času je našel (ali ni našel) rešitev, koliko izračunov lastnih vrednosti in vektorjev je potreboval (označimo kot „Koraki“) ter kako velika je napaka  $|b^*Ab|$ . Če algoritem ne najde rešitve, rečemo, da je napaka neskončna. A algoritma `AlgC` in `AlgCPU` v takem primeru povesta še, da  $\mu \notin W(A)$ . Naš algoritem tega ne pove, razen v primeru, ko sta  $A$  in  $\mu$  realna.

**4.2.1.  $A$  in  $\mu$  realna.** Za prvi primer vzemimo naključno generirano  $10 \times 10$  realno matriko  $A$  in  $\mu = 1.0419$ , ki je znotraj numeričnega zaklada.

`A = randn(10)`

TABELA 1.  $A$  realna,  $\mu = 1.0419$  in obstaja rešitev

Algoritem	Čas	Koraki	Napaka
<code>AlgM</code>	0.024636	1	1.4988e-15
<code>AlgC</code>	0.059675	4	5.2369e-16
<code>AlgCPU</code>	0.041832	2	7.0217e-16

Iz tabele 1 vidimo, da je naš algoritem `AlgM` najhitrejši, saj je samo enkrat računal lastne vektorje in lastne vrednosti, medtem ko sta druga dva algoritma to računala 4- in 2-krat. Naša napaka je malenkost večja od ostalih, vendar še vedno sprejemljiva, glede na to, da smo natančnost nastavili na  $10^{-10}$ .

Za drugi primer vzemimo  $\mu = 0$  in matriko

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}.$$

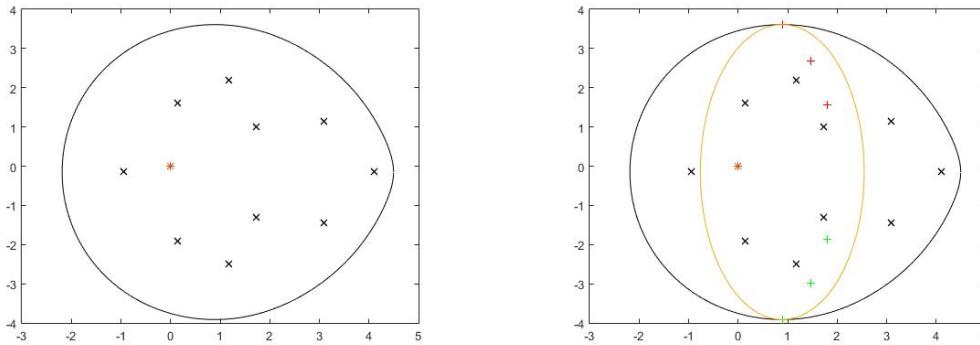
Matrika  $A$  je definitna, zato noben algoritem ne bi smel najti rešitve.

TABELA 2.  $A$  realna in ne obstaja rešitev.

Algoritem	Čas	Koraki	Napaka
AlgM	0.017245	1	Inf
AlgC	0.378286	4	Inf
AlgCPU	0.009840	1	Inf

Kot vidimo v tabeli 2, res noben algoritem ne najde rešitve. Najhitreje to naredi algoritem AlgCPU, skoraj dvakrat počasneje pa naš algoritem, saj sta algoritma samo enkrat računala lastne vrednosti in lastne vektorje. AlgC je porabil veliko več časa, saj je lastne vrednosti in vektorje računal kar štirikrat.

4.2.2.  $A$  realna,  $\mu$  kompleksen. Naj bo  $A$  naključno generirana realna  $10 \times 10$  matrika in  $\mu = -1.4575 + 0.1514i$  kompleksno število iz numeričnega zaklada, slika 5A.



(A) Z rdečo \* označimo izhodišče.

(B) Rdeči + so največji lastni vektorji, zeleni pa najmanjši lastni vektorji matrike  $K$ .<sup>1</sup>

SLIKA 5. Numerični zaklad premaknjene matrike  $A$  in kako grafično najdemo rešitev v enem koraku.

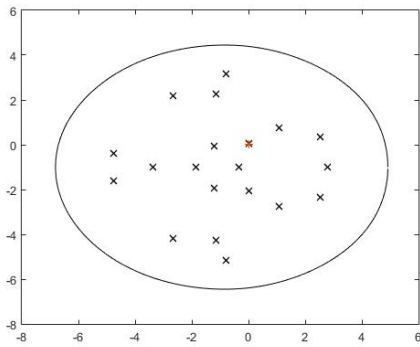
TABELA 3.  $A$  realen,  $\mu = -1.4575 + 0.1514i$ .

Algoritem	Čas	Koraki	Napaka
AlgM	0.018268	1	1.0484e-15
AlgC	0.059066	4	1.1322e-15
AlgCPU	0.032536	2	3.1525e-16

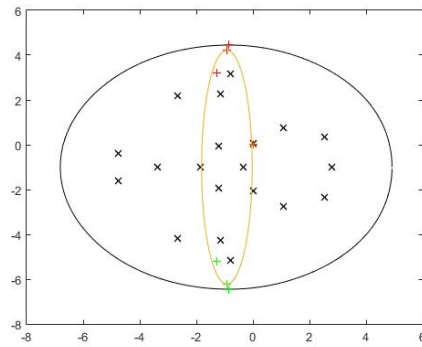
Naš algoritem je spet samo enkrat računal lastne vektorje in lastne vrednosti, zato je tudi najhitrejši kot vidimo v tabeli 3.

Za drugi primer vzamemo realno naključno generirano  $20 \times 20$  matriko  $A$  in  $\mu = 1 + i$ , slika 6A.

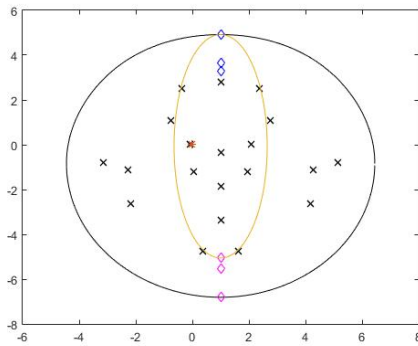
<sup>1</sup>Tukaj ne mislimo, da so v numeričnem zakladu narisani lastni vektorji, ampak točke, ki so izračunane s tem lastnim vektorjem. Npr. če je  $x$  naš lastni vektor, smo točko v numeričnem zakladu izračunali kot  $x^*Ax$ .



(A) Z rdečo \* označimo izhodišče.



(B) Rdeči + so največji lastni vektorji, zeleni pa najmanjši lastni vektorji matrike  $K$ .<sup>1</sup> Z nobeno kombinacijo ne dobimo dovolj velike elipse, ki bi vsebovala izhodišče.



(C) Modri  $\diamond$  so največji lastni vektorji, roza pa najmanjši lastni vektorji matrike  $H$ .<sup>1</sup> Rišemo elipse za matriko  $\iota A$ .

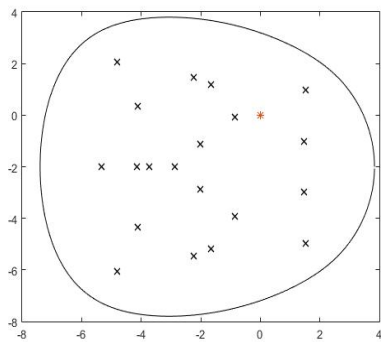
SLIKA 6. Numerični zaklad premaknjene matrike  $A$  in kako grafično najdemo rešitev v dveh korakih.

V tabeli 4 vidimo, da je naš algoritem v tem primeru dvakrat računal lastne vrednosti in lastne vektorje, kar ga je dodatno upočasnilo.

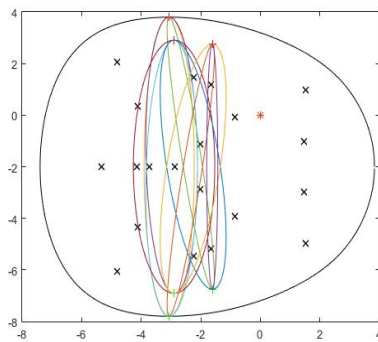
TABELA 4.  $A$  realen,  $\mu = 1 + \iota$ .

Algoritem	Čas	Koraki	Napaka
AlgM	0.025899	2	5.3363e-15
AlgC	0.017477	4	7.1089e-16
AlgCPU	0.012385	2	5.5511e-17

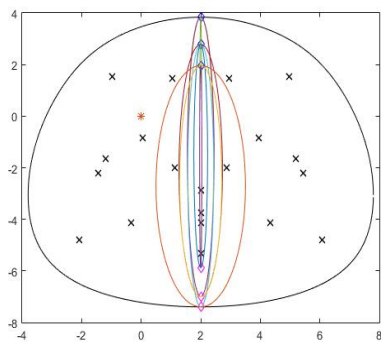
Za tretji primer vzamemo realno naključno generirano  $20 \times 20$  matriko  $A$  in  $\mu = 2 + 2\iota$ , slika 7A.



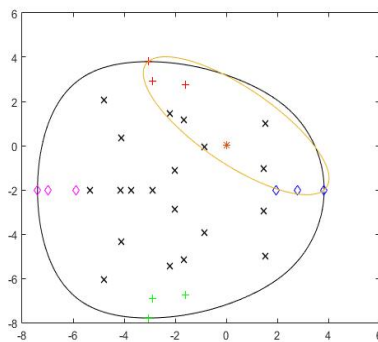
(A) Z rdečo \* označimo izhodišče.



(B) Rdeči + so največji lastni vektorji, zeleni pa najmanjši lastni vektorji matrike  $K$ .<sup>1</sup> Z nobeno kombinacijo ne dobimo dovolj velike elipse, ki bi vsebovala izhodišče.



(C) Modri  $\diamond$  so največji lastni vektorji, roza pa najmanjši lastni vektorji matrike  $H$ .<sup>1</sup> Rišemo elipse za matriko  $\nu A$ . Z nobeno kombinacijo ne dobimo dovolj velike elipse, ki bi vsebovala izhodišče.



(D) Gledamo vse kombinacije lastnih vektorjev  $K$  in  $H$  na matriki  $A$ .<sup>1</sup>

SLIKA 7. Numerični zaklad premaknjene matrike  $A$  in kako grafično najdemo rešitev v treh korakih.

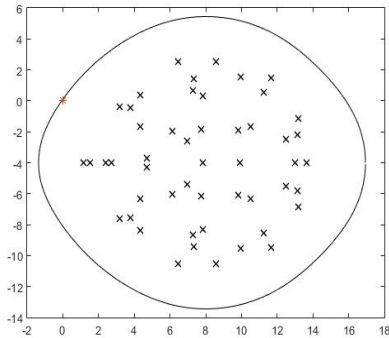
V tabeli 5 vidimo, da naš algoritem potrebuje dva izračuna lastnih vektorjev in lastnih vrednosti, vendar je v tem primeru uporabil kombinacijo lastnih vektorjev matrik  $K$  in  $H$ . Tudi `AlgCPU` porabi dva izračuna in je hitrejši od našega, `AlgC` pa porabi štiri izračune in je tudi hitrejši od našega algoritma.

TABELA 5.  $A$  realen,  $\mu = 2 + 2i$ .

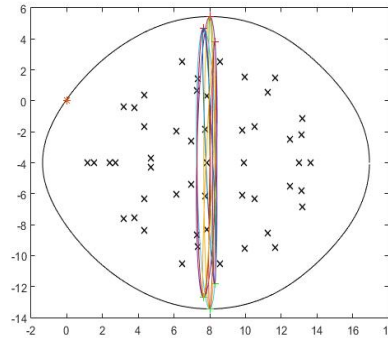
Algoritem	Čas	Koraki	Napaka
<code>AlgM</code>	0.033775	2	3.7752e-16
<code>AlgC</code>	0.020296	4	9.9301e-16
<code>AlgCPU</code>	0.012049	2	4.9651e-16

Poglejmo si grafično kako deluje naš algoritem, ko je  $\mu$  zunaj numeričnega zaklada, za realno naključno generirano  $50 \times 50$  matriko  $A$  in  $\mu = -8 + 4i$ , slika 8. V takem

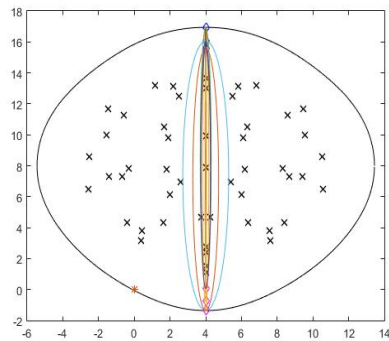
primeru naš algoritem ne ugotavlja ali je  $\mu$  v numeričnem zakladu ali ne, saj do zadnjega trenutka išče možne kombinacije elips. To bi bila ena od možnih izboljšav algoritma.



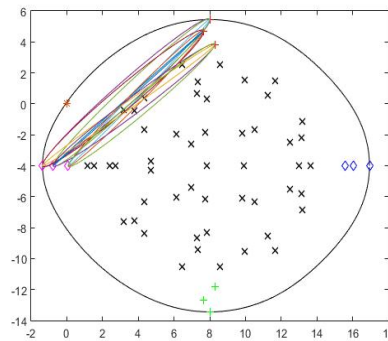
(A) Z rdečo \* označimo izhodišče.



(B) Rdeči + so največji lastni vektorji, zeleni pa najmanjši lastni vektorji matrike  $K$ .<sup>1</sup> Z nobeno kombinacijo ne dobimo dovolj velike elipse, ki bi vsebovala izhodišče.



(C) Modri  $\diamond$  so največji lastni vektorji, roza pa najmanjši lastni vektorji matrike  $H$ .<sup>1</sup> Rišemo elipse za matriko  $\nu A$ . Z nobeno kombinacijo ne dobimo dovolj velike elipse, ki bi vsebovala izhodišče.



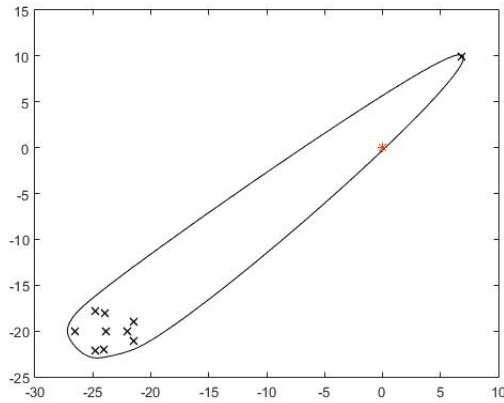
(D) Gledamo vse kombinacije lastnih vektorjev  $K$  in  $H$  na matriki  $A$ .<sup>1</sup> Narisane so le tiste elipse, ki imajo možnost vsebovati izhodišče.

SLIKA 8. Numerični zaklad premaknjene matrike  $A$  in kako grafično ne najdemo rešitve.

Poglejmo primer iz članka [3], za matriko  $A = \text{randn}(10) + (3+3i) * \text{ones}(10)$ . Naš algoritem ne najde rešitve, čeprav je izhodišče znotraj numeričnega zaklada, slika 9, druga algoritma pa jo najdeta. Kako bi se izognili takšnim primerom opišemo v naslednjem razdelku.

TABELA 6.  $A$  realen,  $\mu = 23.2 + 20i$  in naš algoritem ne najde rešitve.

Algoritem	Čas	Koraki	Napaka
AlgM	0.013259	2	Inf
AlgC	0.022176	6	1.1235e-14
AlgCPU	0.001088	3	1.7764e-15



SLIKA 9. Numerični zaklad premaknjene matrike in izhodišče, označeno z rdečo \*, kjer naš algoritem ne najde rešitve, čeprav bi jo moral.

4.2.3. *A in  $\mu$  kompleksna.* V tem poglavju bomo za matriko  $A$  vzeli matriko, ki je uporabljena tudi v [7] in [3], velikosti  $200 \times 200$ , ki se konstruira iz Fiedlerjeve  $F$  in Molerjeve  $M$  matrike. Potem je  $B = F + \nu M$  in  $A$  (v Matlab kodi):

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

Za prvi primer vzemimo  $\mu = 5000 + 10000\nu$ , slika 10. V tabeli 7 vidimo, da je naš algoritem najhitrejši, saj je računal samo lastne vrednosti in vektorje matrike  $K$  in tudi napaka je najmanjša.

TABELA 7.  $A$  kompleksna,  $\mu = 5000 + 10000\nu$ .

Algoritem	Čas	Koraki	Napaka
AlgM	0.012503	1	4.0030e-16
AlgC	0.029377	4	1.2561e-15
AlgCPU	0.017893	2	4.7429e-16

Za drugi primer vzamemo  $\mu = 12000 + 10000\nu$ , ki je bližje robu, slika 10. V tabeli 8 lahko vidimo, da naš algoritem najde rešitev v dveh korakih, zato je počasnejši od AlgCPU, ampak hitrejši od AlgC. V originalnem članku Meuranta iz [7] je za isti primer Meurant našel rešitev s kombinacijo lastnih vektorjev matrik  $K$  in  $H$ , medtem ko naš algoritem najde rešitev brez kombiniranja. Kot smo omenili v opisu algoritma, je to posledica tega, da maksimiziramo elipso, tako je večja verjetnost, da je izhodišče vsebovano v njej.

TABELA 8.  $A$  kompleksna,  $\mu = 12000 + 10000\nu$ .

Algoritem	Čas	Koraki	Napaka
AlgM	0.433484	2	1.7053e-13
AlgC	0.656863	7	9.0949e-12
AlgCPU	0.03295	4	1.8190e-12

Za tretji primer vzamemo  $\mu = 12500 + 10000\nu$ , ki je še bližje robu kot v prejšnjem primeru, slika 10. V tabeli 9 vidimo, da kadar naš algoritem uporabi kombinacijo

lastnih vektorjev in lastnih vrednosti matrik  $K$  in  $H$ , je AlgCPU mnogo hitrejši, medtem ko je AlgC malo počasnejši od našega algoritma.

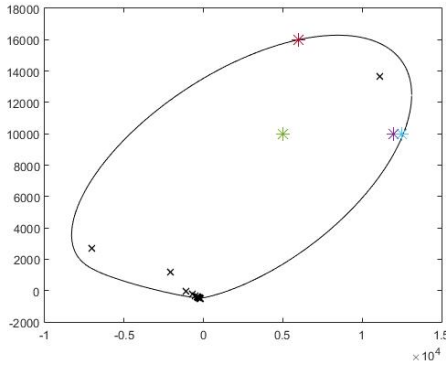
TABELA 9. A kompleksna,  $\mu = 12500 + 10000i$ .

Algoritem	Čas	Koraki	Napaka
AlgM	0.407477	2	1.8270e-12
AlgC	0.453814	8	9.2751e-12
AlgCPU	0.016621	5	7.7901e-13

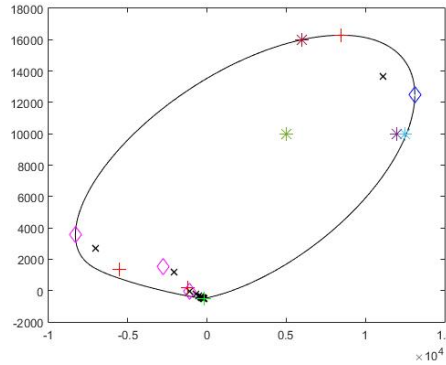
Za zadnji primer vzemimo  $\mu = 6000 + 16000i$ , ki je blizu roba, ampak vseeno ni v numeričnem zakladu, slika 10. Vidimo, da noben algoritem ne najde rešitve, kar je pravilno. Spet to najhitreje naredi AlgCPU, potem naš algoritem in na koncu AlgC.

TABELA 10. A kompleksna,  $\mu = 6000 + 16000i$  in algoritem ne najde rešitve.

Algoritem	Čas	Koraki	Napaka
AlgM	0.423094	2	Inf
AlgC	0.466800	11	Inf
AlgCPU	0.032628	10	Inf



(A) Zelena \* =  $5000 + 10000i$   
 Vijolična \* =  $12000 + 10000i$   
 Modra \* =  $12500 + 10000i$   
 Rdeča \* =  $6000 + 16000i$



(B) Rdeči + so največji lastni vektorji, zeleni pa najmanjši lastni vektorji matrike  $K$ . Modri  $\diamond$  so največji lastni vektorji, roza pa najmanjši lastni vektorji matrike  $H$ .<sup>1</sup>

SLIKA 10. Numerični zaklad matrike A z dodanimi lastnimi vektorji iz katere lahko za vsak primer hitro ugotovimo koliko korakov potrebujemo, da najdemo elipso, ki bo ali ne bo vsebovala  $\mu$ .

**4.3. Izboljšava.** Čeprav naš algoritem že izboljša Meurantov originalen algoritem, bi ga lahko še izboljšali, da bi deloval tudi v primerih, ko je izhodišče zelo blizu roba. To bi storili z bisekcijo kotov, podobno kot v desetem koraku CPU algoritma. Za podrobnejši opis si iz CPU algoritma sposodimo nekaj oznak za točke, ki označujejo ekstremne vrednosti  $W(A)$ , t.j. najmanjši in največji horizontalni in vertikalni razteg. Označimo jih z  $rM$  in  $rm$  za maksimalen in minimalen horizontalni

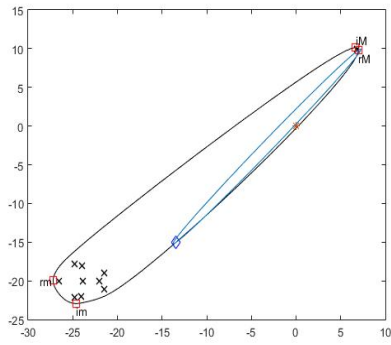


razteg  $W(A)$  in z  $iM$  in  $im$  za maksimalen in minimalen vertikalni razteg  $W(A)$ . Ko naš algoritem ne najde primerne elipse tudi, če pogledamo vse kombinacije lastnih vektorjev  $K$  in  $H$ , si pomagamo z izračunom lastnih vektorjev in vrednosti  $A(\theta) = \cos(\theta)H + \sin(\theta)K$  za kote  $\theta \neq 0, \pi/2$ . Odvisno od položaja izhodišča glede na numerični zaklad lahko ločimo štiri primere:

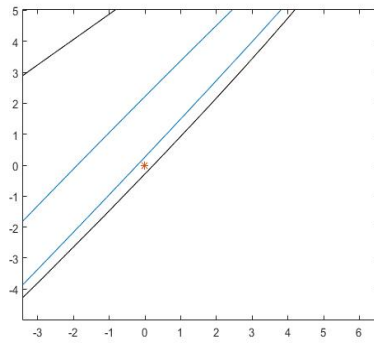
- Če so vsa presečišča elipse z realno osjo pozitivna in je  $\Im(rm) < 0$ , potem naredimo bisekcijo kota  $\pi/2$  in  $\pi$  in izračunamo največjo lastno vrednost  $A(3/4\pi)$  ter pripadajoč lastni vektor  $x_{nov}$ .  $p = x_{nov}^*Ax_{nov}$  je točka na robu numeričnega zaklada med točkama  $rm$  in  $iM$ . Če je  $\Im(p) < 0$ , potem iščemo presečišča  $p-iM$  elipse z realno osjo, če pa je  $\Im(p) > 0$ , potem iščemo presečišča  $rm-p$  elipse z realno osjo.
- Če so vsa presečišča elipse z realno osjo pozitivna in je  $\Im(rm) > 0$ , potem naredimo bisekcijo kota  $3/2\pi$  in  $\pi$  in izračunamo največjo lastno vrednost  $A(5/4\pi)$  ter pripadajoč lastni vektor  $x_{nov}$ .  $p = x_{nov}^*Ax_{nov}$  je točka na robu numeričnega zaklada med točkama  $im$  in  $rm$ . Če je  $\Im(p) < 0$ , potem iščemo presečišča  $p-rm$  elipse z realno osjo, če pa je  $\Im(p) > 0$ , potem iščemo presečišča  $im-p$  elipse z realno osjo.
- Če so vsa presečišča elipse z realno osjo negativna in je  $\Im(rM) < 0$ , potem naredimo bisekcijo kota  $0$  in  $\pi/2$  in izračunamo največjo lastno vrednost  $A(1/4\pi)$  ter pripadajoč lastni vektor  $x_{nov}$ .  $p = x_{nov}^*Ax_{nov}$  je točka na robu numeričnega zaklada med točkama  $rM$  in  $iM$ . Če je  $\Im(p) < 0$ , potem iščemo presečišča  $p-iM$  elipse z realno osjo, če pa je  $\Im(p) > 0$ , potem iščemo presečišča  $rM-p$  elipse z realno osjo.
- Če so vsa presečišča elipse z realno osjo negativna in je  $\Im(rM) > 0$ , potem naredimo bisekcijo kota  $3/2\pi$  in  $2\pi$  in izračunamo največjo lastno vrednost  $A(7/4\pi)$  ter pripadajoč lastni vektor  $x_{nov}$ .  $p = x_{nov}^*Ax_{nov}$  je točka na robu numeričnega zaklada med točkama  $im$  in  $rM$ . Če je  $\Im(p) < 0$ , potem iščemo presečišča  $p-rM$  elipse z realno osjo, če pa je  $\Im(p) > 0$ , potem iščemo presečišča  $im-p$  elipse z realno osjo.

Bisekcijo izvajamo poljubno dolgo, dokler ne najdemo presečišči elipse z realno osjo na obeh straneh 0, ali dokler  $A(\theta)$  ne postane definitna.

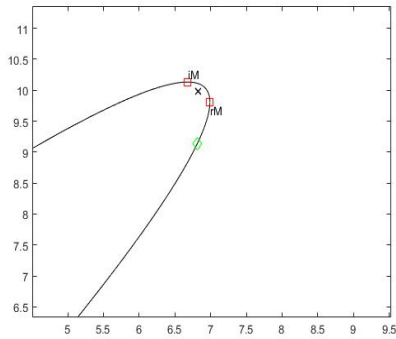
Vrnimo se k primeru iz tabele 8 in pogledajmo kako bi uporabili bisekcijo kota v tem primeru. Očitno bomo uporabili zadnjo točko iz bisekcije, saj so vsa presečišča negativna in  $\Im(rM) > 0$ . Kako smo našli rešitev, je grafično predstavljeno na sliki 11.



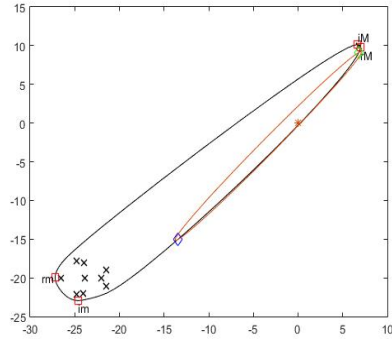
(A) Prva točka  $p$ , ki jo dobimo iz bisekcije, je označena z modrim  $\diamond$ . Očitno moramo iskati presečišči  $p-rM$  elipse z realno osjo.



(B) Vidimo, da izhodišče ni vsebovano v elipsi, kar pomeni, da ne najdemo presečišči z realno osjo na vsaki strani ničle.



(C) Drugo točko  $p$  dobimo iz bisekcije kota  $7/4\pi$  in  $2\pi$ , označeno z zelenim  $\diamond$ .



(D) Očitno moramo iskati presečišči  $p-p_{star}$  elipse z realno osjo. Tokrat je izhodišče vsebovano v elipsi in končamo.

SLIKA 11. Kako najdemo rešitev z uporabo bisekcije kota v dveh korakih.

## 5. ZAKLJUČEK

V delu diplomskega seminarja smo pokazali, kako najdemo izotropne vektorje, če imamo realno ali kompleksno začetno matriko. Za realne matrike smo ugotovili, da je dovolj gledati le simetričen del začetne matrike in opisali enostaven algoritem, ki nam vrne poljubno število izotropnih vektorjev pod pogojem, da imamo tri različne lastne vrednosti, ki ne smejo biti istega predznaka. Za kompleksne matrike smo opisali tri različne algoritme treh različnih avtorjev. Algoritem Meuranta išče izotropne vektorje s pomočjo lastnih vektorjev in lastnih vrednosti matrike  $K$ , če to ne deluje, poskusi z matriko  $H$ , če pa še to ne deluje pa kombinira lastne vektorje matrik  $K$  in  $H$ . Algoritem najde izotropne vektorje tako, da poišče presečišči realne osi z elipso, ki gre skozi pare točke, ki smo jih izračunali z lastnimi vektorji, nato uporabi lemo 3.1. Algoritem Cardna deluje tako, da najde tri take točke, da je 0 v njihovi konveksni ogrinjači in da za njih pozna pripadajoče izotropne vektorje, potem je iskan izotropni vektor konveksna kombinacija drugih izotropnih vektorjev. Algoritem CPU najprej izračuna ekstremne vrednosti numeričnega zaklada, potem pa išče presečišči elipse, ki gre skozi vsak par teh točk, z realno osjo. Če najde taki točki, izračuna rešitev z uporabo leme 3.1, če ne pa uporabi bisekcijo kota in ponovi postopek. Algoritem Meuranta smo tudi sprogramirali v Matlabu in preizkusili na primerih, ko sta  $A$  in  $\mu$  realna, ko je  $A$  realen in  $\mu$  kompleksen ter ko sta  $A$  in  $\mu$  kompleksna. Opazili smo, da naš algoritem deluje najhitreje, če najde rešitev samo z lastnimi vektorji in lastnimi vrednostmi matrike  $K$ , drugače je algoritem CPU hitrejši v vseh primerih, Cardnov algoritem pa je hitrejši samo pri majhnih matrikah. Napake so povsod primerljive in sprejemljive, glede na našo toleranco  $10^{-10}$ . Vključili smo tudi primere, ko algoritem ne deluje. To je, ko je  $A$  definitna,  $\mu$  zunaj numeričnega zaklada ali zelo blizu roba. V primerih, ko je  $\mu$  znotraj numeričnega zaklada, a naš algoritem ne najde rešitve, smo opisali možno izboljšavo. Ta postopek je bisekcija kota, ki glede na položaj izhodišča, poišče novo točko na robu numeričnega zaklada. Potem iščemo presečišči realne osi z elipso nove točke in primerne ekstremne točke numeričnega zaklada. Če sta novi presečišči na vsaki strani izhodišča, uporabimo lemo 3.1 in končamo, če ne ponovimo bisekcijo. Tako lahko v večini primerov najdemo izotropni vektor.

## LITERATURA

- [1] R. Carden, *A simple algorithm for the inverse field of values problem*, Inverse Probl. **25** (2009) 1–9.
- [2] R. Carden, *inversefov.m*, verzija 6. 2. 2011, [ogled 5. 5. 2016], dostopno na [http://www.caam.rice.edu/tech\\_reports/2009/](http://www.caam.rice.edu/tech_reports/2009/).
- [3] C. Chorianopoulos, P. Psarrakos in F. Uhlig, *A method for the inverse numerical range problem*, Electron. J. Linear Algebra **20** (2010) 198–206.
- [4] N. Ciblak, H. Lipkin, *Orthonormal isotropic vector bases*, v: Proceedings of DETC'98, 1998 ASME Design Engineering Technical Conferences (1998).
- [5] W. Henao, *fovals.m*, [ogled 27. 7. 2016], dostopno na [https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/4679/versions/1/previews/fovals.m/index.html?access\\_key=](https://www.mathworks.com/matlabcentral/mlc-downloads/downloads/submissions/4679/versions/1/previews/fovals.m/index.html?access_key=).
- [6] Johnson, C. R., *Numerical determination of the field of values of a general complex matrix*, SIAM J. Numer. Anal. **15** (1978) 595–602.
- [7] G. Meurant, *The computation of isotropic vectors*, Numer. Alg. **60** (2012) 193–204.
- [8] P. Psarrakos, M. Tsatsomeros, *Numerical range: (in) a matrix nutshell*, Math. Notes from Washington State University, Vol **45**, No. 2 (2002) 45–57.
- [9] P. Psarrakos, *nr.m*, [ogled 12. 6. 2017], dostopno na <http://www.math.wsu.edu/faculty/tsat/files/matlab/nr.m>.
- [10] F. Uhlig, *invfovCPU.m*, verzija 22. 3. 2011, [ogled 5. 5. 2016], dostopno na [http://www.auburn.edu/~uhligfd/m\\_files/invfovCPU.m](http://www.auburn.edu/~uhligfd/m_files/invfovCPU.m).

## 6. PRILOGA

```

function [b, napaka, korak] = izotropniMeurant(A, mu)
%ce je mozno, ta funkcija izracuna izotropne vektorje b, ki so
%resitev enacbe mu=b'Ab.
%Vhod: A...realna ali kompleksna matrika
%      mu ... kompleksno stevilo
%
%
%Izhod: b...izotropni vektor za mu
%       napaka ... norm(b'Ab-mu)
%       korak ... 0, če algoritem ne izračuna nobenih 1. vekt.
%                1, če algoritem izračuna le lastne vektorje K
%                2, če algoritem izračna še lastne vektorje H
%                3, če algoritem uporabi lastne vektorje K in H
warning off

if nargin ==1,
    mu=0;
    %tol = 1e-14;
end

[n, m] = size(A);
%preverimo, ce A kvadratna matrika
if n~=m,
    disp('Matrika A ni kvadratna!')
    return
end

%problem preobrnemo v b*(A-muI)b=0
A = A-mu*eye(n);

%A realna, algoritem za realne mtr.
if isreal(A)==1,
    [b, napaka, korak] = izotropniMeurantR(A);
    return
end

K=(A-A')/(2*1i);
opts.disp = 0;
opts.maxit = 1000;
opts.tol = 10^-4;

napaka = Inf;
korak = 0;
b = 0;

[x, vred_k1] = eigs(K,3,'lr',opts);

```

```

[y, vred_k2] = eigs(K,3,'sr',opts);

korak = korak + 1;

X = [x,y];
LV = [diag(vred_k1);diag(vred_k2)];

[b, napaka] = funkcija(A,X,LV,n);

if b~=0,
    return
end

%ponovimo isti postopek za H in matriko iA
H=(A+A')/2;
[xx, vred_h1] = eigs(H,3,'lr');

[yy, vred_h2] = eigs(H,3,'sr');

korak = korak + 1;

XX = [xx,yy];
LV1 = [diag(vred_h1);diag(vred_h2)];

[b, napaka] = funkcija(1i*A,XX,LV1,n);

if b~=0,
    return
end

korak = korak +1;

for k=1:size(X,2),
    for j = 1: size(XX,2),
        fi=(log(X(:,k)'*A*XX(:,j)*inv(XX(:,j)'*A*X(:,k))))/(2i);

        [b1, b2] = xtheta(X(:,k)*exp(fi*1i), XX(:,j), A);

        if sum(abs(b1)<ones(n,1)*1e-10)==0 &&
            sum(abs(b2)<ones(n,1)*1e-10)==0,
            b1 = b1/norm(b1);
            b2 = b2/norm(b2);

            if (abs(imag(b1'*A*b1))<1e-10) &&
                (abs(imag(b2'*A*b2))<1e-10),
                b = lema_31(b1,b2,A);
                if sum(abs(b)<ones(n,1)*1e-10)==0,
                    napaka = abs(b'*A*b);

```

```

        return
    end
end
end
end
end
disp('Algoritem ne najde resitve, uporabi algoritem CPU')

end

function [b, napaka, korak] = izotropniMeurantR(A)
%Resi problem izotropnih vektorjev, če je matrika A realna in mi
%realen oz. 0

%racunamo b'Hb=0
H=(A+A')/2;

opts.disp = 0;
opts.maxit = 1000;
opts.tol = 10^-4;

[x, vred_h1] = eigs(H,1,'la',opts);

[y, vred_h2] = eigs(H,1,'sa',opts);

if (vred_h1 * vred_h2) < 0,
    b1 = sqrt(vred_h1 / (vred_h1 + abs(vred_h2)))*y +
sqrt( abs(vred_h2) / (vred_h1 + abs(vred_h2)))*x;
    b2 = -sqrt(vred_h1 / (vred_h1 + abs(vred_h2)))*y +
sqrt( abs(vred_h2) / (vred_h1 + abs(vred_h2)))*x;
    b = [b1 ,b2];
    napaka = [abs(b1'*H*b1),abs(b2'*H*b2)];
    korak = 1;
else
    b = 0;
    napaka = Inf;
    korak = 0;
    disp('H je definitna')
end

function [b1,b2] = xtheta(x,y,A)

H=(A+ A')/2;
K=(A- A')/(2*1i);

alfa = imag(x'*H*x + 1i*x'*K*x);
beta = imag(y'*H*y + 1i*y'*K*y);
gama = imag(x'*H*y + y'*H*x + 1i*(x'*K*y + y'*K*x));

```

```

%resimo enacbo  $\beta t^2 + \gamma t + \alpha = 0$ 
t1 = (-gamma + sqrt(gama^2 -4*beta*alfa))/(2*beta);
t2 = (-gamma - sqrt(gama^2 -4*beta*alfa))/(2*beta);

%preverimo, če sta resitvi realni
if (abs(imag(t1))<1e-10),
    theta1 = atan(real(t1));%t = tan(theta)
    %(predpostavili smo, da  $\cos(\theta) \neq 0$ )
    b1 = cos(theta1)*x + sin(theta1)*y;
else
    b1 = 0;
end

if (abs(imag(t2))<1e-10),
    theta2 = atan(real(t2));
    b2 = cos(theta2)*x + sin(theta2)*y;
else
    b2 = 0;
end

function [b, napaka] = funkcija(A,X,LV,n)
%pomožna funkcija
b = 0;
napaka = Inf;
for k=1: size(X,2)-1,
    for j = (k+1):size(X,2),
        if (LV(k,:)*LV(j,:) < 0),
            fi = (log(X(:,k)'*A*X(:,j)*inv(X(:,j)'*A*X(:,k))))/(2i);
            [b1, b2] = xtheta(X(:,k)*exp(fi*i), X(:,j), A);
            if sum(abs(b1)<ones(n,1)*1e-10)==0 &&
                sum(abs(b2)<ones(n,1)*1e-10)==0,
                b1 = b1/norm(b1);
                b2 = b2/norm(b2);
                if (abs(imag(b1'*A*b1))<1e-10) &&
                    (abs(imag(b2'*A*b2))<1e-10),
                    b = lema_31(b1,b2,A);
                    if sum(abs(b)<ones(n,1)*1e-10)==0,
                        napaka = abs(b'*A*b);
                        return
                    end
                end
            end
        end
    end
end
end
end
end

function [b] = lema_31(b1,b2,A)

```



```

%Vhod: enotska vektorja b1 in b2, za katera velja  $\text{imag}(b_i' * A * b_i) = 0$  za
%i=1,2, za dano matriko A
%Izhod: izotropni vektor b

a1 = real(b1'*A*b1);
a2 = real(b2'*A*b2);

if a1<0,
    if a2>0,
        alfa1 = a1;
        alfa2 = a2;
    else
        b=0;
        return
    end
elseif a1>0,
    if a2<0,
        alfa2 = a1;
        alfa1 = a2;
        [b2, b1] = deal(b1, b2);
    else
        b=0;
        return
    end
else
    b = 0;
    return
end
bb = @(t,th) exp(-1i*th)*b1 + t*b2;
al = @(th) exp(1i*th)*b1'*A*b2 + exp(-1i*th)*b2'*A*b1;

th = angle(b2'*A*b1 - b1.'*conj(A)*conj(b2));
t1 = (-al(th) + sqrt(al(th)^2 -4*alfa1*alfa2))/(2*alfa2);

b = bb(t1,th)/norm(bb(t1,th));

end

```