

## 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

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

**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*

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

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

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

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

*Nechť  $v_1$  je vlastní vektor příslušný  $\lambda_1$ . Ukažte, že*

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

## 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  $D = \text{diag}(\lambda_1, \dots, \lambda_n)$  jsou vlastní čísla seřazena sestupně, tj.

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

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

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

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  $(\lambda_1, s_1)$ , 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.

---

**Algoritmus 1** Mocninná metoda

---

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

---

**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

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

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

**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)`.

**Úloha 4.** Mějme pět diagonálních matic

$$A_1 = \begin{bmatrix} 4 & & & \\ & 4 & & \\ & & 2 & \\ & & & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 4 & & & \\ & 3.9 & & \\ & & 3.8 & \\ & & & 3.7 \end{bmatrix}, \quad A_3 = \begin{bmatrix} -4 & & & \\ & 3 & & \\ & & 2 & \\ & & & 1 \end{bmatrix},$$
$$A_4 = \begin{bmatrix} 4 & & & \\ & -3 & & \\ & & 2 & \\ & & & 1 \end{bmatrix}, \quad A_5 = \begin{bmatrix} 4 & & & \\ & 3.9 & & \\ & & 2 & \\ & & & 1 \end{bmatrix},$$

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  $v_k \rightarrow s_1$  a chyba  $\|s_1 - v_k\|$ . Na základě těchto úvah přiřadte matice  $A_1, \dots, A_5$  k jednotlivým řádkům Obrázku 1.

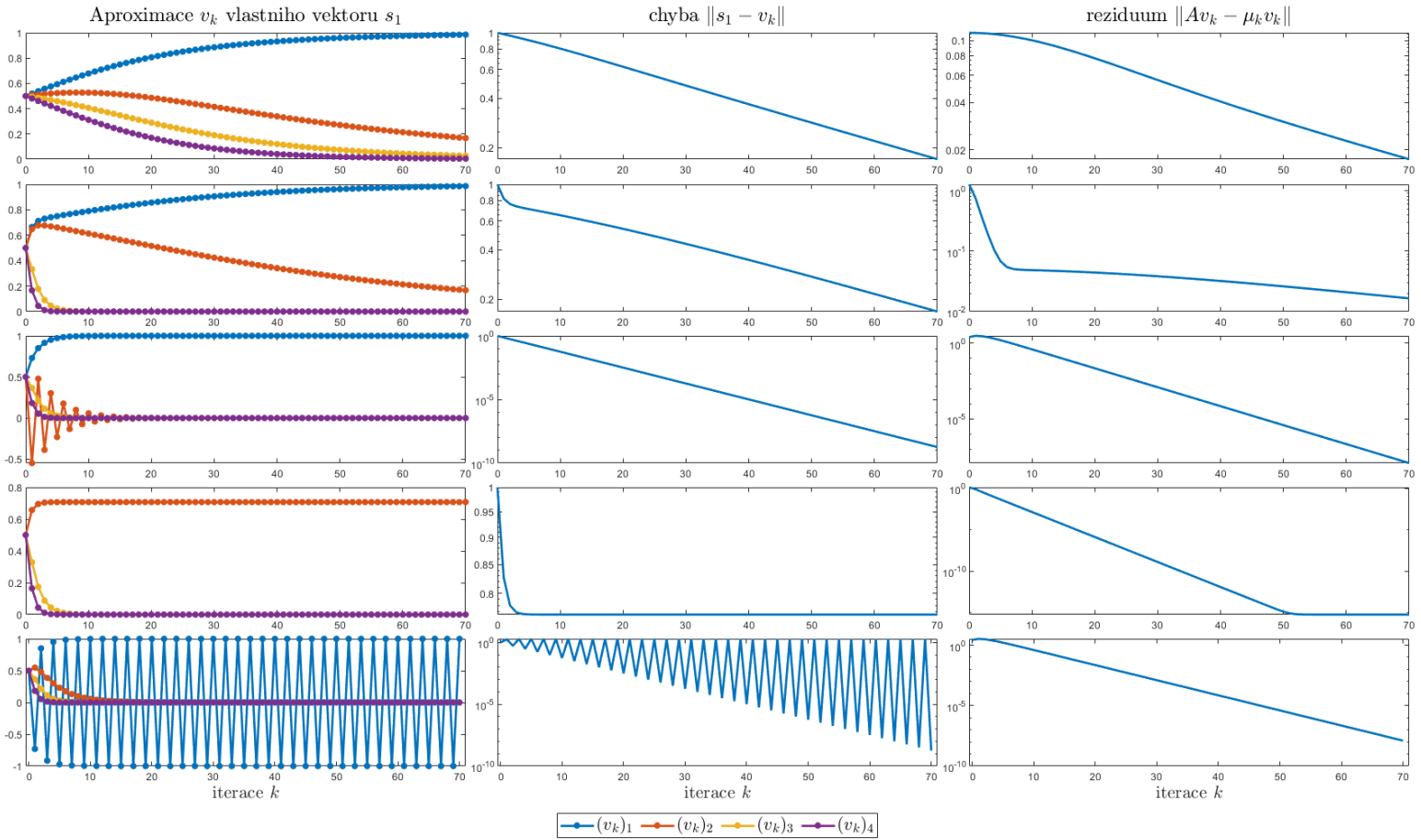
**Úloha 5.** 1. Zkontrolujte si řešení Úlohy 3 pomocí skriptu `power_method_test.m`.

2. [Navíc]: Na druhém řádku Obrázku 1 chyba  $i$  reziduum konvergují ve dvou různých ale stálých režimech - nejprve rychleji (prvních cca pět iterací) a poté 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).

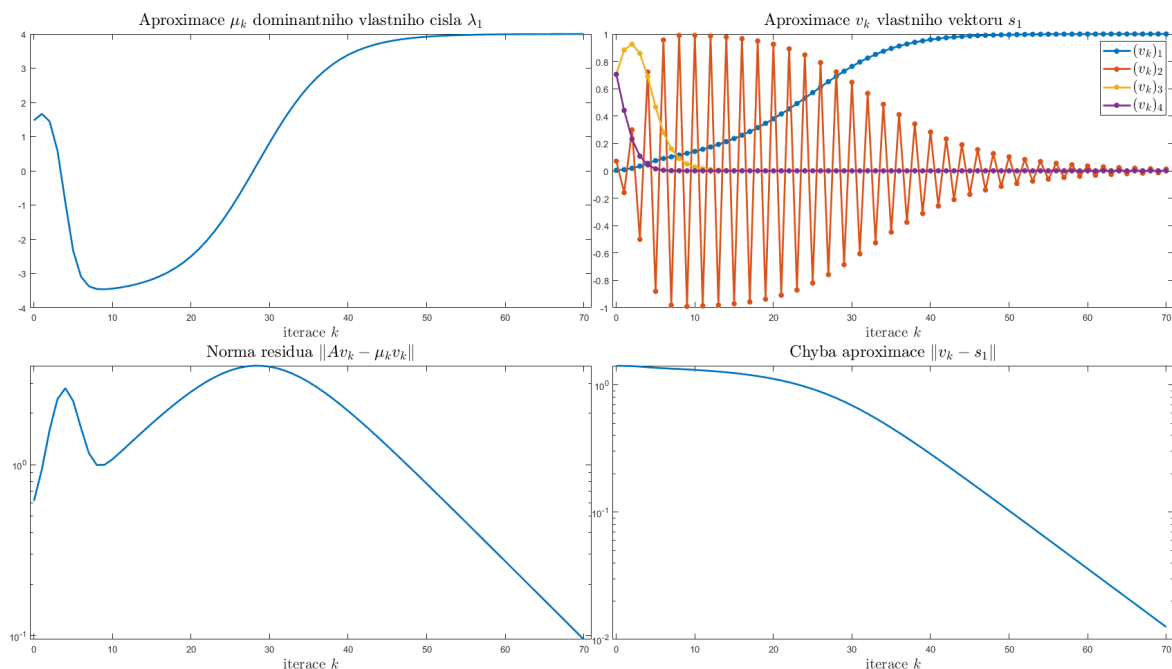
**Úloha 6.** 1. Pro matici  $A$  v (1) najděte nový počáteční vektor  $\tilde{v}_0$  tak aby mocninná metoda konvergovala až k druhému dominantnímu vlastnímu páru, tj.  $v_k \rightarrow s_2$ .

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  $v_k \rightarrow s_1$  nebo naopak  $v_k \rightarrow s_2$ , naším cílem se stává skloubit tyto dva jevy dohromady a tudíž je rozumné hledat  $\hat{v}_0$  jako nějakou vhodnou kombinaci  $v_0$  a  $\tilde{v}_0$ . Zkombinujeme tento nápad s analýzou mocninné metody z přednášky.



Obrázek 1: Aproximace dominantního vlastního vektoru  $s_1$  pomocí mocninné metody pro pět různých diagonálních matic  $A_1, \dots, A_5$  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  $(v_k)_2$ .



Obrázek 2: Aproximace dominantního vlastního vektoru  $s_1$  pomocí mocninné metody.

3. [Navíc]: Najděte matici a počáteční vektor, pro které mocninná metoda konverguje kvalitativně podobně jako na Obrázku 2.

### 3 Inverzní iterace

**Úloha 7.** Nechť  $A$  je matice s vlastními čísly  $\lambda_j$  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.

**Úloha 8.** [Navíc]: Pro danou matici  $A$ , skalár  $\alpha$  a počáteční vektor  $v_0$  naprogramujte v MATLABu funkci, která na základě Úlohy 7 iterativně aproximuje vlastní vektor s matice  $A$  odpovídající vlastnímu číslu  $\lambda$  matice  $A$ , které "je nejbližší  $\alpha$ ", 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  $(A - \alpha \cdot \text{eye}(n)) \backslash v$ , nebo jedním LU-rozkladem matice a opakovaným řešením trojúhelníkových soustav.