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

Računanje izotropnih vektorjev

Povzetek

Delo diplomskega seminarja se ukvarja z izračunom izotropnih vektorjev $b \in \mathbb{R}^n(\mathbb{C}^n)$, ki so rešitev enačbe $b^*Ab = 0$, za dano matriko $A \in \mathbb{R}^{n\times n}(\mathbb{C}^{n\times n})$, $det(A) \neq 0$. Pomemben pojem pri iskanju teh vektorjev je numerični zaklad, definiran kot $W(A) = \{x^*Ax \colon x \in \mathbb{C}^n,\ x^*x = 1\}$. Predstavljene in dokazane so njegove uporabne lastnosti, kot je npr. konveksnost. Računanje izotropnih vektorjev ločimo na 2 dela, 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 2 izotropna vektorja iz 2 lastnih vektorjev, če je A nedefinitna, in še več, če ima H vsaj 3 različne lastne vrednosti,ki ne smejo biti istega predznaka, pokažemo, da obstaja neskončno izotropnih vektorjev. Ko imamo kompleksno matriko, je problem težji in postopkov, kako pridemo do rešitve, več. Predstavimo 3 teoretične postopke Meuranta, Cardna in Chorianopoulos, Psarrakos, Uhliga. Meurantov algoritem najprej uporabi lastne vrednosti in vektorje hermitske matrike $K = (A - A*)/2\imath$, če to ne deluje uporabi hermitsko matriko H, če pa še to ne deluje pa uporabi kombinacijo lastnih vektorjev od K in H. Po tem postopku tudi sprogramiramo kodo v Matlabu. Nadalje naš algoritem primerjamo z drugima algoritmoma na vseh možnih primerih in ugotovimo ...

The computation of isotropic vectors

Abstract

The graduation thesis deals with the computation of isotropic vectors $b \in \mathbb{R}^n(\mathbb{C}^n)$, that are solutions of the equation $b^*Ab = 0$, for a given matrix $A \in \mathbb{R}^{n\times n}(\mathbb{C}^{n\times n})$, $det(A) \neq 0$. An important concept when searching for these vectors is the field of values (or numerical range), defined as $W(A) = \{x^*Ax : x \in \mathbb{C}^n,\ x^*x = 1\}$. Some of its useful properties are presented and proved, such as convexity. We divide the computation of isotropic vectors into 2 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 2 isotropic vectors from 2 eigevectors, if A is non-definitive, and furthermore, if H has at least 3 distinct eigenvalues of different signs, there exists an infinite number of isotropic vectors. When we are dealing with a complex matrix the problem is more difficult and more methods of how to find a solution exist. We present 3 theoretical procedures by Meurant, Carden and Chorianopoulos, Psarrakos, Uhlig. Meurant's algorithm first uses eigenvalues and eigenvectors of the hermitian matrix $K = (A - A^*)/2\imath$, if this does not work it uses hermitian matrix H, and if this also does not work it uses a combination of eigenvectors from K and H. We write a code in Matlab after this algorithm. Further, we compare our algorithm with the algorithms from the other two papers on every possible example ...

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 naš problem iskanja izotropnih vektorjev ter vse pojme, ki nam bodo v nadaljnem 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 in ga primerjali s še dvema algoritmoma. 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

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

pravimo mu *izotropni vektor*.
Bolj splošen je problem *inverznega numeričnega zaklada*, kjer iščemo enotski vektor b, za katerega velja:

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

kjer je μ dano kompleksno število.
Očitno je, da je problem (2) možno prevesti na problem (1) za drugo matriko, saj je (2) ekvivalentno

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

Zato bomo od sedaj naprej vse vrednosti μ enačili z 0.
Če je μ lastna vrednost matrike A, torej velja $Av = \mu v$, kjer je v lastni vektor, ki pripada μ, potem je rešitev problema inverznega numeričnega zaklada vektor v. Če pa μ ni lastna vrednost matrike A, je $A-\mu I$ nesingularna in moramo izotropni vektor izračunati. Tudi če matrika A ni realna imamo opravka s kompleksno matriko, ko je μ kompleksno število.

Izrek 1.1. [7] *Naj imata A in b realne ali kompleksne elemente. Potem veljajo enakosti:*

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

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 seštevkom 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

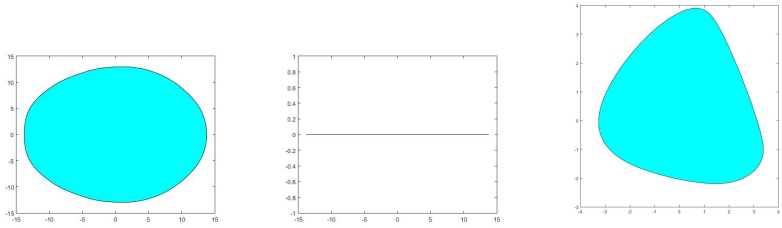
$$H = \frac{A + A^*}{2} \quad \text{in} \quad K = \frac{A - A^*}{2\imath}.$$

1.2. **Numerični zaklad.**

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

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

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. ([8])



Slika 1. Numerični zaklad realne, hermitske in kompleksne matrike, narisan 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 ([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 $\forall z \in \mathbb{C}$.
(v) Množenje s skalarjem: $W(zA) = zW(A)$ za $\forall z \in \mathbb{C}$.
(vi) Subaditivnost: Za $\forall A, B \in \mathbb{C}^{n\times n}$ velja $W(A + B) \subseteq W(A) + W(B)$.
(vii) Projekcija: $W(H) = \Re(W(A))$, za $\forall A \in \mathbb{C}^{n\times n}$, kjer s H označimo hermitsko matriko.
(viii) Če je A normalna, potem $W(A) = \mathrm{Co}(\sigma(A))$, kjer s 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 pokazali v naslednjem poglavju.
- Kompaktnost: Množica $W(A)$ je zaloga vrednosti zvezne funkcije $x \to x^*Ax$ na definicijskem območju $\{x : x \in \mathbb{C}^n,\ x^*x = 1\}$ (površina Euklidske enotske krogle), 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 neničeln vektor $x \in \mathbb{C}^n$ (za katerga lahko predpostavimo, da je enotski), za katerega velja $Ax = \lambda x$

in zato je

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

.

(iii) Zaradi unitarne transformacije postane površina enotske Evklidske krogle invariantna in 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

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

(v) Imamo

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

(vi) Imamo

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

(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, potem je $A = U^*\Lambda U$, kjer je $\Lambda = \operatorname{diag}\{\lambda_1, \ldots, \lambda_n\}$ diagonalna in U unitarna. Po lastnosti (3) je $W(A) = W(\Lambda)$ in ker

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

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

(ix) Ker je hermitska matrika tudi normalna velja

$$W(H) = \mathrm{Co}(\sigma(H)),$$

zaradi lastnosti (3). Vemo, da se numerični zaklad hermitske matrike projecira na realno os ter da je $\sigma(H)$ množica vseh realnih lastnih vrednosti matrike H. Konveksna ogrinjača $\sigma(H)$ je množica vseh daljic med lastnimi vrednostmi matrike H in ker bo numerični zaklad samo na realni osi, je to daljica med najmanjšo in največjo realno lastno vrednostjo H.

□

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

Lema 1.3. [8] *Naj bo $H \in \mathbb{C}^{n\times n}$ hermitska matrika in $\mu \in W(H)$. Potem je množica $\mathcal{L}_H(\mu) = \{x \in \mathbb{C}^n : x^*x = 1,\ x^*Hx = \mu\}$ povezana s potmi (t.j. obstaja pot med poljubnima dvema točkama iz te množice)*

Dokaz. Zaradi lastnosti (3) in (4) lahko brez škode za splošnost predpostavimo, da je $\mu = 0$ in $H = \mathrm{diag}\{d_1, d_2, \ldots, d_n\}$ realna diagonalna matrika. Potem je

$$W(H) = \Big\{ \sum_{j=1}^{n} d_j |x_j|^2 \colon x_1, x_2, \ldots, x_n \in \mathbb{C},\ \sum_{j=1}^{n} |x_j|^2 = 1 \Big\}$$

Naj bo sedaj $x = (x_j), y = (y_j) \in \mathcal{L}_H(0)$ 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

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

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

$$[r_1 \quad r_2 \quad \ldots \quad r_n]^T \in \mathcal{L}_H(0)$$

z zvezno krivuljo

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

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

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

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

Seveda, zgornja lema drži za vsako skalarno množenje hermitske matrike. Še več, za realno diagonalno matriko $H = \mathrm{diag}\{d_1, d_2, \ldots, d_n\}$, $d_1 \geq d_2 \geq \ldots \geq d_n$, vrednost $\sum_{j=1}^{n} d_j |x_j|^2$ (kjer je $x_1, x_2, \ldots, x_n \in \mathbb{C}$, $\sum_{j=1}^{n} |x_j|^2 = 1$) doseže maksimum

d_1, pri $|x_1|= 1$ in $x_2 = x_3 = \cdots = x_n = 0$ in doseže minimum d_n, pri $|x_n|= 1$ in $x_1 = x_2 = \cdots = x_{n-1} = 0$. Velja naslednja posledica.

Posledica 1.4. [8] *Naj bo $H \in \mathbb{C}^{n\times n}$ hermitska matrika in μ 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 μ.*

Izrek 1.5 (*Toeplitz-Hausdorfferjev izrek*)**.** [8] *Za $\forall A \in \mathbb{C}^{n\times n}$ je $W(A)$ konveksna.*

Dokaz. Dokazati moramo, da če vzamemo poljubni točki $\mu, \nu \in W(A)$, mora daljica s krajiščema μ in ν ležati v $W(A)$. Zaradi lastnosti (4) 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 jih definirali v prejšnjem podpoglavju. 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 $z(t) : [0, 1] \to \mathcal{L}_K(0)$, da sta $z(0) = x$ in $z(1) = y$. Sledi, da je funkcija

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

realna in zvezna glede na spremenljivko t in zadošča pogojema

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

in

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

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

1.3. **Uporaba.** Zanimanje za izračun izotropnih vektorjev je povezano s preučevanjem delne stagnacije GMRES algoritma 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:

$$b^*Hb = 0, \tag{3}$$

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

Lema 2.1. [4] *Izotropni vektorji realne matrike A so identični izotropnim vektorjem njenega simetričnega dela.*

Dokaz. To sledi iz $b^TAb = b^TA_{sim}b + b^TA_{psim}b = b^TA_{sim}b$, 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. □

Velja enakost:

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

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 λ_n najmanjša in λ_1 največja lastna vrednost matrike H. Naj bosta x_1 in x_n realna lastna vektorja, pripadajoča λ_1 in λ_n. Potem sta $x_1^T Ax_1 = x_1^T Hx_1 = \lambda_1$ in $x_n^T Ax_n = x_n^T Hx_n = \lambda_n$ 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 Λ matrika, ki ima na diagonali lastne vrednosti λ_i, ki so realna števila. X je ortogonalna matrika lastnih vektorjev, tako da $X^T X = I$. Potem uporabimo ta spektralni razcep v (3):

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

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

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

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

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

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

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

Ker je Λ diagonalna matrika, lahko $c^*\Lambda c$ zapišemo kot vsoto komponent $\bar{c_i}\lambda_i c_i = \lambda_i |c_i|^2$, za $i = 1, 2, ...n$. 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.

□

Opomba 2.3. Enačbi veljata samo za realna števila. Zaradi izreka 2.2 mora biti 0 konveksna kombinacija lastnih vrednosti λ_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 če imamo tri lastne vrednosti in nedefinitno matriko A.

Če niso vse lastne vrednosti H enako predznačene, potem mora za najmanjšo lastno vrednost λ_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:

$$t_s = \frac{\lambda_k}{\lambda_k - \lambda_n}. \tag{6}$$

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 λ_n in λ_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

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

Konstruirani rešitvi sta neodvisni in še več, ortogonalni, če $\lambda_k = -\lambda_n$.
Predpostavimo, da je b realen.

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

Dokaz.

$$b_1^T b_2 = (\sqrt{\lambda_k} x_n + \sqrt{|\lambda_n|} x_k)^T (-\sqrt{\lambda_k} x_1 + \sqrt{|\lambda_n|} x_k) = -(\lambda_n + \lambda_k).$$

□

Za ta postopek lahko uporabimo vsak par pozitivnih in negativnih lastnih vrednosti, uporaba najmanjše lastne vrednosti λ_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.
Ko sta A in b realna smo dokazali naslednji izrek:

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

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)

$$\lambda_1 t_1 + \lambda_2 t_2 + \lambda_3 (1 - t_1 - t_2) = (\lambda_1 - \lambda_3) t_1 + (\lambda_2 - \lambda_3) t_2 + \lambda_3 = 0 \tag{7}$$

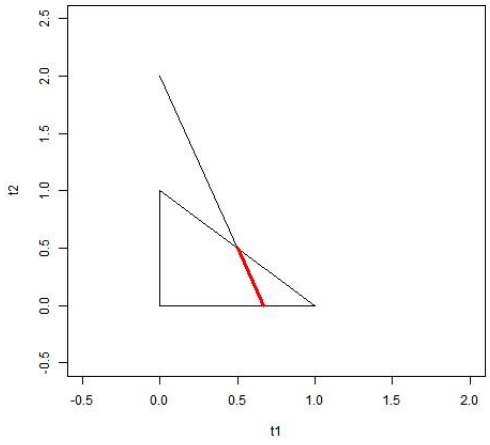
s pogoji: $t_i \geq 0, i = 1, 2$ in $t_1 + t_2 \leq 1$. Torej velja zveza

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

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 matriko 3×3 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 daljici, pobarvani rdeče, v trikotniku omejitev.

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 $c = X^T b$ pa 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] *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:

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

Prva enačba opisuje hiperravnino v kateri moramo poiskati presečišča te hiperravnine z volumnom telesa definiranega 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 v [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],[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 $\alpha_1 = \Re(b_1^* A b_1) < 0$ in $\alpha_2 = \Re(b_2^* A b_2) > 0$, smo končali.([7])

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

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

$$\begin{aligned}
b(t,\theta)^* A b(t,\theta) &= (e^{-i\theta} b_1 + t b_2)^* A (e^{-i\theta} b_1 + t b_2) \\
&= (b_1^* (\overline{e^{-i\theta}})^T + b_2^* t) A (e^{-i\theta} b_1 + t b_2) \\
&= (b_1^* (e^{\overline{-i\theta}})^T + b_2^* t) A (e^{-i\theta} b_1 + t b_2) \\
&= (b_1^* (e^{i\theta})^T A + b_2^* t A)(e^{-i\theta} b_1 + t b_2) \\
&= b_1^* e^{i\theta} A e^{-i\theta} b_1 + b_1^* e^{i\theta} A t b_2 + b_2^* t A e^{-i\theta} b_1 + b_2^* t A t b_2 \\
&= b_1^* A b_1 + e^{i\theta} b_1^* A b_2 t + e^{-i\theta} b_2^* A b_1 t + t^2 b_2^* A b_2
\end{aligned}$$

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

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

To bo držalo, ko bo

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

□

Algoritem:

1. S kombiniranjem lastnih vektorjev K, pripadajočim pozitivnim in negativinim lastnim vrednostim, izračunamo vektorja b_1 in b_2, taka da $\alpha_1 = \Re(b_1^* A b_1) < 0$ in $\alpha_2 = \Re(b_2^* A b_2) > 0$. 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 $\imath A$.
3. Če postopek ne deluje niti za $\imath A$, uporabimo kombinacijo lastnih vektorjev K in H, kjer z x označimo lastni vektor K in z y lastni vektor H.
4. Upoštevamo vektorje $X_\theta = \cos(\theta)x + \sin(\theta)y$, $0 \leq \theta \leq \pi$. $X_\theta^* A X_\theta$ opiše elipso znotraj numeričnega zaklada.

5. Za dan par (x, y) iščemo presečišča elipse $X_\theta^* A X_\theta$ z realno osjo. Z upoštevanjem, da je $A = H + iK$, računamo:

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

   Naj bo $\alpha = \Im(x^* H x + i x^* K x), \beta = \Im(y^* h y + i y^* K y)$ in $\gamma = \Im(x^* H y + y^* H x + i[x^* K y + y^* K x])$. Ko izenačimo imaginarni del $X_\theta^* A X\theta$ z 0, dobimo enačbo:

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

   Predpostavimo, da $\cos(\theta) \neq 0$ 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 θ, ki nam vrnejo take vektorje X_θ, da $\Im(X_\theta^* A X_\theta) = 0$.
7. Če tudi ta konstrukcija ne deluje, uporabimo algoritem iz [3], opisan čez dva razdelka.

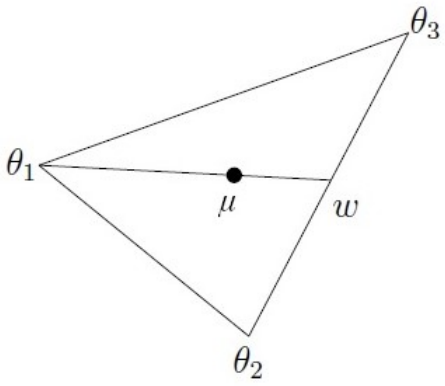
Opomba 3.2. Ko je A realna in imamo $b_1 = x_1, b_2 = x_2$ za lastne vektorje H, potem je $\theta = 0$ in lema 3.1 pove, kako se izračuna en izotropni vektor. Vendar v kompleksnem primeru ne moremo vedno najti primernih vektorjev b_1 in b_2. Konstrukcija ne deluje, če imajo vrednosti $\Re(y_i^* H y_j)$ enak predznak, kjer so y_i lastni vektorji K. Ekstremen primer je Jordanski blok s kompleksno vrednostjo α na diagonali in elementi 1 na naddiagonali. Potem je $\Re(y_i^* H y_j) = 0$, ko $i \neq j$, in $y_i^* H y_i = -\Re(\alpha)$. Zato so realni deli $b^* A b$ za vse vektorje b, ki se lahko konstruirajo, enaki.([7])

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 par 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 μ v konveksni ogrinjači treh točk $\theta_i \in W(A)$, za katere smo lahko izračunali izotropne vektorje b_i. Konveksna ogrinjača teh treh točk θ_i je trikotnik (lahko je izrojen). Radi bi, da je μ na daljici, ki ima take robne točke, da za njih vemo ali lahko izračuamo izotropne vektorje. Zato lahko brez škode za splošnost predpostavimo, da je θ_1 ena od robnih točk te daljice. Za drugo robno točko vzamemo w, ki je presečišče daljice med θ_2 in θ_3 s premico, ki teče skozi θ_1 in μ. Ker je w konveksna kombinacija θ_2 in θ_3, mu lahko določimo pripadajoč izotropni vektor. ([7]) Ker pa je μ konveksna kombinacija w in θ_1, lahko tudi njemu določimo izotropni vektor, 3. ([1])

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

1. Poiščemo zunanjo aproksimacijo $W(A)$ (je pravokotnik, katerega stranice so vzporedne z realno in imaginarno osjo), tako da zračunamo najbolj levo in najbolj desno lastno vrednost $H_\theta = (e^{i\theta} A + e^{-i\theta} A^*)/2$ za $\theta = 0, \pi/2$. Če μ



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

ni v zunanji aproksimaciji, potem $\mu \notin W(A)$ in ustavimo algoritem, drugače nadaljujemo.

2. Če je višina ali širina zunanje aproksimacije manj kot ε, potem je $W(A)$ približno hermitska ali poševno-hermitska (ali kompleksen premik katere od teh) in je numerični zaklad približno točka. V obeh primerih ugotovimo ali je $\mu \in W(A)$. Če je, poiščemo pripadajoč izotropni vektor.
3. Če $\mu \notin W(A)$, nadaljujemo s konstrukcijo notranje aproksimacije $W(A)$ (je štirikotnik z oglišči v robnih točkah $\partial W(A)$) z uporabo lastnih vektorjev najbolj leve in desne lastne vrednosti H_θ.
4. Če μ leži v notranji aproksimaciji, lahko poiščemo izotropni vektor, ki generira μ. Uporabimo postopek, ki je bil opisan v začetku tega razdelka. Če μ ne leži v notranji aproksimaciji, določimo katera stranica notranje aproksimacije mu leži najbližje.
5. Izračunamo $\hat{\mu}$, ki je najbližja točka do μ, ki leži na notranji aproksimaciji. Če je $|\hat{\mu} - \mu| < \varepsilon$, izračunamo izotropni vektor za $\hat{\mu}$ in ga sprejmemo kot izotropni vektor za μ ter ustavimo algoritem.
6. Posodobimo notranjo in zunanjo aproksimacijo z izračunom največje lastne vrednosti in pripadajočega lastnega vektorja H_θ, kjer je smer θ pravokotna na stranico notranje aproksimacije, ki je najbližja μ. Če ne dobimo nove robne točke, ki se ni dotikala notranje aproksimacije, potem $\mu \notin W(A)$.
7. Ponovno preverimo, če je μ v novi zunanji aproksimaciji. Če je, se vrnemo na 4. korak, drugače $\mu \notin W(A)$.

Opomba 3.4. Korake 4.-7. ponavljamo, dokler ni notranja aproksimacija ε blizu μ ali dokler zunanja aproksimacija ne vsebuje μ. 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 μ ($\in W(A)$ ali $\notin W(A)$) leži zelo blizu roba numeričnega zaklada $\partial W(A)$ ([3]). Ta algoritem uporabi za iskanje izotropnih vektorjev le nekaj najbolj osnovnih lastnosti

numeričnega zaklada poleg konveksnosti, kot sta (4) in (5).

Algoritem:

1. Izračunamo do 4 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 4 točke p_i, ki označujejo ekstremne vrednosti $W(A)$, tj. najmanjši in največji horizontalni in vetikalni razteg. Označimo jih z rM in rm za maksimalen in minimalen horizontalni razteg $W(A)$ in z iM in im za maksimalen in minimalen vertikalen 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 od 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

$$(9)\quad \begin{aligned}(tx + (1-t)y)^* A (tx + (1-t)y) = (x^*Ax + y^*Ay - (x^*Ay + y^*Ax))t^2 \\ + (-2y^*Ay + (x^*Ay + y^*Ax))t + y^*Ay.\end{aligned}$$

   Njene ničle določijo koordinatne osi $W(A)$ točk na elipsah skozi točke x^*Ax, $y^*Ay \in \partial W(A)$ 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 polinomsko enačbo z realnimi koeficienti:

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

   za $q = \Im(x^*Ax)$, $p = \Im(y^*Ay)$ in $r = \Im(x^*Ay + y^*Ax)$. Označimo $f = p + q - r$ in $g = (r - 2p)/f$.
8. Enačba (10) ima realni rešitvi t_i, $i = 1, 2$, ki vrneta 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 realno os na vsaki strani 0, potem preverimo, če to stori njihova skupna množica in ponovimo isti postopek.
10. Če niti skupna množica ne gre, 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], vendar pa 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, najden 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 2 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 matrike 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 3 največjim in 3 najmanjšim lastnim vrednostim. Edina razlika med našim algoritmom in algoritmom iz 3.1.2 nastane zaradi premalo informacij iz članka [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 funckija `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_θ definiramo kot

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

kjer je

$$\varphi = \frac{1}{2\imath}\ln(x^*Ay(y^*Ax)^{-1}).$$

Če maksimiziramo elipso je večja verjetnost, da bo vsebovala izhodišče in s tem je večja verjetnost, da najdemo rešitev v manj korakih. Če najdemo primerna b_1 in b_2 da lahko uporabimo lemo 3.1, ki je definirana s funkcijo `lema31`, smo končali.

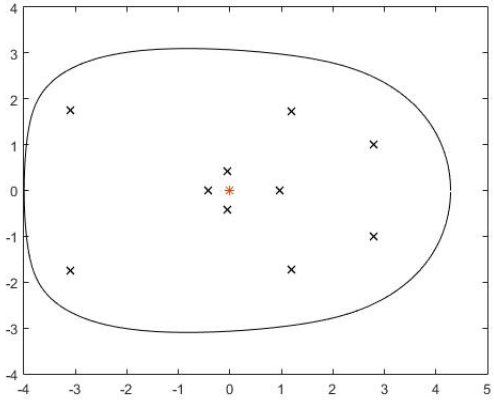
4.2. **Primerjava.** Označimo algoritme z `AlgM`, `AlgC` in `AlgCPU`. Numerične zaklade bomo narisali s funkcijo iz [5], ki nariše tudi lastne vrednosti matrike A s križci. Za vsak algoritem bomo preverili v kolikšnem času je našel (ali ni našel) rešitev, koliko izračunov lastnih vrednosti in vektorjev je potreboval (označimo kot "Koraki") ter kako velika je napaka $|b^*Ab|$.

4.2.1. *A in μ realna.* Za prvi primer vzamimo naključno generirano 10×10 realno matriko A in $\mu = 1,0419$, ki je znotraj numeričnega zaklada, Slika 4.

```
A = randn(10)
```

Tabela 1. A realna, $\mu = 1,0419$ in obstaja rešitev

Algoritem	Čas	Koraki	Napaka
`AlgM`	0.024636	1	1.4988e-15
`AlgC`	0.059675	4	5.2369e-16
`AlgCPU`	0.242559	2	7.0217e-16



Slika 4. Numerični zaklad premaknjene matrike A in z rdečo $*$ označeno izhodišče.

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

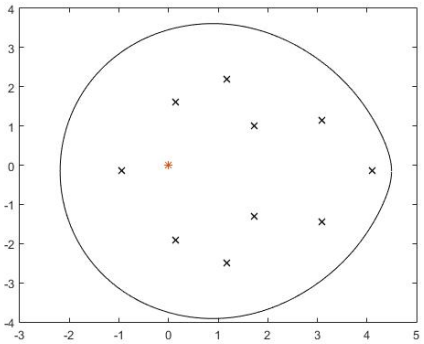
$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}.$$

Matrika A je definitna, zato algoritem ne najde rešitve.

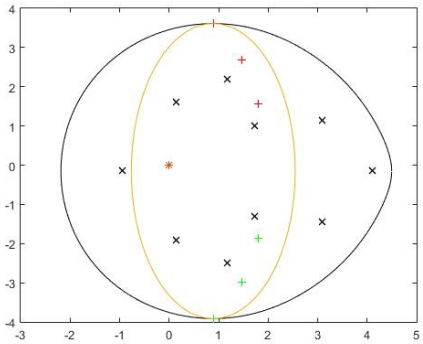
Tabela 2. A realna in ne obstaja rešitev.

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

4.2.2. *A realna, μ kompleksen.* Naj bo A naključno generirana realna 10×10 matrika in $\mu = -1,4575 + 0,1514\imath$ kompleksno število iz numeričnega zaklada, Slika 5A.



(A) Z rdečo $*$ označimo izhodišče.



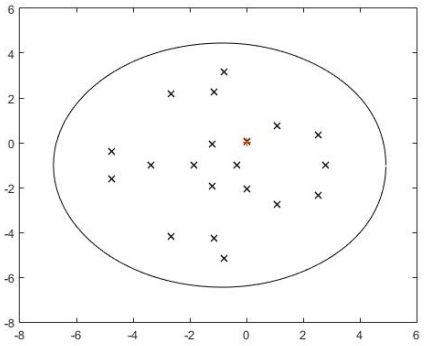
(B) Rdeči + so največji lastni vektorji, zeleni pa najmanjši lastni vektorji matrike K.

Slika 5. Numerični zaklad premaknjene matrike A in kako grafično najdemo rešitev v 1 koraku.

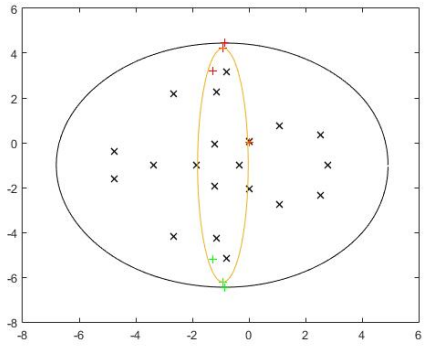
TABELA 3. A realen, $\mu = -1,4575 + 0,1514\imath$ in rešitev najdemo v 1 koraku.

Algoritem	Čas	Koraki	Napaka
`AlgM`	0.217679	1	3.8912e-16
`AlgC`	0.059066	4	1.1322e-15
`AlgCPU`	0.032536	2	3.1525e-16

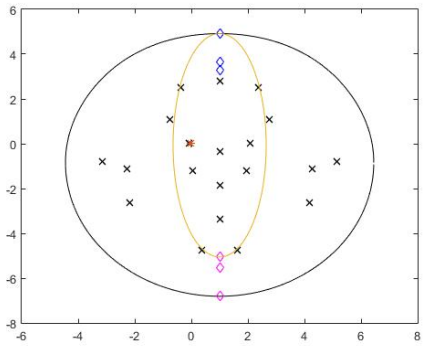
Za drugi primer vzamemo realno naključno generirano 20×20 matriko A in $\mu = 1 + \imath$, Slika 6A.



(A) Z rdečo $*$ označimo izhodišče.



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



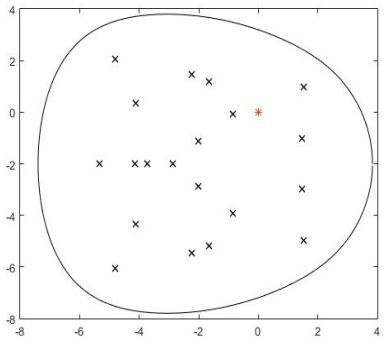
(C) Modri $\diamond$ so največji lastni vektorji, roza pa najmanjši lastni vektorji matrike H.[1] Rišemo elipse za matriko $\imath A$.

SLIKA 6. Numerični zaklad premaknjene matrike A in kako grafično najdemo rešitev v 2 korakih.

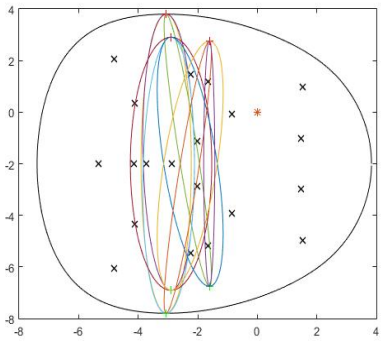
TABELA 4. A realen, $\mu = 1 + \imath$ in rešitev najdemo v 2 korakih.

Algoritem	Čas	Koraki	Napaka
`AlgM`	0.123142	2	6.2804e-16
`AlgC`	0.017274	4	1.1102e-15
`AlgCPU`	0.215924	2	5.5511e-17

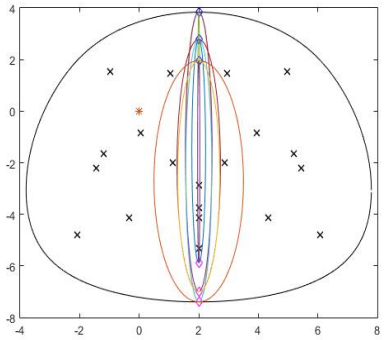
Za tretji primer vzamemo realno naključno generirano 20×20 matriko A in $\mu = 2 + 2\imath$, Slika 7A.



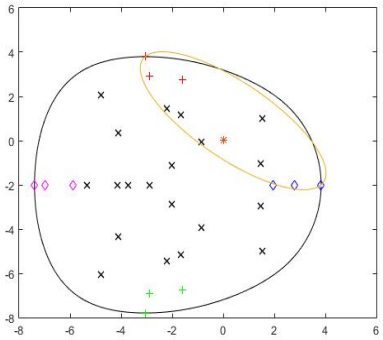
(A) Z rdečo $*$ označimo izhodišče.



(B) Rdeči + so največji lastni vektorji, zeleni pa najmanjši lastni vektorji matrike K.[1] 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.[1] Rišemo elipse za matriko $\imath A$. Z nobeno kombinacijo ne dobimo dovolj velike elipse, ki bi vsebovala izhodišče.



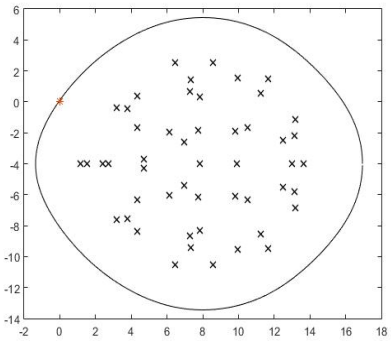
(D) Gledamo vse kombinacije lastnih vektorjev K in H na matriki A.[1]

SLIKA 7. Numerični zaklad premaknjene matrike A in kako grafično najdemo rešitev v 3 korakih.

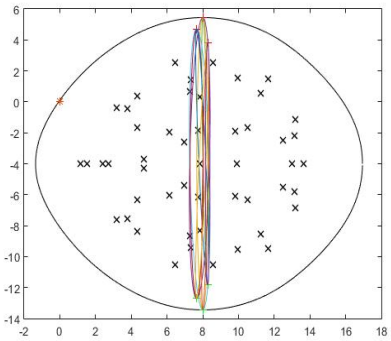
TABELA 5. A realen, $\mu = 2 + 2\imath$ in rešitev najdemo v 3 korakih.

Algoritem	Čas	Koraki	Napaka
`AlgM`	0.033775	2	3.7752e-16
`AlgC`	0.020296	4	9.9301e-16
`AlgCPU`	0.012049	2	4.9651e-16

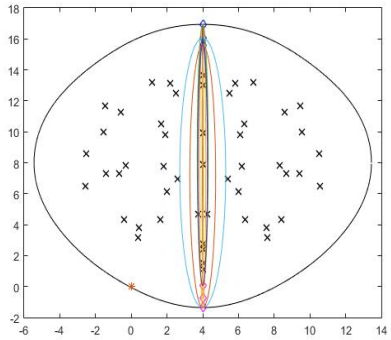
Poglejmo še en primer, kjer algoritem ne deluje za realno naključno generirano 50×50 matriko A in $\mu = -8 + 4\imath$, ki je na robu numeričnega zaklada, Slika 8A.



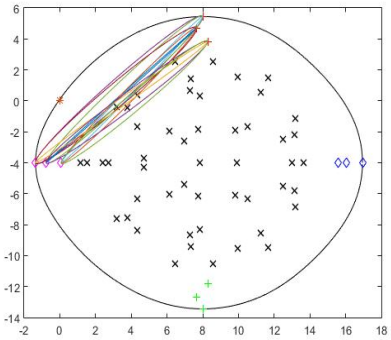
(A) Z rdečo $*$ označimo izhodišče.



(B) Rdeči + so največji lastni vektorji, zeleni pa najmanjši lastni vektorji matrike K.[1] 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.[1] Rišemo elipse za matriko $\imath A$. Z nobeno kombinacijo ne dobimo dovolj velike elipse, ki bi vsebovala izhodišče.



(D) Gledamo vse kombinacije lastnih vektorjev K in H na matriki A.[1] Narisane so le tiste elipse, ki imajo možnost vsebovat izhodišče.

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

TABELA 6. A realen, $\mu = -8 + 4\imath$ in naš algoritem ne najde rešitve.

Algoritem	Čas	Koraki	Napaka
`AlgM`	0.068600	2	Inf
`AlgC`	0.066789	4	Inf
`AlgCPU`	0.164600	1	Inf

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

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

Za prvi primer vzamimo $\mu = 5000 + 10000\imath$, Slika 9.

TABELA 7. A in $\mu = 5000 + 10000\imath$ kompleksna in algoritem najde rešitev v 1 koraku.

Algoritem	Čas	Koraki	Napaka
`AlgM`	0.023934	1	1.0168e-12
`AlgC`	0.680576	5	1.0914e-11
`AlgCPU`	0.03086	4	1.3903e-12

Za drugi primer vzamemo $\mu = 12000 + 10000\imath$, ki je bližje robu, Slika 9.

TABELA 8. A kompleksna, $\mu = 12000 + 10000\imath$ in algoritem najde rešiev v 2 korakih.

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 $\mu = 12500 + 10000\imath$, ki je še bližje robu kot v prejšnjem primeru, Slika 9.

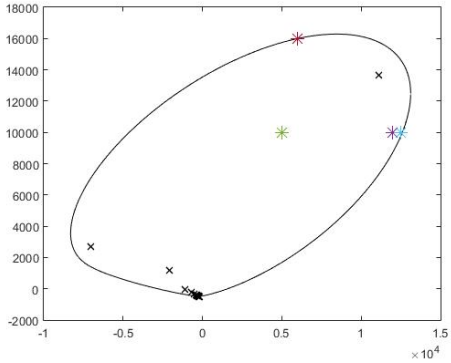
TABELA 9. A kompleksna, $\mu = 12500 + 10000\imath$ in algoritem najde rešitev v 3 korakih.

Algoritem	Čas	Koraki	Napaka
`AlgM`	0.586810	2	1.3074e-12
`AlgC`	0.481654	8	1.4665e-11
`AlgCPU`	0.038650	5	6.5169e-13

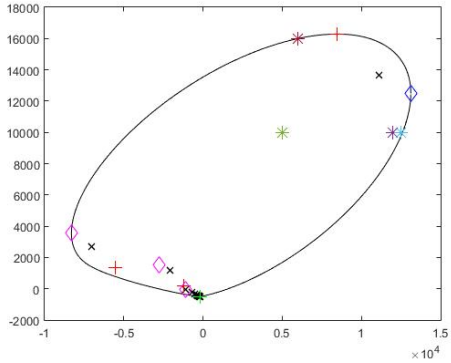
Za zadnji primer vzamimo $\mu = 6000 + 16000\imath$, ki je blizu roba, ampak vseeno ni v numeričnem zakladu, Slika 9.

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

Algoritem	Čas	Koraki	Napaka
`AlgM`	0.456248	2	Inf
`AlgC`	0.485319	11	Inf
`AlgCPU`	0.045177	10	Inf



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



(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.[1]

SLIKA 9. 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 μ.

5. ZAKLJUČEK

[1]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.

```
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 l. 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

H=(A+A')/2;
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)];
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);
            %#ok<*MINV>
            [b1, b2] = xtheta(X(:,k)*exp(fi*1i), 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


%ponovimo isti postopek za H in matriko iA
[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)];

for k=1: size(XX,2)-1,
    for j = (k+1): size(XX,2),
        if (LV1(k,:)*LV1(j,:)< 0),
          fi=(log(XX(:,k)'*A*XX(:,j)*inv(XX(:,j)'*A*XX(:,k))))/(2i);
```

```
            %#ok<*MINV>
            [b1, b2] = xtheta(XX(:,k)*exp(fi*1i), XX(:,j), 1i*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'*(1i*A)*b1))<1e-10) &&
                    (abs(imag(b2'*(1i*A)*b2))<1e-10),
                    b = lema_31(b1,b2,(1i*A));
                    if sum(abs(b)<ones(n,1)*1e-10)==0,
                        napaka = abs(b'*(1i*A)*b);
                        return
                    end
                end
            end
        end
    end
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);
            %#ok<*MINV>
            [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 iztotropnih 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 + gama*t + alfa = 0
t1 = (-gama + sqrt(gama^2 -4*beta*alfa))/(2*beta);
t2 = (-gama - 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)~=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
end

function [b] = lema_31(b1,b2,A)
%Vhod: enotska vektorja b1 in b2, za katera velja imag(bi'*A*bi)=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
```

Slovar strokovnih izrazov

convex hull konveksna ogrinjača
eigenanalysis izračun lastnih vrednosti in vektorjev
field of values/numerical range numerični zaklad
inverse field of values problem/inverse numerical range problem problem inverznega numeričnega zaklada
isotropic vector/generating vector izotropni vektor
origin izhodišče, točka $(0, 0)$
skew-Hermitian matrix poševno-hermitska matrika
tetrahedron tetraeder

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/`.

[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*, In: 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=`.

[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)

[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`.