Cvičení 7 - zadání a řešení úloh

Základy numerické matematiky - NMNM201 Verze z 13. listopadu 2024

1 Rayleighův podíl (Rayleigh quotient)

Pro hermitovskou matici $A \in \mathbb{C}^{n\times n}$ a nenulový vektor $x \in \mathbb{C}^n$ definujeme Rayleighův podíl

$$R(A,x) = \frac{x^*Ax}{x^*x}.$$

Úloha 1. *Nechť $A \in \mathbb{C}^{n\times n}$ je hermitovská. Ukažte následující vlastnosti:*

(i) $R(\alpha A, \beta x) = \alpha R(A,x)$ *pro každé* $\alpha,\ \beta \in \mathbb{C},\ \beta \neq 0$.

(ii) $R(A-\alpha I, x) = R(A,x) - \alpha$ *pro každé* $\alpha \in \mathbb{C}$.

(iii) Nechť $Av = \lambda v$ *pro nějaký nenulový* $v \in \mathbb{C}^n$. *Potom* $R(A,v) = \lambda$.

(iv) Nechť $\lambda_{\min}$ *a* $\lambda_{\max}$ *jsou nejmenší, respektive největší vlastní číslo matice* A. *Ukažte, že*

$$R(A,x) \in [\lambda_{\min}, \lambda_{\max}] \qquad \forall x \in \mathbb{C}^n.$$

Proč požadujeme, aby matice A byla hermitovská (a nestačí například diagonalizovatelná)?

(v) Nechť jsou vlastní čísla A seřazena sestupně a největší je jednonásobné, tedy

$$\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n.$$

Nechť v_1 je vlastní vektor příslušný λ_1. Ukažte, že

$$R(A,x) \in [\lambda_n, \lambda_2] \qquad \forall x \in v_1^{\perp}.$$

(vi) $R(A,x) = \arg\min_{\mu\in\mathbb{R}} \|(A-\mu I)x\|^2$.

Řešení.

(i) $R(\alpha A, \beta x) = \dfrac{(\beta x)^*\alpha A(\beta x)}{(\beta x)^*\beta x} = \dfrac{\bar\beta\beta(x^*\alpha A x)}{\bar\beta\beta x^*x} = \dfrac{\alpha(x^*Ax)}{x^*x} = \alpha R(A,x).$

(ii) $R(A-\alpha I, x) = \dfrac{x^*(A-\alpha I)x}{x^*x} = \dfrac{x^*Ax - x^*\alpha I x}{x^*x} = \dfrac{x^*Ax}{x^*x} - \alpha\dfrac{x^*x}{x^*x} = R(A,x) - \alpha.$

(iii) $R(A,v) = \dfrac{v^*Av}{v^*v} = \dfrac{v^*\lambda v}{v^*v} = \lambda\dfrac{v^*v}{v^*v} = \lambda.$

(iv) Nejprve pro obecnost uvažujme A unitárně diagonalizovatelnou, či ekvivalentně, normální. Nechť unitární U diagonalizuje A:

$$U^*AU = D = \operatorname{diag}(\lambda_1,\ldots,\lambda_n).$$

Libovolné $x \in \mathbb{C}^n$ můžeme psát ve tvaru $x = Uy$ s nějakým $y \in \mathbb{C}^n$ a tedy platí

$$R(A,x) = \frac{x^*Ax}{x^*x} = \frac{(Uy)^*A(Uy)}{(Uy)^*Uy} = \frac{y^*U^*AUy}{y^*\underbrace{U^*U}_{=I}y} = \frac{y^*Dy}{y^*y} = \frac{\sum_{i=1}^n \lambda_i|y_i|^2}{\|y\|^2}.$$

Zřejmě platí

$$\frac{|y_i|^2}{\|y\|^2} \in [0,1] \quad \text{pro všechna } i = 1,2,\ldots,n, \qquad \sum_{i=1}^n \frac{|y_i|^2}{\|y\|^2} = 1,$$

což ukazuje, že $R(A,x)$ je konvexní kombinací $\{\lambda_1,\ldots,\lambda_n\}$.

Pro Hermitovskou A je spektrum reálné a z výše uvedeného okamžitě plyne, že $R(A,x) \in [\lambda_{\min}, \lambda_{\max}]$.

(v) Jsou-li vlastní čísla očíslována, jak je požadováno, pak v předchozí notaci v_1 je první sloupec U a tedy $0 = v_1^* x = v_1^* U y = y_1$. Tedy $\frac{|y_1|^2}{\|y\|^2}$ ve výše uvedené konvexní kombinaci nepřispívá a tedy $R(A, x)$ je konvexní kombinací $\{\lambda_2, \ldots, \lambda_n\}$.

(vi) Máme minimalizovat funkci $g\colon \mathbb{R} \to \mathbb{R}$,

$$g(\mu) = \|(A - \mu I)x\|^2 = x^*(A - \mu I)^*(A - \mu I)x = \|Ax\|^2 - 2\mu x^* Ax + \mu^2 \|x\|^2$$
$$= \|x\|^2 \left(\mu - \frac{x^* Ax}{\|x\|^2}\right)^2 + \|Ax\|^2 - \frac{|x^* Ax|^2}{\|x\|^2},$$

což je kvadratická funkce, která skutečně nabývá minima pro $\mu = \frac{x^* Ax}{\|x\|^2} = R(A, x)$. □

2 Mocninná metoda

Ne vždy je potřeba (a někdy to není ani technicky možné) nalézt celé spektrum dané matice. Cílem mocninné metody je nalezení jednoho vlastního páru (vlastního čísla a vlastního vektoru) matice.

Nechť $A \in \mathbb{C}^{n \times n}$ je diagonalizovatelná matice (tj. má n lineárně nezávislých vlastních vektorů, které tvoří bázi prostoru $\mathbb{C}^n$),

$$A = SDS^{-1},$$

kde pro $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ jsou vlastní čísla seřazena sestupně, tj.

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|,$$

a matice $S = [s_1, \ldots, s_n]$ je tvořena příslušnými normalizovanými vlastními vektory.

Pro jednoduchost předpokládejme, že λ_1 je dominantní vlastní číslo, tj. $|\lambda_1| > |\lambda_2|$ a v je nenulový "startovací" (počáteční) vektor. Mocninnou metodu pak můžeme zapsat jako Algoritmus 1.

Algoritmus 1 Mocninná metoda

Vstup: A, v
  $v_0 = v/\|v\|$
  for $k = 1, \ldots$ **do**
    $w = Av_{k-1}$
    $v_k = w/\|w\|$
    $\mu_k = v_k^* A v_k$
  end for

Když chceme měřit rychlost konvergence, lze se dívat buď na *chybu aproximace vlastního vektoru* $\|s_1 - v_k\|$ nebo na *chybu aproximace vlastního čísla* $\|\lambda_1 - \mu_k\|$ nebo na *reziduum aproximace* $\|Av_k - \mu_k v_k\|$. V praxi ale nelze měřit ani jednu z chyb (nemáme k dispozici vlastní pár (λ_1, s_1), o tom to celé je), takže se musíme spokojit s reziduem. Ale pro naše zjednodušené a ilustrativní příklady se můžeme dívat jak na chybu tak na reziduum.

Úloha 2. *Naprogramujte mocninnou metodu na základě Algoritmu 1 do předpřipraveného skriptu* `power_method.m` *a poté spusťte* `power_method_test.m` *pro vstupní data*

$$A = \begin{bmatrix} 4 & & & \\ & 3 & & \\ & & 2 & \\ & & & 1 \end{bmatrix} \quad \text{a} \quad v = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}. \tag{1}$$

K čemu a jak rychle mocninná metoda konverguje? Odhadněte předpokládanou velikost chyby approximace vlastního vektoru $\|s_1 - v_k\|$ *po 70 iteracích na základě analýzy konergence mocninné metody z přednášky a porovnejte s výsledky v MATLABu.*

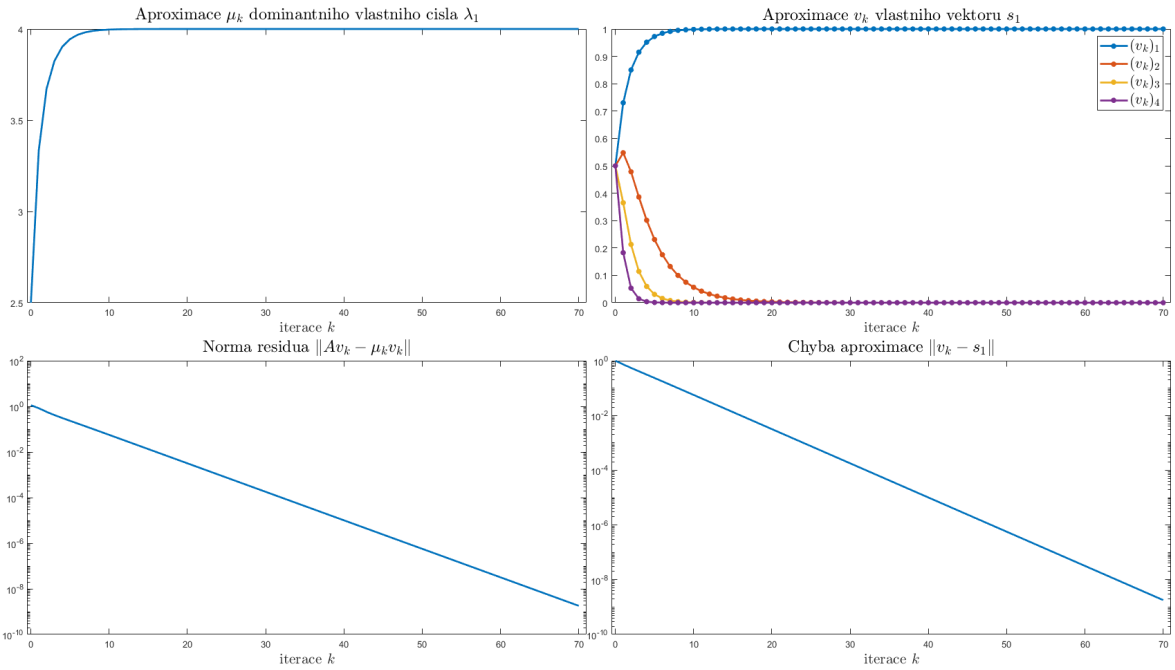
Řešení. Správný MATLAB kód vypadá následovně

```
function [mu,v,res_2norm] = power_method(A,v0,niter)

%%% inicializujeme si ulozny prostor pro nase vysledky
mu = nan(niter+1,1); % approximace nejvetsiho vl. cisla
v = nan(length(v0),niter+1); % approximace prislusneho vl. vektoru
res_2norm = nan(niter+1,1); % normy residua || Av_k - mu_k v_k ||

%%% prvni ulozena data odpovidaji hodnotam pocatecniho vektoru
v(:,1) = v0/norm(v0);
mu(1) = v(:,1)'*A*v(:,1);
res_2norm(1) = norm(A*v(:,1)-mu(1)*v(:,1));

%%% nasledne postupujeme jako u Algoritmu 1
for k = 2:niter+1
     w = A*v(:,k-1); % A*v_{k-1}
     v(:,k) = w/norm(w); % normalizace vektoru w
     mu(k) = v(:,k)' * A * v(:,k); % Rayleighuv podil
     res_2norm(k) = norm(A*v(:,k)-mu(k)*v(:,k)); % reziduum
end
end
```

Na přednášce jsme viděli, že máme-li A s dominantním vlastním párem (λ_1, s_1) a počátečním vektorem v_0, který není kolmý na s_1 (tj. $c_1 := s_1^T v_0 \neq 0$), pak by měla mocninná metoda konvergovat "asymptoticky lineárně"("lineární konvergenci"jste na analýze říkali geometrická konvergence, tj., posloupnost $\{a_k\}$ konverguje lineárně/geometricky pokud $a_k = q^k \cdot a_0$ pro nějaký konvergenční faktor q a pro všechny indexy k; "asymptoticky"pak znamená, že se ta posloupnost takto chová přinejhorším až od určitého k_0 dále) s konvergenčním faktorem $|\lambda_2/\lambda_1|$. V našem případě máme $|\lambda_2/\lambda_1| = 3/4$ a tedy po 70 iteracích můžeme "naivně"očekávat (tj. pokud by konvergence byla opravdu lineární/geometrická už od začátku), že chyba aproximace vlastního vektoru $\|v_k - s_1\|$ bude mít velikost $(3/4)^{70} \approx 1.8 \times 10^{-9}$.

Tak to opravdu v MATLABu vyjde a na grafu vidíme, že vyneseno v logaritmickém měřítku (tj. když se díváme na řád spíše než na přesnou hodnotu) chyba lineárně konverguje skutečně téměř od první iterace mocninné metody a po 70 iteracích opravdu dosáhne hodnoty přibližně $(3/4)^{70} \approx 1.8 \times 10^{-9}$ jak vidíme na Obrázku 1 níže. Podobně to funguje i pro spočítané reziduum.

Jako úkol navíc si zkuste rozmyslet, čím to, že konvergence byla lineární hned od začátku a jakou roli v tom hraje matice A a počáteční vektor v_0. □



Obrázek 1: Obrázek odpovídající správnému řešení úlohy 2.

Úloha 3. *V Úloze 2 naše aproximace v_k dominantního vlastního vektoru s_1 dosáhla chyby $\approx 1.8 \times 10^{-9}$ během prvních 70 iterací. Změňte matici A tak, aby byla chyba $\|s_1 - v_k\|$ na stejné úrovni již po 35 iteracích (se stejným počátečním vektorem $v_0 = [1, 1, 1, 1]^T$). [Hint]: Zkuste nejprve změnit jen druhý diagonální prvek A z (1). Pro pomocný výpočet se může hodit MATLAB funkce* `nthroot(x,n)`.

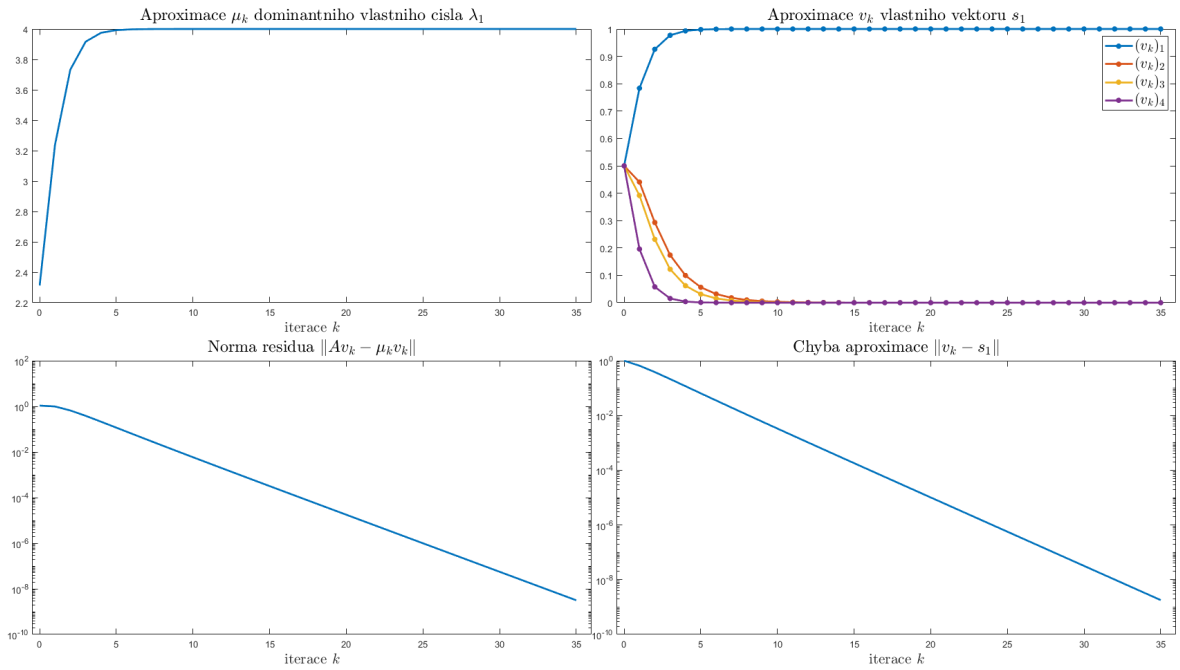
Řešení. V Úloze 2 jsme pozorovali lineární konvergenci s konvergenčním faktorem $|\lambda_2/\lambda_1| = 3/4$. Chceme-li dosáhnout stejné chyby za polovinu kroků, musí platit

$$|\lambda_2^{\text{new}}/\lambda_1^{\text{new}}|^{35} \approx 1.8 \times 10^{-9},$$

tedy například zachováme-li dominantní vlastní číslo $\lambda_1^{\text{new}} = \lambda_1 = 4$, pak nám vyjde, že

$$\lambda_2^{\text{new}} \approx \left(1.8 \times 10^{-9}\right)^{1/35} \cdot 4 \approx 2.21,$$

a tedy jedna možná volba matice A je $A = \text{diag}(4, 2.25, 2, 1)$, jak ukazujeme na Obrázku 2 níže. □



Obrázek 2: Obrázek odpovídající správnému řešení Úlohy 3.

Úloha 4. *Mějme pět diagonálních matic*

$$A_1 = \begin{bmatrix} 4 & & & \\ & 4 & & \\ & & 2 & \\ & & & 1 \end{bmatrix}, \; A_2 = \begin{bmatrix} 4 & & & \\ & 3.9 & & \\ & & 3.8 & \\ & & & 3.7 \end{bmatrix}, \; A_3 = \begin{bmatrix} -4 & & & \\ & 3 & & \\ & & 2 & \\ & & & 1 \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 4 & & & \\ & -3 & & \\ & & 2 & \\ & & & 1 \end{bmatrix}, \; A_5 = \begin{bmatrix} 4 & & & \\ & 3.9 & & \\ & & 2 & \\ & & & 1 \end{bmatrix},$$

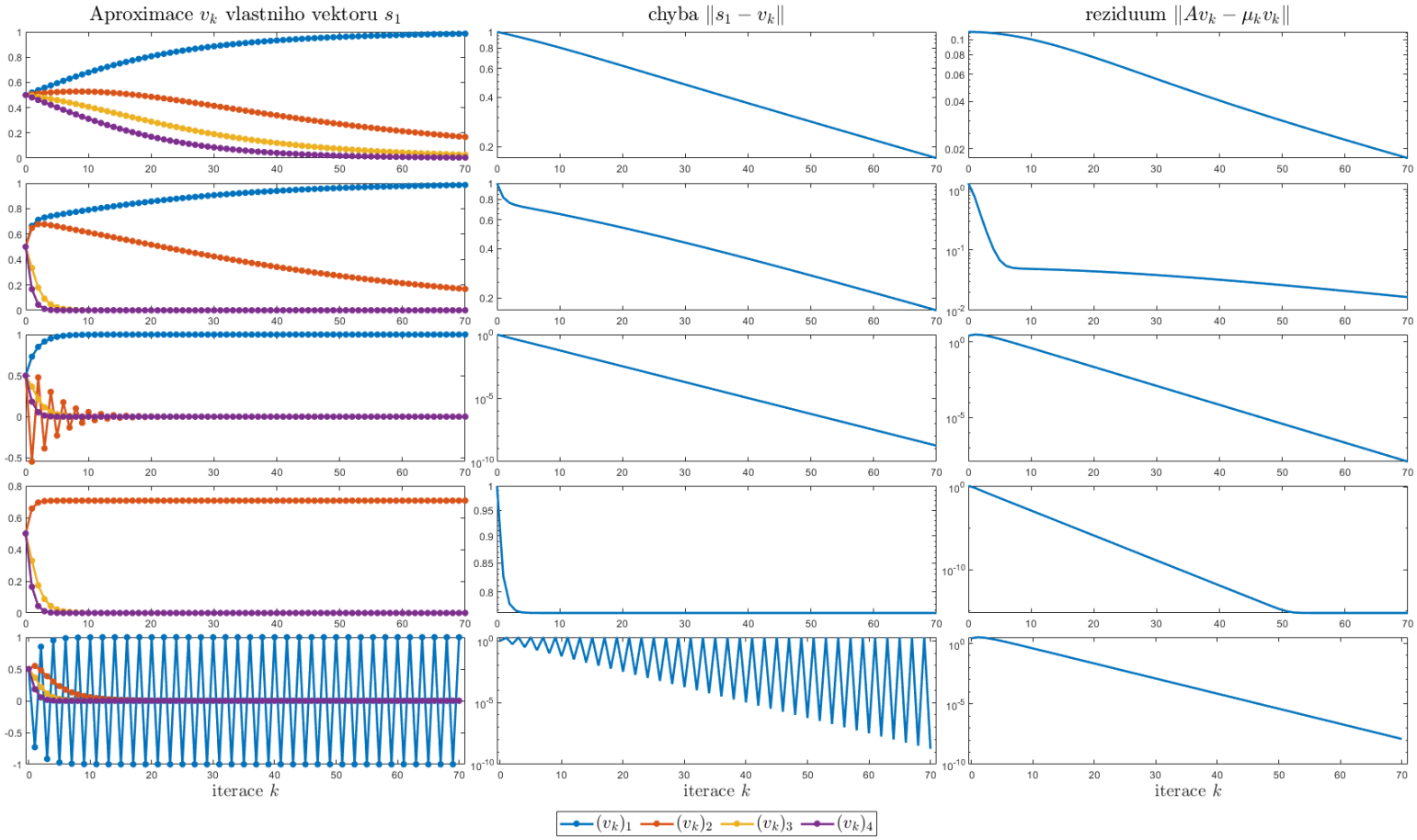
na které použijeme mocninou metodu (Algoritmus 1) s počátečním vektorem $v_0 = [1, 1, 1, 1]^T$. *Rozmyslete si jak budou vypadat grafy konvergence mocninné metody pro složky vektoru* $v_k \to s_1$ *a chyba* $\|s_1 - v_k\|$. *Na základě těchto úvah přiřaďte matice* $A_1, \ldots, A_5$ *k jednotlivým řádkům Obrázku 3.*

Řešení. Když se podíváme na levý sloupec Obrázku 3 všimneme si, že:

- u 3. a 5. řádku nám některé složky oscilují;
- u všech řádků až na 4. konvergují všechny složky k nule až na první;
- u všech řádků až na 1., 2. a 5. konvergují všechny složky vcelku rychle - ustálí se během prvních 20 iterací. Nadto, v pátém řádku to "kazí"pouze fakt, že $(v_k)_1$ nekonverguje, ale osciluje - tato oscilace je ale během několika iterací "stabilní"a kdybychom se dívali na absolutní hodnotu složek v_k, pak by to byly pouze řádky 1 a 2, které konvergují výrazně pomaleji.

S těmito pozorováními se obrátíme k maticím $A_1, \ldots, A_5$:

- matice A_3 a A_4 mají každá negativní prvek na diagonále. Zjevně negativní prvek na pozici $(1, 1)$ v matici A_3 může ovlivnit pouze první složku vektorů v_k (a naopak negativní prvek na pozici $(2, 2)$ v matici A_4 pouze druhou) a tedy s jistotou můžeme přiřadit A_3 k pátému řádku a A_4 ke třetímu



Obrázek 3: Aproximace dominantního vlastního vektoru s_1 pomocí mocninné metody pro pět různých diagonálních matic $A_1, \ldots, A_5$ s použitím počátečního vektoru $v_0 = [1, 1, 1, 1]^T$. Ve čtvrtém řádku v prvním sloupci jsou složky $(v_k)_1$ a $(v_k)_2$ prakticky totožné, takže se na grafu překrývají a jsou vidět pouze $(v_k)_2$.

řádku (Tento argument by v obecném případě nediagonálních matic pracoval s projekcemi v_k na vektory s_i – vzpomeneme si na rozepsání vektoru $A^k v_0$ do báze vlastních vektorů a můžeme uvažovat analogicky, ale vizualizace by zdaleka nebyla tak přímočará).

- matice A_1 má jako jediná opakující se prvek na diagonále a nadto zrovna tak, že ani nemá dominantní vlastní číslo - tedy konvergence tak jak byla odvozená na přednášce se nedá aplikovat. Intuice by nám však mohla napovědět, že problém leží v otázce *zda* metoda konverguje. Ale pokud ano, tak bychom očekávali, že bude konvergovat k nějakému vlastnímu vektoru vl. č. $\lambda_1 = \lambda_2 = 4$. Zjevně, libovolný nenulový vektor s nulovou třetí a čtvrtou složkou je vl. vektorem. Zde si všimneme, že u čtvrtého řádku konvergují první dvě složky v_k k nenulovým hodnotám a tudíž je párování A_1 a čtvrtý řádek logické. Pozorováním, že A_1 je jedinou maticí bez dominantního vl. č. a naopak že složky $(v_k)_1, (v_k)_2 \not\longrightarrow 0$ pouze ve čtvrtém řádku nás utvrdí ve správnosti.

- Matice A_2 a A_5 se liší pouze ve vlastních číslech λ_3 a λ_4 a zároveň vidíme, že podíl $|\lambda_2/\lambda_1|$ je výrazně blíže 1 než u ostatních matic. To odpovídá tomu, že řádky 1 a 2 konvergují výrazně pomaleji než ostatní. Otázku párování rozsekneme pohledem na konvergenci složek $(v_k)_3, (v_k)_4$ – ta se dle odvození na přednášce řídí podíly λ_3/λ_1 a λ_4/λ_1 tudíž snadno přiřadíme A_2 k prvnímu řádku a A_5 k druhému řádku. □

Úloha 5.

1. *Zkontrolujte si řešení Úlohy 3 pomocí scriptu* `power_method_test.m`.

2. *[Navíc]: Na druhém řádku Obrázku 3 chyba i reziduum konvergují ve dvou různých ale stálých režimech - nejprve rychleji (prvních cca pět iterací) a posléze výrazně pomaleji (od desáté iterace dále). Dokážete vysvětlit proč tomu tak je pro chybu $\|s_1 - v_k\|$? Zkonstruujte příklady, kde je tento rozdíl ještě více extrémní (tj. rozdíl mezi rychlostí konvergence je větší).*

Úloha 6.

1. *Pro matici A v* (1) *najděte nový počáteční vektor $\tilde{v}_0$ tak aby mocninná metoda konvergovala až k druhému dominantnímu vlastnímu páru, tj. $v_k \to s_2$.*

2. *Na základě v_0 a $\tilde{v}_0$ najděte třetí počáteční vektor $\hat{v}_0$, pro který mocninná metoda nejprve zdánlivě konverguje k s_2 (prvních deset iterací) ale nakonec zkonverguje k s_1 (po sedmdesáti iteracích).*

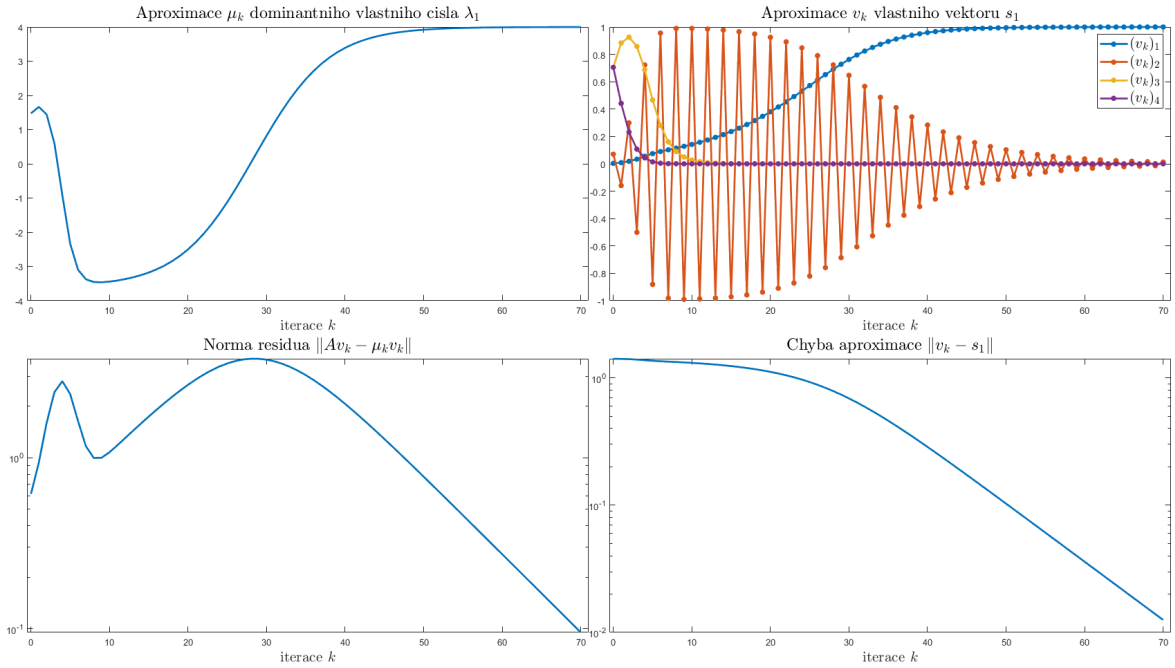
   [Hint]: Víme-li jak najít počáteční vektor tak aby $v_k \to s_1$ nebo naopak $v_k \to s_2$, naším cílem se stává skloubit tyto dva jevy dohromady a tudíž je rozumné hledat $\hat{v}_0$ jako nějakou vhodnou kombinaci v_0 a $\tilde{v}_0$. Zkombinujme tento nápad s analýzou mocninné metody z přednášky.

3. *[Navíc]: Najděte matici a počáteční vektor, pro které mocninná metoda konverguje kvalitativně podobně jako na Obrázku 4.*

Řešení.

1. Na přednášce jsme viděli, že toto nastane právě tehdy, když $c_1 := s_1^T \tilde{v}_0 = 0$ (viz Úloha 1(v)), tedy v našem případě lze volit libovolné $\tilde{v}_0$ pro které $(\tilde{v}_0)_1 = 0$.

2. Opakovaným působením diagonální matice na libovolný vektor pouze násobíme každou složku vektoru jiným číslem. Kdybychom v Algoritmu 1 vektory nenormalizoivali, všechny složky by se řídily vzorečkem $(v_k)_i = A_{ii}^k (v_0)_i$. Tedy pro $\hat{v}_0 = [\alpha, 1, 1, 1]^T$ platí $v_k = [\alpha 4^k, 3^k, 2^k, 1]^T$ a po vytknutí 4^k dostaneme $v_k = 4^k[\alpha, (3/4)^k, (1/2)^k, (1/4)^k]^T$, stejně jako při odvození konvergence na přednášce. Pokud ovšem $\alpha \ll 1$, pak nejprve uvidíme dominanci druhé složky (přesněji pro k takové, že $\alpha \ll (3/4)^k$ a zároveň $(1/2)^k, (1/4)^k \ll (3/4)^k$). Ovšem pro libovolné α od nějakého k_0 budeme v situaci kdy pro všechna další k platí $(3/4)^k, (1/2)^k, (1/4)^k \ll \alpha$ a od té chvíle bude dominantní první složka. Naprosto stejná úvaha funguje i pro mocninou metodu v Algoritmu 1 a tedy například volba $\hat{v}_0 = [0.001, 1, 1, 1]^T$ vyústí v kýženou konvergenční křivku. □



Obrázek 4: Aproximace dominantního vlastního vektoru s_1 pomocí mocninné metody.

3 Inverzní iterace

Úloha 7. *Nechť A je matice s vlastními čísly λ_j a odpovídajícími vlastními vektory s_j.*

1. *Nechť $\alpha \in \mathbb{C}$ a $(A - \alpha I)$ je regulární. Ukažte, že vlastní čísla a vlastní vektory matice $(A - \alpha I)^{-1}$ jsou rovny $(\lambda_j - \alpha)^{-1}$ a s_j.*

2. *Zvolme $\alpha = 1.1$ a matici A jako v (1). K jakému vlastnímu páru bude konvergovat mocninná metoda pro matici $(A - \alpha I)^{-1}$ a počáteční vektor $v_0 = [1, 1, 1, 1]^T$? Zobecněte pro libovolnou matici A a skalár $\alpha \in \mathbb{C}$, který nepatří do jejího spektra.*

Řešení.

1. Je-li $As_j = \lambda_j s_j$ s nenulovým vektorem s_j, máme $(A - \alpha I)s_j = (\lambda_j - \alpha)s_j$. Protože $A - \alpha I$ je nesingulární, máme tedy $s_j = (\lambda_j - \alpha)(A - \alpha I)^{-1}s_j$. Protože $A - \alpha I$ je nesingulární, tak také $\lambda_j - \alpha \neq 0$ a tedy $(A - \alpha I)^{-1}s_j = (\lambda_j - \alpha)^{-1}s_j$.

2. Vlastní čísla takové matice budou $1/2.9, 1/1.9, 1/0.9$ a $-1/0.1$. Tudíž největší vl. č. v absolutní hodnotě je to poslední a mocninná metoda tedy bude konvergovat k e_4, čtvrtému vlastnímu vektoru matice A (a tedy i matice $(A - \alpha I)^{-1}$) a vlastnímu číslu -10 matice $(A - \alpha I)^{-1}$.

   Pro obecnou matici A bude největší vlastní číslo matice $(A - \alpha I)^{-1}$ odpovídat λ_i (tj. bude rovno $1/(\lambda_i - \alpha)$), které je nejblíže hodnotě α (čím menší absolutní hodnota jmenovatele, tím větší absolutní hodnota zlomku). Volbou α tedy lze najít vlastní vektory matice A odpovídající vlastnímu číslu nejblíže α. □

Úloha 8. *[Navíc]: Pro danou matici A, skalár α a počáteční vektor v_0 naprogramujte v MATLABu funkci, která na základě Úlohy 7 iterativně aproximuje vlastní vektor s matice A odpovídající vlastnímu číslu λ matice A, které "je nejblíže α", tj. pro které je $|\lambda - \alpha|$ minimální.*

Tato metoda se jmenuje Inverzní iterace (inverse iteration).

Poznámka: Opakovaný výpočet $(A - \alpha I)^{-1}v_k$ v MATLABu buď řešte jako `(A-alpha*eye(n))\v`*, nebo jedním LU-rozkladem matice a opakovaným řešením trojúhelníkových soustav.*

Řešení.

```
function [lambda, v, history] = inverse_iteration(A,v,alpha,niter)
matrix = A - alpha*eye(size(A));
history = zeros(1,niter);
for k = 1:niter
     v = matrix\v;
     v = v/norm(v);
     history(k) = v'*(A*v);
end
lambda = history(end);
```

□