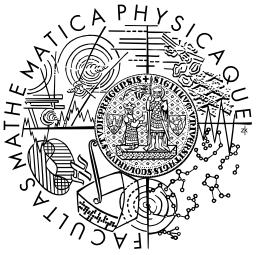


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Jan Cirbus

Implicitní QR algoritmus s násobnými shifty

Katedra numerické matematiky

Vedoucí bakalářské práce: doc. RNDr. Iveta Hnětynková, Ph.D.

Studijní program: Obecná matematika

Studijní obor: Obecná matematika

Praha 2025

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Velké poděkovaní patří vedoucí práce doc. RNDr. Ivetě Hnětynkové, Ph.D. za konzultace, věnovaný čas, cenné rady a postřehy, trpělivost a podporu nejen při psaní bakalářské práce.

Název práce: Implicitní QR algoritmus s násobnými shifty

Autor: Jan Cirbus

Katedra: Katedra numerické matematiky

Vedoucí bakalářské práce: doc. RNDr. Iveta Hnětynková, Ph.D., Katedra numerické matematiky

Abstrakt: V této práci se budeme zabývat hledáním vlastních čísel komplexních čtvercových matic. Pro tuto potřebu detailně odvodíme implicitní QR algoritmus s násobnými shifty. Při odvozování se seznámíme s mocninnou metodou, podprostorovými a simultánními iteracemi a explicitním QR algoritmem. Zavedeme do explicitního QR algoritmu shift a ukážeme jeho ekvivalenci s implicitním QR algoritmem. Následně zobecníme implicitní QR algoritmus pro aplikaci libovolného počtu shiftů a uvedeme pár shiftovacích strategií. Konvergenci a vliv volby shiftovací strategie na implicitní QR algoritmus ilustrujeme na numerických experimentech v prostředí MATLAB.

Klíčová slova: problém vlastních čísel, QR algoritmus, Francisův algoritmus, shifty, unitární transformace

Title: Implicit QR algorithm with multishifts

Author: Jan Cirbus

Department: Department of Numerical Mathematics

Supervisor: doc. RNDr. Iveta Hnětynková, Ph.D., Department of Numerical Mathematics

Abstract: In this paper, we will focus on finding the eigenvalues of complex square matrices. For this purpose, we will derive in detail the implicit QR algorithm with multiple shifts. In the derivation, we will acquaint ourselves with the power method, subspace and simultaneous iterations, and the explicit QR algorithm. We will introduce shift into the explicit QR algorithm and demonstrate its equivalence with the implicit QR algorithm. Subsequently, we will generalize the implicit QR algorithm to apply any number of shifts and present a few shifting strategies. The convergence and impact of the choice of shifting strategy on the implicit QR algorithm will be illustrated with numerical experiments in the MATLAB environment.

Keywords: eigenvalue problem, QR algorithm, Francis algorithm, shifts, unitary transformation

Obsah

Seznam použitých zkratek

$\mathbb{N}$	množina přirozených čísel
$\mathbb{R}$	množina reálných čísel
$\mathbb{C}$	množina komplexních čísel
$\mathcal{S}$,$\mathcal{T}$,$\mathcal{U}$	podprostory $\mathbb{C}^n$
$\mathcal{S} \cap \mathcal{T}$	průnik podprostorů $\mathcal{S}$ a $\mathcal{T}$
$\mathcal{S}^{\perp}$	ortogonální doplněk podprostoru $\mathcal{S}$
$dim(\mathcal{S})$	dimenze podprostoru $\mathcal{S}$
A	matice
a_{ij}	prvek matice A v i-tém řádku a j-tém sloupci
I_n	jednotková matice řádu n
A^*	hermitovsky sdružená matice k matici A
A^{-1}	inverzní matice k matici A
$Ker(A)$	jádro matice A
$\lambda_1, \ldots, \lambda_n$	vlastní čísla
$v_1, \ldots, v_n$	vlastní vektory
$span\,\{v_1, \ldots, v_n\}$	lineární obal vektorů $v_1, \ldots, v_n$
$e_1, \ldots, e_n$	kanonická báze prostoru $\mathbb{C}^n$
$\lvert a \rvert$	absolutní hodnota čísla a
$\lVert v \rVert$	norma vektoru v
$\lVert v \rVert_2$	eukleidovská norma vektoru v
$\lVert A \rVert$	norma matice A
$\lVert A \rVert_2$	spektrální norma matice A
$v \cdot w$	standardní skalární součin vektorů v a w
$\overline{z}$	komplexně sdružené číslo ke komplexnímu číslu z
$O(f(n))$	výpočetní složitost

Úvod

Úloha hledání vlastních čísel čtvercových matic se objevuje nejenom v matematice, ale také v mnoha oblastech vědy. Vlastní čísla nacházejí uplatnění v lineární algebře, funkcionální analýze, teorii řízení, elektrických obvodech nebo kvantové mechanice.

Z lineární algebry víme, že hledání vlastních čísel čtvercové matice rozměru $n \times n$ je ekvivalentní řešení polynomiální rovnice s příslušným charakteristickým polynomem stupně n. Ovšem z Abel-Ruffiniho věty plyne neexistence vzorce pro kořeny polynomu stupně vyššího než čtyři, a tedy problém vlastních čísel obecné matice velkého rozměru je analyticky neřešitelný. Proto má smysl rozvíjet a studovat numerické metody zabývající se problematikou vlastních čísel.

V této práci odvodíme jeden z nejpoužívanějších algoritmů v praxi. Začneme jednoduchou mocninnou metodou a postupnými kroky budeme odvozovat složitější a sofistikovanější algoritmy (Watkins (2002), Watkins (2008)). Během odvozování ukážeme, že je výhodné provádět explicitní, později i implicitní, QR algoritmus pro matice v Hessenbergově tvaru. Předvedeme, jak obecnou matici pomocí Householderových reflexí převést do tohoto tvaru (Watkins (2002)). Podrobně se seznámíme s explicitním QR algoritmem, ukážeme jeho konvergenci, jak ho efektivně implementovat Givensovými rotacemi a pomocí principu duality v něm zavedeme shift (Watkins (2002)). Hlavním výdobytkem práce je implicitní QR algoritmus (Watkins (2002), Aurentz a kol. (2018)), také nazývaný Francisův, který je matematicky ekvivalentní explicitnímu QR algoritmu. Poté algoritmus zobecníme pro jakoukoli zvolenou velikost shiftu (Aurentz a kol. (2018)) a zmíníme strategie, jak tyto shifty vybírat (David a Watkins (2006), Vandebril a Watkins (2012)).

V závěru práce provedeme řadu numerických experimentů v prostředí MATLAB. Otestujeme na vlastní implementaci Francisova algoritmu jeho konvergenci pro různé volby shiftů. Dále vyzkoušíme různé kombinace velikostí shiftů a shiftovacích strategií a pokusíme se experimenty najít optimální velikost shiftu. Porovnáme vliv shiftovacích strategií na rychlost konvergence. Kromě přesnosti získaných aproximací vlastních čísel nás bude zajímat také výpočetní náročnost celého algoritmu pro námi zvolený rozměr shiftu a zvolenou shiftovací strategii.

1. Mocninná metoda, podprostorové iterace a jejich modifikace

V této kapitole se budeme zabývat odvozením mocninné metody, podprostorových iterací a jejich modifikací. Tyto metody nám poslouží jako mezikroky pro odvození implicitního QR algoritmu. Začneme úlohou hledání jednoho vlastního čísla a postupnými kroky odvodíme proces, pomocí kterého se nám podaří najít libovolný počet vlastních čísel. Budeme vycházet z knihy Watkins (2002) a článku Watkins (2008).

1.1 Mocninná metoda

Zabývejme se nejprve následujícím případem. Mějme matici $A \in \mathbb{C}^{n\times n}$ a hledejme jedno vlastní číslo. O matici A navíc předpokládejme, že existuje n navzájem různých vlastních vektorů a největší vlastní číslo je jednonásobné a navíc nenulové, to označme λ_1 a v_1 příslušný vlastní vektor. Zbylá vlastní čísla označme $\lambda_2, \lambda_3, \ldots, \lambda_n$ a příslušné vlastní vektory $v_2, v_3, \ldots, v_n$, tak, aby platilo

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|\,.$$

Nyní zvolme nenulový startovní vektor $q \in \mathbb{C}^n$, který navíc splňuje $q \cdot v_1 \neq 0$. Jelikož existuje n vlastních vektorů, tak z nich můžeme sestavit bázi prostoru $\mathbb{C}^n$ a psát

$$q = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$$

pro nějaká $c_1, \ldots, c_n \in \mathbb{C}$.

Tvrzení 1. *Nechť q je vektor a A matice definovány výše. Nechť navíc platí $q \cdot v_1 \neq 0$ a $\lambda_1 \neq 0$. Pak posloupnost vektorů $\frac{1}{\|q\|}q, \frac{1}{\|Aq\|}Aq, \frac{1}{\|A^2q\|}A^2q, \ldots$ konverguje k násobku vlastního vektoru v_1.*

Důkaz. Využijeme zápisu vektoru q v bázi složené z vlastních vektorů matice A a elementárního vztahu $Av = \lambda v$, kde v je vlastní vektor matice A a λ příslušné vlastní číslo. Podívejme se na Aq:

$$\begin{aligned} Aq &= c_1 Av_1 + c_2 Av_2 + \cdots + c_n Av_n \\ Aq &= c_1\lambda_1 v_1 + c_2\lambda_2 v_2 + \cdots + c_n\lambda_n v_n. \end{aligned}$$

Pro obecné $j \in \mathbb{N}$ dostáváme

$$\begin{aligned} A^j q &= c_1 A^j v_1 + c_2 A^j v_2 + \cdots + c_n A^j v_n \\ A^j q &= c_1\lambda_1^j v_1 + c_2\lambda_2^j v_2 + \cdots + c_n\lambda_n^j v_n. \end{aligned}$$

Nyní vektory $A^j q$ přeškálujme a označme $q_j = 1/(c_1\lambda_1^j)A^j q$. Tím jsme změnili pouze normu těchto vektorů, ale jejich směr zůstává stejný. Vlastní číslo λ_1 je

nenulové podle předpokladu a c_1 je nenulové z předpokladu $q \cdot v_1 \neq 0$. Ověřme, že vektory q_j konvergují k vektoru v_1 v libovolné vektorové normě. Tedy počítáme:

$$\|q_j - v_1\| = \left\| v_1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^j v_2 + \cdots + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^j v_n - v_1 \right\| \leq$$
$$\leq \left|\frac{c_2}{c_1}\right| \left|\frac{\lambda_2}{\lambda_1}\right|^j \|v_2\| + \cdots + \left|\frac{c_n}{c_1}\right| \left|\frac{\lambda_n}{\lambda_1}\right|^j \|v_n\|.$$

Uvedená nerovnost plyne z trojúhelníkové nerovnosti a vytýkání skalárů z normy. Podíváme-li se nyní na limitu uvedeného výrazu, dostaneme:

$$\lim_{j\to\infty} \|q_j - v_1\| \leq \lim_{j\to\infty} \left|\frac{c_2}{c_1}\right| \left|\frac{\lambda_2}{\lambda_1}\right|^j \|v_2\| + \cdots + \left|\frac{c_n}{c_1}\right| \left|\frac{\lambda_n}{\lambda_1}\right|^j \|v_n\| = 0.$$

Rovnost výše je platná, jelikož $\lambda_1 > \lambda_k$ pro každé $k \geq 2$ a tedy $\lambda_k/\lambda_1 < 1$. Tím jsme získali $q_j \to v_1$ pro $j \to \infty$. Konvergence posloupnosti ze znění k násobku vektoru v_1 plyne z výpočtu

$$0 = \lim_{j\to\infty} \|q_j - v_1\| = \lim_{j\to\infty} \left(\frac{\|A^j q\|}{|c_1|\,|\lambda_1|^j} \left\| \frac{1}{\|A^j q\|} A^j q - \frac{c_1 \lambda_1^j}{\|A^j q\|} v_1 \right\| \right) \leq$$
$$\leq \|v_1\| \lim_{j\to\infty} \left\| \frac{1}{\|A^j q\|} A^j q - \frac{c_1 \lambda_1^j}{\|A^j q\|} v_1 \right\|.$$

□

Poznámka (volba q). Předpoklad $q \cdot v_1 \neq 0$ nelze předem ověřit. Pokud volíme vektor q náhodně, tak je tato podmínka splněna s pravděpodobností blízko jedné. Jelikož $q \cdot v_1 = 0$ platí právě tehdy, když $q \in span\{v_1\}$, tak pro matice $A \in \mathbb{C}^{n\times n}$, kde $n \geq 2$, je Lebesgueova míra tohoto podprostoru rovna nule.

Poznámka (přeškálování). V důkazu tvrzení 1 jsme posloupnost vektorů přeškálovali koeficientem $1/c_1\lambda_1^j$, ale hodnoty c_1 ani λ_1 předem neznáme. Při budování posloupnosti z tohoto tvrzení škálujeme vektory vhodným koeficientem σ_j. Dobrou volbou σ_j může být například největší absolutní hodnota prvků vektoru v předchozí iteraci.

Proces budování posloupnosti z tvrzení 1 se nazývá *mocninná metoda*. Začali jsme s náhodným vektorem q, který jsme vynásobili maticí A a získaný vektor jsme přeškálovali největší absolutní hodnotou jeho prvků. Tento proces můžeme iteračně opakovat a postupně získávat lepší aproximaci vlastního vektoru v_1. Naším cílem je ovšem aproximovat vlastní čísla. Jak využít získané aproximace vlastního vektoru v_1 na odhad vlastního čísla λ_1? Pokud bychom měli přesný vlastní vektor v, pak platí rovnost $Av = \lambda v$, kde λ je vlastní číslo příslušné tomuto vlastnímu vektoru. Vynásobme tuto rovnost zleva vektorem v^* a vydělme číslem v^*v. Tím dostaneme vyjádření vlastního čísla výrazem $\lambda = v^*Av/v^*v$. Tato úvaha nás vede na následující definici.

Definice 1 (Rayleighův kvocient)**.** *Nechť $A \in \mathbb{C}^{n\times n}$ je matice a $q \in \mathbb{C}^n$ nenulový vektor. Pak definujeme Rayleighův kvocient vektoru q jako číslo*

$$\rho = \frac{q^*Aq}{q^*q}.$$

Je-li navíc $\|q\|_2 = 1$, pak $\rho = q^*Aq$.

Poznámka. Dodatek v definici plyne z následující rovnosti $\|q\|_2 = \sqrt{q \cdot q} = \sqrt{q^* q}$ pro $q \in \mathbb{C}^n$.

Chtěli bychom, aby Rayleighův kvocient ρ byl dobrou aproximací vlastního čísla λ, když máme k dispozici dobrou aproximaci q vlastního vektoru v. Tato skutečnost je obsahem dalšího tvrzení.

Tvrzení 2. *Nechť $A \in \mathbb{C}^{n\times n}$ je matice, $v \in \mathbb{C}^n$ je vlastní vektor matice A splňující $\|v\|_2 = 1$ a λ příslušné vlastní číslo. Nechť $q \in \mathbb{C}^n$ je vektor splňující $\|q\|_2 = 1$ a $\rho = q^* A q$ Rayleighův koeficient vektoru q. Pak platí následující odhad*

$$|\lambda - \rho| \leq 2 \|A\|_2 \|v - q\|_2 .$$

Důkaz. Důkaz tohoto tvrzení je možné najít v knize Watkins (2002, Kapitola 5, Theorem 5.3.25).

□

Poznámka. Norma matice A, která se vyskytuje v odhadu z tvrzení, se nazývá spektrální norma a je definována jako $\|A\|_2 = \sqrt{\lambda_{max}(A^* A)}$, kde $\lambda_{max}(A^* A)$ je největší vlastní číslo matice $A^* A$.

Mějme dobrou aproximaci q vlastního vektoru v splňující $\|v - q\|_2 < \epsilon$ pro $\epsilon > 0$ malé. Pak z tvrzení 2 ihned dostáváme, že Rayleighův kvocient ρ vektoru q splňuje $|\lambda - \rho| \leq 2 \|A\|_2 \epsilon$. Tedy pokud umíme získat vektor q tak, aby ϵ bylo libovolně malé, tak jsme schopni se pomocí Rayleighova kvocientu libovolně blízko přiblížit skutečné hodnotě vlastního čísla λ. Poznamenejme, že hodnota $\|A\|_2$ může být velká a tedy pro nějaké matice A může být i pro dobrou aproximaci vlastního vektoru q Rayleighův kvocient ρ rozdílný od skutečné hodnoty vlastního čísla.

1.2 Modifikace mocninné metody

Pomocí mocninné metody se nám podařilo získat aproximaci vlastního vektoru příslušného největšímu vlastnímu číslu. V této sekci si ukážeme, jak získat aproximaci libovolného vlastního vektoru. Na začátek uveďme dvě tvrzení, která nás navedou, jak modifikovat mocninnou metodu.

Tvrzení 3. *Nechť $A \in \mathbb{C}^{n\times n}$ je regulární matice, $v \in \mathbb{C}^n$ je vlastní vektor matice A a $\lambda \in \mathbb{C}$ příslušné vlastní číslo. Pak matice A^{-1} má vlastní vektor v s příslušným vlastním číslem λ^{-1}.*

Důkaz. Jelikož je matice A regulární, tak existuje matice A^{-1} a z definice inverzní matice platí vztah $A^{-1}A = I_n$, kde I_n je jednotková matice. Nechť v je vlastní vektor matice A příslušný vlastnímu číslu λ. Pak z definice platí rovnost $Av = \lambda v$. Vynásobme tuto rovnost maticí A^{-1} zleva a následně číslem λ^{-1}. Poznamenejme, že z regularity matice A je λ nenulové a tedy λ^{-1} existuje. Proveďme výše uvedený výpočet:

$$\begin{aligned} Av &= \lambda v \\ A^{-1}Av &= A^{-1}\lambda v \\ \lambda^{-1} I_n v &= \lambda^{-1}\lambda A^{-1} v \\ \lambda^{-1} v &= A^{-1} v. \end{aligned}$$

Z posledního řádku dostáváme z definice vlastního vektoru a vlastního čísla, že v je vlastní vektor matice A^{-1} příslušný vlastnímu číslu λ^{-1} a tím dostáváme požadovaný závěr.

□

Poznámka. Je-li $A \in \mathbb{C}^{n\times n}$ regulární matice s vlastními vektory $v_1,\ldots,v_n \in \mathbb{C}^n$ a příslušnými vlastními čísly $\lambda_1,\ldots,\lambda_n \in \mathbb{C}$, pak z tvrzení 3 ihned dostáváme, že matice A^{-1} má vlastní vektory $v_1,\ldots,v_n$ s příslušnými vlastními čísly $\lambda_1^{-1},\ldots,\lambda_n^{-1}$.

Tvrzení 4. *Nechť $A \in \mathbb{C}^{n\times n}$ je matice, $v \in \mathbb{C}^n$ je vlastní vektor matice A, $\lambda \in \mathbb{C}$ příslušné vlastní číslo a $\rho \in \mathbb{C}$ číslo. Pak matice $A - \rho I_n$ má vlastní vektor v s příslušným vlastním číslem $\lambda - \rho$.*

Důkaz. Z definice vlastního vektoru platí rovnost $Av = \lambda v$. Od této rovnosti odečteme vektor ρv. Proveďme odvození:

$$\begin{aligned} Av &= \lambda v \\ Av - \rho v &= \lambda v - \rho v \\ Av - \rho I_n v &= (\lambda - \rho)v \\ (A - \rho I_n)v &= (\lambda - \rho)v. \end{aligned}$$

Z poslední rovnosti a definice vlastního vektoru a vlastního čísla dostáváme, že v je vlastní vektor matice $A - \rho I_n$ příslušný vlastnímu číslu $\lambda - \rho$, a tedy závěr dokazovaného tvrzení.

□

Poznámka. Obdobně jako v poznámce za tvrzením 3 platí následující. Je-li $A \in \mathbb{C}^{n\times n}$ s vlastními vektory $v_1,\ldots,v_n \in \mathbb{C}^n$ a příslušnými vlastními čísly $\lambda_1,\ldots,\lambda_n \in \mathbb{C}$, pak matice $A - \rho I_n$ má vlastní vektory $v_1,\ldots,v_n \in \mathbb{C}^n$ s příslušnými vlastními čísly $\lambda_1 - \rho,\ldots,\lambda_n - \rho$.

Uvažujme nyní regulární matici $A \in \mathbb{C}^{n\times n}$, jejíž vlastní čísla splňují následující podmínku:

$$|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_{n-1}| > |\lambda_n|\,.$$

Příslušné vlastní vektory označme $v_1,\ldots,v_n$. Pak podle poznámky za tvrzením 3 má matice A^{-1} vlastní vektory $v_1,\ldots,v_n$ a vlastní čísla $\lambda_1^{-1},\ldots,\lambda_n^{-1}$, která splňují opačnou nerovnost, tedy

$$|\lambda_1|^{-1} \leq |\lambda_2|^{-1} \leq \cdots \leq |\lambda_{n-1}|^{-1} < |\lambda_n|^{-1}\,.$$

Zvolme nenulový vektor $q \in \mathbb{C}^n$ splňující $q \cdot v_n \neq 0$ a proveďme mocninnou metodu s tímto vektorem a maticí A^{-1}. Tedy budujme posloupnost

$$\frac{1}{\|q\|}q, \frac{1}{\|A^{-1}q\|}A^{-1}q, \frac{1}{\|A^{-2}q\|}A^{-2}q, \ldots,$$

která podle tvrzení 1 konverguje k násobku vlastního vektoru v_n příslušného největšímu vlastnímu číslu λ_n^{-1}. Podařilo se nám získat aproximaci vlastního vektoru v_n, která je také aproximací vlastního vektoru původní matice A. Pomocí Rayleighova kvocientu můžeme spočítat aproximaci vlastního čísla λ_n. Výše popsaná modifikace se nazývá *inverzní mocninná metoda*.

Poznámka. V každé iteraci inverzní mocninné metody je potřeba spočítat vektor q_j ze vztahu $q_j = A^{-1}q_{j-1}$, kde q_{j-1} je aproximace vlastního vektoru z předchozí iterace. V praxi nepočítáme inverzi matice A, ale řešíme $Aq_j = q_{j-1}$ jako soustavu lineárních rovnic metodami numerické matematiky.

Mějme nyní k dispozici aproximaci $\rho \in \mathbb{C}$ vlastního čísla $\lambda_k \in \mathbb{C}$ regulární diagonalizovatelné matice $A \in \mathbb{C}^{n\times n}$ pro $k \in \{1,\dots,n\}$. Předpokládejme, že ρ je dostatečně dobrá aproximace, ale $\rho \neq \lambda_k$. Tím rozumíme platnost nerovnosti $|\lambda_k - \rho| < |\lambda_j - \rho|$ pro každé $j \in \{1,\dots n\}$ různé od k. Podle poznámky za tvrzením 4 má matice $A - \rho I_n$ vlastní vektory $v_1,\dots,v_n$ s příslušnými vlastními čísly $\lambda_1 - \rho,\dots,\lambda_n - \rho$. Dostali jsme se do podobné situace jako výše, tj. matice $A - \rho I_n$ má v absolutní hodnotě nejmenší vlastní číslo $\lambda_k - \rho$. Matice $A - \rho I_n$ je regulární z regularity matice A a z poznámky za tvrzením 3 platí, že matice $(A - \rho I_n)^{-1}$ má vlastní vektory $v_1,\dots,v_n$ a příslušná vlastní čísla $(\lambda_1 - \rho)^{-1},\dots,(\lambda_n - \rho)^{-1}$. Jednoduchou úpravou nerovnosti $|\lambda_k - \rho| < |\lambda_j - \rho|$ dostaneme ekvivalentní nerovnost $|\lambda_k - \rho|^{-1} > |\lambda_j - \rho|^{-1}$ pro každé $j \in \{1,\dots,n\}$ různé od k. A jelikož je $(\lambda_k - \rho)^{-1}$ dominantním vlastním číslem matice $(A - \rho I_n)^{-1}$, tak na tuto matice můžeme aplikovat mocninnou metodu a budovaná posloupnost z tvrzení 1 bude konvergovat k násobku vlastního vektoru v_k. Jinými slovy jsme aplikovali inverzní mocninnou metodu na matici $A - \rho I_n$. Tímto způsobem se nám podařilo získat aproximaci vlastního vektoru v_k, který je také aproximací vlastního vektoru původní matice A. Číslo ρ používané k „posunutí“ spektra matice A nazýváme shift. Podle tohoto označení říkáme této modifikaci mocninné metody *invert-shift strategie*. Tato metoda má mnoho výhod oproti klasické mocninné metodě. Dovoluje nám najít libovolný vlastní vektor matice, pokud máme dobrý odhad pro příslušné vlastní číslo. Další důležitou výhodu ilustrujme na příkladě.

Příklad. Mějme matici $A = \begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix}$. Chceme najít vlastní vektor pomocí mocninné metody a pomocí invert-shift strategie. Zřejmě má tato matice vlastní čísla $\lambda_1 = 2$ a $\lambda_2 = -2$ a příslušné vlastní vektory jsou $v_1 = (1 \quad 0)^T$ a $v_2 = (0 \quad 1)^T$. Aplikujme mocninnou metodu se startovním vektorem $q_0 = (1 \quad 1)^T$. Podmínka $q_0 \cdot v_1 \neq 0$ je splněna. Pokud provedeme mocninnou metodu a výsledný vektor přeškálujeme, dostaneme posloupnost:

$$q_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad q_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad q_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad q_3 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \dots$$

Tato posloupnost osciluje a tedy diverguje. Důvodem je nesplněná podmínka oddělení největší absolutní hodnoty vlastních čísel, jelikož platí $|\lambda_1| = |\lambda_2|$. Zkusme nyní aplikovat invert-shift strategii. Zvolme například shift $\rho = 1$ a startovní vektor q_0 jako výše. Pokud nyní aplikujeme mocninnou metodu na matici $(A - \rho I_n)^{-1}$, dostaneme posloupnost:

$$q_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad q_1 = \begin{pmatrix} 1 \\ -1/3 \end{pmatrix}, \quad q_2 = \begin{pmatrix} 1 \\ 1/9 \end{pmatrix}, \quad q_3 = \begin{pmatrix} 1 \\ -1/27 \end{pmatrix}, \quad \dots$$

Tato posloupnost zřejmě konverguje k vlastnímu vektoru $v_1 = (1 \quad 0)^T$.

Z příkladu je vidět, že pomocí invert-shift strategie můžeme řešit rozsáhlejší třídu matic. Nejsme omezeni podmínkou oddělení absolutních hodnot vlastních čísel, čehož využijeme v následujících kapitolách o QR algoritmu.

1.3 Podprostorové a simultánní iterace

Podprostorové iterace jsou zobecněním mocninné metody, pomocí které budeme schopni najít více vlastních vektorů studované matice. Než uvedeme podprostorové iterace, budeme muset definovat důležitý pojem.

Definice 2 (vzdálenost podprostorů)**.** *Nechť $\mathcal{S}_1$,$\mathcal{S}_2$ jsou podprostory $\mathbb{C}^n$ splňující $dim(\mathcal{S}_1) = dim(\mathcal{S}_2)$. Vzdálenost podprostorů $\mathcal{S}_1$ a $\mathcal{S}_2$ definujeme jako sinus největšího kanonického úhlu mezi $\mathcal{S}_1$ a $\mathcal{S}_2$. Značíme ji $d(\mathcal{S}_1,\mathcal{S}_2)$.*

Poznámka. Definici kanonických úhlů můžeme najít v knize Golub a Van Loan (1996, Kapitola 12, Podkapitola 12.4.3).

Z definice plyne následující fakt. Pokud $d(\mathcal{S}_1,\mathcal{S}_2) = 0$, pak $\mathcal{S}_1 \subseteq \mathcal{S}_2$ nebo $\mathcal{S}_2 \subseteq \mathcal{S}_1$. Vzdálenost podprostorů je důležitým nástrojem, pokud chceme hovořit o konvergenci podprostorů. Uvažujme posloupnost $\{\mathcal{S}_j\}_{j=1}^{\infty}$ podprostorů $\mathbb{C}^n$ dimenze k. Řekneme, že tato posloupnost konverguje k podprostoru $\mathcal{S}$ dimenze k(značíme $\mathcal{S}_j \to \mathcal{S}$), pokud $\lim_{j\to\infty} d(\mathcal{S}_j,\mathcal{S}) = 0$. Nyní se můžeme vrátit k problematice hledání vlastních vektorů. Mějme diagonalizovatelnou matici $A \in \mathbb{C}^{n\times n}$ jejíž vlastní čísla splňují $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. Připomeňme mocninnou metodu. Zvolme startovní vektor $q \in \mathbb{C}^n$ a budujme posloupnost

$$\frac{1}{\|q\|}q, \frac{1}{\|Aq\|}Aq, \frac{1}{\|A^2q\|}A^2q, \ldots,$$

která podle tvrzení 1 konverguje k násobku vlastního vektoru v_1 příslušného vlastnímu číslu λ_1. Na tento proces můžeme nahlížet z pohledu podprostorů. Posloupnost konverguje k libovolnému násobku vektoru v_1, což je reprezentant podprostoru $span\{v_1\}$. Podobně můžeme nahlížet na jednotlivé prvky posloupnosti. Tedy tvoříme posloupnost $\{span\{A^jq\}\}_{j=0}^{\infty}$. Začali jsme s podprostorem $span\{q\}$, který má dimenzi jedna. Přirozeně se můžeme ptát, co se stane, pokud zvětšíme dimenzi startovního podprostoru. Tedy začneme s podprostorem $\mathcal{S} \subseteq \mathbb{C}^n$ dimenze k a budeme budovat posloupnost

$$\mathcal{S}, A\mathcal{S}, A^2\mathcal{S}, \ldots,$$

kde $A^j\mathcal{S} = \{A^js : s \in \mathcal{S}\}$. Bude tato posloupnost konvergovat k podprostoru $\mathcal{T}_k = span\{v_1,\ldots,v_k\}$? Výše popsaný proces budování posloupnosti nazýváme *podprostorové iterace*. Posloupnost skutečně konverguje k podprostoru $\mathcal{T}_k$, což je předmětem následujícího tvrzení.

Tvrzení 5. *Nechť $A \in \mathbb{C}^{n\times n}$ je diagonalizovatelná matice s vlastními vektory $v_1,\ldots,v_n \in \mathbb{C}^n$ a příslušná vlastní čísla splňují $|\lambda_1| \geq \cdots \geq |\lambda_k| > |\lambda_{k+1}| \geq \cdots \geq |\lambda_n|$ pro $k \in \mathbb{N}$ takové, že $1 \leq k \leq n-1$. Označme $\mathcal{T}_k = span\{v_1,\ldots,v_k\}$ a $\mathcal{U}_k = span\{v_{k+1},\ldots,v_n\}$. Nechť $\mathcal{S} \subseteq \mathbb{C}^n$ je podprostor dimenze k splňující $\mathcal{S} \cap \mathcal{U}_k = \{0\}$. Pak $A^j\mathcal{S}$ konverguje k $\mathcal{T}_k$ pro $j \to \infty$.*

K důkazu tvrzení 5 budeme potřebovat následující lemma.

Lemma 6. *Nechť $\mathcal{S}$ je podprostor $\mathbb{C}^n$ dimenze k, kde $k \in \mathbb{N}$ splňuje $1 \leq k \leq n$. Dále nechť $A \in \mathbb{C}^{n\times n}$ je matice splňující $Ker(A) \cap \mathcal{S} = \{0\}$. Pak podprostor $A\mathcal{S} = \{As : s \in \mathcal{S}\}$ má dimenzi k.*

Důkaz. Buď $\mathcal{S}$ podprostor $\mathbb{C}^n$ dimenze k. Zvolme jeho bázi $s_1,\ldots,s_k$. Jelikož $s_1,\ldots,s_k$ je báze, tak platí $\sum_{j=1}^{k} c_j s_j \neq 0$ pro libovolné skaláry $c_j \in \mathbb{C}$, z nichž je alespoň jeden nenulový. Vynásobíme-li $\sum_{j=1}^{k} c_j s_j$ maticí A, tak dostaneme $\sum_{j=1}^{k} c_j A s_j$. Tato lineární kombinace je také nenulová, protože $As_j \neq 0$ pro každé $j = 1,\ldots,k$ z předpokladu $Ker(A) \cap \mathcal{S} = \{0\}$. Zřejmě $As_j \in A\mathcal{S}$ pro každé $j = 1,\ldots,k$ a podle výše ukázaného je množina vektorů $\{As_1,\ldots,As_k\}$ lineárně nezávislá a tedy tvoří bázi podprostoru $A\mathcal{S}$. Tudíž $A\mathcal{S}$ má k-prvkovou bázi a proto $dim(A\mathcal{S}) = k$.

□

Poznámka. V důkazu lemmatu 6 se nám podařilo ukázat následující. Mějme bázi $s_1,\ldots,s_k$ podprostoru $\mathcal{S} \subseteq \mathbb{C}^n$. Pak $As_1,\ldots,As_k$ je báze podprostoru $A\mathcal{S}$. Tuto skutečnost využijeme později.

Nyní máme vše připravené pro důkaz tvrzení 5.

Důkaz. (Tvrzení 5) Důkaz provedeme ve dvou krocích. Nejprve ukážeme inkluzi $\lim_{j\to\infty}(A^j\mathcal{S}) \subseteq \mathcal{T}_k$ a následně ukážeme, že $dim(A^j\mathcal{S}) = k$. Pak totiž nutně musí nastat rovnost podprostorů, jelikož $\mathcal{T}_k$ má dimenzi k a $A^j\mathcal{S}$ má pro libovolné j také dimenzi k.
První krok: Nechť $q \in \mathcal{S}$ je libovolný nenulový vektor. Zapišme q v bázi složené z vlastních vektorů matice A a vynásobme ho maticí A^j. Tím dostaneme:

$$\begin{aligned} q &= c_1 v_1 \quad + c_2 v_2 \quad + \ldots + c_n v_n \\ A^j q &= c_1 A^j v_1 + c_2 A^j v_2 + \ldots + c_n A^j v_n \\ A^j q &= c_1 \lambda_1^j v_1 + c_2 \lambda_2^j v_2 + \ldots + c_n \lambda_n^j v_n, \end{aligned}$$

kde $c_i \in \mathbb{C}$ jsou jednoznačně určené skaláry. Z předpokladu $|\lambda_k| > |\lambda_{k+1}|$ dostáváme $|\lambda_k| > 0$. Tedy můžeme poslední vyjádření $A^j q$ vydělit číslem λ_k^j a dostaneme:

$$\frac{1}{\lambda_k^j} A^j q = c_1 \left(\frac{\lambda_1}{\lambda_k}\right)^j v_1 + \cdots + c_k \left(\frac{\lambda_k}{\lambda_k}\right)^j v_k + c_{k+1} \left(\frac{\lambda_{k+1}}{\lambda_k}\right)^j v_{k+1} + \cdots + c_n \left(\frac{\lambda_n}{\lambda_k}\right)^j v_n.$$

Všimněme si, že alespoň jedno z $c_1,\ldots,c_k$ je nenulové, jelikož $q \notin \mathcal{U}_k$. Dále velikost $\lambda_1/\lambda_k,\ldots,\lambda_k/\lambda_k$ roste nebo zůstává konstantní pro $j \to \infty$ z předpokladu $|\lambda_1| \geq \cdots \geq |\lambda_k|$. Naopak $\lambda_{k+1}/\lambda_k,\ldots,\lambda_n/\lambda_k$ konvergují k nule pro $j \to \infty$ díky $|\lambda_k| > |\lambda_{k+1}| \geq \cdots \geq |\lambda_n|$. Dostáváme tedy $A^j q \to \tilde{v} \in \mathcal{T}_k$ pro $j \to \infty$. Jelikož jsme $q \in \mathcal{S}$ volili libovolně, dostáváme požadovanou inkluzi

$$\lim_{n\to\infty}(A^j\mathcal{S}) \subseteq \mathcal{T}_k.$$

Druhý krok: Chceme ukázat $dim(A^j\mathcal{S}) = k$ pro každé j. Použijeme lemma 6. Podprostor $\mathcal{S}$ má dimenzi k z předpokladu věty. Musíme ověřit podmínku $Ker(A^j) \cap \mathcal{S} = \{0\}$ pro každé j. V předpokladech věty máme podmínku $\mathcal{S} \cap \mathcal{U}_k = \{0\}$. Ukažme, že $Ker(A^j) \subset \mathcal{U}_k$. Pokud se to podaří, tak budeme mít ověřené předpoklady lemmatu. Nechť $w \in Ker(A)^j$. Připomeňme, že $w \in Ker(A^j)$ právě tehdy, když $A^j w = 0$. Napišme vektor w pomocí báze složené z vlastních vektorů matice A, tj. $w = \sum_{i=1}^{n} \mu_i v_i$, kde $\mu_i \in \mathbb{C}$ jsou jednoznačně určené koeficienty. Dosaďme

toto vyjádření w do $A^j w = 0$ a dostaneme:

$$\sum_{i=1}^{n} \mu_i A^j v_i = 0$$
$$\sum_{i=1}^{n} \mu_i \lambda_i^j v_i = 0$$

Aby platila rovnost, tak musí být nulové λ_i^j nebo μ_i. Už víme, že $|\lambda_1| \geq \cdots \geq |\lambda_k| > 0$ a tedy pro $i = 1,\ldots,k$ je také $\lambda_i^j > 0$. Dostáváme, že $\mu_i = 0$ pro $i = 1,\ldots,k$. A tedy $w \in \mathcal{U}_k$, což jsme chtěli ukázat. Z lemmatu dostáváme $dim(A^j\mathcal{S}) = k$ pro každé j. Celkem tedy $\lim_{j\to\infty}(A^j\mathcal{S}) = \mathcal{T}_k$.

□

Poznámka. V předpokladech tvrzení 5 požadujeme, aby $\mathcal{S} \cap \mathcal{U}_k = 0$. Tento předpoklad je analogií předpokladu $q \cdot v_1 = 0$ v tvrzení 1. Jde ukázat, že pro náhodnou volbu báze prostoru $\mathcal{S}$ je tento předpoklad splněn s pravděpodobností blízko jedné. Později ukážeme, že pro matice ve speciálním (tzv. horním Hessenbergově) tvaru je tento předpoklad splněn, pokud zvolíme kanonickou bázi $e_1,\ldots,e_k$.

V tvrzení 5 budujeme posloupnost $\mathcal{S}$,$A\mathcal{S}$,$A^2\mathcal{S}$,... pro $\mathcal{S} \subseteq \mathbb{C}^n$ dimenze k a matici $A \in \mathbb{C}^{n\times n}$ splňující předpoklady tvrzení. Jak takovou posloupnost sestavit v praxi? Každý vektor $s \in \mathcal{S}$ můžeme vyjádřit pomocí báze podprostoru $\mathcal{S}$. Označme tuto bázi $q_1^{(0)},\ldots,q_k^{(0)}$. Podle poznámky za lemmatem 6 je $Aq_1^{(0)},\ldots,Aq_k^{(0)}$ báze podprostoru $A\mathcal{S}$. Tímto způsobem můžeme iterovat a získávat další báze budované posloupnosti. Problémem je, že tyto báze jsou špatně podmíněné. Uvažujme, že platí $|\lambda_1| > |\lambda_2|$. Budou-li splněny předpoklady tvrzení 1, pak každý z vektorů báze $q_1^{(0)},\ldots,q_k^{(0)}$ konverguje k $span\,\{v_1\}$. To znamená, že vektory báze mají po dostatečně mnoha iteracích skoro stejný směr.

Této nepříjemné vlastnosti se můžeme zbavit ortonormalizací báze v každé iteraci. Začněme s ortonormální bází $q_1^{(0)},\ldots,q_k^{(0)}$ podprostoru $\mathcal{S}$. Proveďme násobení $Aq_1^{(0)},\ldots,Aq_k^{(0)}$ a následně ortonormalizujme tuto bázi a označme ji $q_1^{(1)},\ldots,q_k^{(1)}$. Tento proces můžeme opakovat a postupně budeme získávat lepší aproximace podprostoru $\mathcal{T}_k = span\,\{v_1,\ldots,v_k\}$. Právě popsaná modifikace podprostorových iterací se nazývá *simultánní iterace*. V praxi k ortonormalizaci využíváme Gram-Schmidtův proces, který zachovává lineární obaly prvních i vektorů, tj.

$$span\left\{Aq_1^{(m-1)},\ldots,Aq_i^{(m-1)}\right\} = span\left\{q_1^{(m)},\ldots,q_i^{(m)}\right\}$$

pro libovolné $1 \leq i \leq k$. Důsledkem je, že simultánní iterace na podprostor $\mathcal{S}$ automaticky provádí simultánní iterace i na podprostory $\mathcal{S}_1,\ldots,\mathcal{S}_{k-1}$, kde $\mathcal{S}_j = span\left\{q_1^{(0)},\ldots,q_j^{(0)}\right\}$ pro $j = 1,\ldots,k-1$.

2. Explicitní a implicitní QR algoritmus

V této části odvodíme ze simultánních iterací explicitní verzi QR algoritmu a následně ukážeme ekvivalenci s implicitní verzí QR algoritmu. Zdrojem pro tuto část jsou knihy Watkins (2002) a Aurentz a kol. (2018) a článek Watkins (2008). Než odvodíme explicitní QR algoritmus, uvedeme důležitý tvar matice, takzvaný horní Hessenbergův, a jeho vlastnosti.

2.1 Matice v horním Hessenbergově tvaru

Definice 3 (Horní Hessenbergův tvar)**.** *Řekneme, že matice $A \in \mathbb{C}^{n\times n}$ s prvky $(a_{ij})_{i,j=1}^{n}$ je v horním Hessenbergově tvaru, pokud pro každou dvojici indexů $i,j \in \{1,\dots,n\}$ splňující $i > j+1$ platí $a_{ij} = 0$. Pokud jsou navíc prvky matice $(a_{ij})_{i,j=1}^{n}$ s indexy $i,j \in \{1,\dots,n\}$ splňujícími $i = j+1$ různé od nuly, pak řekneme, že matice A je v úplném horním Hessenbergově tvaru.*

Pro ilustraci uveďme grafické znázornění pro matici $A \in \mathbb{C}^{6\times 6}$

$$A = \begin{pmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ & * & * & * & * & * \\ & & * & * & * & * \\ & & & * & * & * \\ & & & & * & * \end{pmatrix}.$$

Ve znázornění jsou prázdná místa nulová a hvězdičky libovolná komplexní čísla. Pokud je navíc matice A hermitovská, pak má matice tridiagonální tvar

$$A = \begin{pmatrix} * & * & & & & \\ * & * & * & & & \\ & * & * & * & & \\ & & * & * & * & \\ & & & * & * & * \\ & & & & * & * \end{pmatrix}.$$

Dále budeme v textu název horní Hessenbergův tvar zkracovat na Hessenbergův tvar, respektive horní Hessenbergova matice na Hessenbergova matice. Jak později ukážeme, každou obecnou matici $A \in \mathbb{C}^{n\times n}$ můžeme převést do Hessenbergova tvaru pomocí podobnostní transformace. Připomeňme tento pojem z lineární algebry a jednu jeho vlastnost.

Definice 4 (Podobné matice)**.** *Matice $A,B \in \mathbb{C}^{n\times n}$ se nazývají podobné, pokud existuje regulární matice $R \in \mathbb{C}^{n\times n}$ taková, že $B = R^{-1}AR$. Tato rovnost se nazývá podobnostní transformace maticí R.*

Poznámka. Důležitým případem je takzvaná unitární transformace, tedy transformace unitární maticí U. Následná transformace nabývá tvaru $B = U^*AU$.

Tvrzení 7. *Jsou-li matice A,$B \in \mathbb{C}^{n\times n}$ podobné, pak mají stejná vlastní čísla.*

Důkaz. Protože jsou matice A a B podobné, tak existuje regulární matice $R \in \mathbb{C}^{n\times n}$ taková, že $B = R^{-1}AR$. Mějme nyní libovolné vlastní číslo $\lambda \in \mathbb{C}$ s příslušným vlastním vektorem $v \in \mathbb{C}^n$ matice B. Pak platí

$$\begin{aligned} Bv &= \lambda v \\ R^{-1}ARv &= \lambda v \\ ARv &= \lambda Rv. \end{aligned}$$

Označme vektor $u = Rv$. Tento vektor je nenulový, jelikož v je nenulový a matice R je regulární. Tím dostáváme rovnost $Au = \lambda u$ a tedy λ je vlastní číslo s příslušným vlastním vektorem u matice A.

□

Na podobností transformaci můžeme nahlížet jako na změnu báze. Podobné matice mají stejné spektrum, a tedy pokud převedeme matici do Hessenbergova tvaru, tak můžeme hledat vlastní čísla matice v tomto výhodném tvaru. Na každou iteraci v později odvozeném QR algoritmu pro matici v obecném tvaru potřebujeme $O(n^3)$ operací. Kdežto pro matici v Hessenbergově tvaru je možné provést každou iteraci s výpočetní složitostí $O(n^2)$ operací. Další výhodou matic v Hessenbergově tvaru je skutečnost, že předpoklad $\mathcal{S} \cap \mathcal{U}_k = \{0\}$ z tvrzení 5 je splněn pro volbu eukleidovské báze $e_1,\ldots,e_k$. Jelikož QR algoritmus odvodíme ze simultánních iterací, tak pro konvergenci budeme požadovat splnění tohoto předpokladu. Ukažme nyní, že tomu tak doopravdy je.

Tvrzení 8. *Nechť $A \in \mathbb{C}^{n\times n}$ je matice v úplném Hessenbergově tvaru a $k \in \{1,\ldots,n-1\}$ takové, že platí $|\lambda_k| > |\lambda_{k+1}|$, kde λ_k,$\lambda_{k+1} \in \mathbb{C}$ jsou vlastní čísla matice A. Označme $\mathcal{S} = span\,\{e_1,\ldots,e_k\}$ a dále $\mathcal{U}_k = span\,\{v_{k+1},\ldots,v_n\}$, kde $v_{k+1},\ldots,v_n \in \mathbb{C}^n$ jsou vlastní vektory příslušné vlastním číslům $\lambda_{k+1},\ldots,\lambda_n$ matice A. Pak $\mathcal{S} \cap \mathcal{U}_k = \{0\}$.*

Důkaz. Nejprve si uvědomme, že $dim(\mathcal{U}_k) = n - k$. Jelikož je prostor $\mathcal{U}_k$ lineárním obalem vlastních vektorů, tak je A-invariantní. A tedy platí, kdykoliv máme $u \in \mathcal{U}_k$, pak je také $Au \in \mathcal{U}_k$. Dále mějme $v \in \mathcal{S}$. Bez újmy na obecnosti můžeme předpokládat, že prvních k složek vektoru v je nenulových a zbylé jsou nulové. Nyní násobme maticí A vektor v. Z předpokladu, že matice A je v úplném Hessenbergově tvaru, dostaneme, že prvních $k+1$ složek vektoru Av je nenulových a zbylé jsou nulové. V násobení můžeme pokračovat a vždy přibyde jedna další nenulová složka. Tímto postupem získáme, že vektor $A^{n-k}v$ má všechny složky nenulové. Tedy množina vektorů $\{v,Av,\ldots,A^{n-k}v\}$ je lineárně nezávislá, jelikož vždy přibyde jedna nenulová složka. Poznamenejme, že dimenze této množiny je $n-k+1$. Nyní sporem ukážeme, že $\mathcal{S} \cap \mathcal{U}_k = \{0\}$. Předpokládejme tedy, že existuje nenulové $w \in \mathcal{S} \cap \mathcal{U}_k$. Jelikož $w \in \mathcal{U}_k$, pak i $A^j w \in \mathcal{U}_k$ pro libovolné $j \in \mathbb{N}$. Na druhou stranu máme, že $w \in \mathcal{S}$ a tedy posloupnost $\{w,Aw,\ldots,A^{n-k}w\}$ délky $n-k+1$ je lineárně nezávislá. Tím dostáváme spor s $dim(\mathcal{U}_k) = n-k$. Proto je $\mathcal{S} \cap \mathcal{U}_k = \{0\}$.

□

V důkazu tvrzení 8 používáme A-invariantnost podprostoru složeného z vlastních vektorů. Definujme nyní tento pojem a následně dokažme tuto vlastnost pro požadovaný podprostor.

Definice 5 (A-invariantní podprostor)**.** *Nechť $A \in \mathbb{C}^{n\times n}$ je matice. Řekneme, že podprostor $\mathcal{S} \subset \mathbb{C}^n$ je A-invariantní, pokud pro každé $v \in \mathcal{S}$ platí $Av \in \mathcal{S}$.*

Lemma 9. *Nechť $A \in \mathbb{C}^{n\times n}$ je matice a $k \in \{1,\dots,n\}$. Označíme-li $\mathcal{S} = span\,\{v_1,\dots,v_k\}$, kde $v_1,\dots,v_k \in \mathbb{C}$ jsou vlastní vektory matice A, pak podprostor $\mathcal{S}$ je A-invariantní.*

Důkaz. Jelikož jsou vlastní vektory $v_1,\dots,v_k$ lineárně nezávislé, tak můžeme libovolný vektor $s \in \mathcal{S}$ zapsat ve tvaru $s = c_1v_1 + \cdots + c_kv_k$ pro nějaké skaláry $c_1,\dots,c_k \in \mathbb{C}$. Pokud nyní vynásobíme vektor s maticí A, dostaneme

$$As = c_1Av_1 + \cdots + c_kAv_k$$
$$As = c_1\lambda_1v_1 + \cdots + c_k\lambda_kv_k.$$

Při výpočtu jsme aplikovali definici vlastního čísla, čímž jsme zjistili, že $\lambda_1,\dots,\lambda_k$ reprezentují vlastní čísla příslušná vlastním vektorům $v_1,\dots,v_k$. S ohledem na to, že $\lambda_1c_1,\dots,\lambda_kc_k$ jsou opět skaláry, závěrem máme $As \in \mathcal{S}$.

□

Tímto oddílem jsme osvětlili, proč je výhodné hledat vlastní čísla matic v Hessenbergově tvaru. Nyní se zmíníme o podobnostní transformaci, pomocí které převedeme obecnou čtvercovou matici na matici v Hessenbergově tvaru. Využijeme k tomu Householderovy reflexe, které jsou zavedeny v knize Watkins (2002, Kapitola 3, Podkapitola 3.2) včetně jejich vlastností. Pomocí těchto reflexí můžeme nulovat prvky vektoru, přesněji existuje Householderova matice H taková, že vektor $v \in \mathbb{C}^n$ má po vynásobení maticí H tvar $Hv = \begin{pmatrix} * & 0 & \cdots & 0 \end{pmatrix}^T$, kde hvězdička je komplexní číslo. Uvedeme dvě vlastnosti Householderových matic. Tyto matice jsou unitární a hermitovské, a tedy platí $H^{-1} = H^* = H$.

Mějme tedy matici $A \in \mathbb{C}^{n\times n}$ a začněme s prvním krokem transformace. Rozdělme matici A blokově na

$$A = \left(\begin{array}{c|c} a_{11} & y^T \\ \hline \tilde{x}_1 & A_1 \end{array}\right).$$

V blokovém rozdělení je a_{11} prvek matice A na pozici (1,1), $\tilde{x}_1$,$y \in \mathbb{C}^{n-1}$ jsou vektory a $A_1 \in \mathbb{C}^{(n-1)\times(n-1)}$ je matice. Nyní najdeme Householderovu matici, která vynuluje všechny složky vektoru $\tilde{x}_1$ až na první, a označíme ji $\hat{H}_1$. Pak tedy dostáváme $\hat{H}_1\tilde{x}_1 = \begin{pmatrix} b_1 & 0 & \cdots & 0 \end{pmatrix}^T$, kde $b_1 \in \mathbb{C}$. Sestavme nyní matici $H_1 \in \mathbb{C}^{n\times n}$. Tato matice má tvar

$$H_1 = \left(\begin{array}{c|c} 1 & 0^T \\ \hline 0 & \hat{H}_1 \end{array}\right).$$

Vynásobíme-li matici A maticí H_1 zleva, dostaneme

$$H_1A = \left(\begin{array}{c|c} a_{11} & y^T \\ \hline b_1 & \\ 0 & \\ 0 & \hat{H}_1A_1 \\ \vdots & \\ 0 & \end{array}\right).$$

Protože požadujeme, aby výsledná transformace byla podobnostní, musíme vynásobit matici A také maticí H_1^{-1} zprava. Víme ovšem, že matice H_1 je také Householderova a tedy platí $H_1^{-1} = H_1^* = H_1$. Poznamenejme, že vynásobení matice $H_1 A$ zprava maticí H_1 nezmění strukturu již vytvořených nul v prvním sloupci. Tím je dokončen první krok s výslednou maticí

$$H_1 A H_1 = \left(\begin{array}{c|ccc} a_{11} & * & \cdots & * \\ \hline b_1 & & & \\ 0 & & & \\ 0 & & \hat{H}_1 A_1 \hat{H}_1 & \\ \vdots & & & \\ 0 & & & \end{array}\right) = \left(\begin{array}{c|ccc} a_{11} & * & \cdots & * \\ \hline b_1 & & & \\ 0 & & & \\ 0 & & A_2 & \\ \vdots & & & \\ 0 & & & \end{array}\right).$$

V dalším kroku postupujeme obdobně. Najdeme Householderovu matici $\hat{H}_2$, která nuluje první sloupec matice A_2 až na první dva prvky. Následně sestavíme matici H_2 ve tvaru

$$H_2 = \left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & \hat{H}_2 \end{array}\right).$$

Jako výše vynásobíme již získanou matici $H_1 A H_1$ zleva i zprava maticí H_2. I zde nedojde k porušení vytvořených nul v prvním a druhém sloupci a získáváme

$$H_2 H_1 A H_1 H_2 = \left(\begin{array}{cc|ccc} a_{11} & * & * & \cdots & * \\ b_1 & * & * & \cdots & * \\ \hline 0 & b_2 & & & \\ 0 & 0 & & & \\ 0 & 0 & & A_3 & \\ \vdots & \vdots & & & \\ 0 & 0 & & & \end{array}\right).$$

Nyní už by mělo být zřejmé, jak pokračovat dál. Po $n-2$ krocích se nám povedlo pomocí podobnostních transformací převést matici A na matici

$$A_0 = H_{n-2} \dots H_2 H_1 A H_1 H_2 \dots H_{n-2}.$$

Tato výsledná matice je v Hessenbergově tvaru. Dokonce se jedná o unitární transformaci, a tedy tento proces má dobré numerické vlastnosti. Dá se ukázat, že výše uvedený proces má výpočetní náročnost $O(n^3)$ operací. Poznamenejme nakonec, že pokud začneme transformaci s hermitovskou maticí $A \in \mathbb{C}^{n\times n}$, tak po provedení celého výpočtu bude výsledná matice A_0 hermitovská tridiagonální.

2.2 Explicitní QR algoritmus

Na začátek této sekce uvedeme jedno tvrzení a dva rozklady matic, které budeme používat.

Tvrzení 10. *Nechť $A \in \mathbb{C}^{n\times n}$ je matice a $\mathcal{S} \subset \mathbb{C}^n$ je A-invariantní podprostor dimenze k, kde $k \in \{1,\dots,n-1\}$. Označme $x_1,\dots,x_k \in \mathbb{C}^n$ bázi prostoru $\mathcal{S}$.*

Nechť $x_{k+1},\dots,x_n \in \mathbb{C}^n$ jsou takové vektory, že $x_1,\dots,x_n$ tvoří bázi prostoru $\mathbb{C}^n$. Dále označme $X_1 = (x_1|\cdots|x_k)$, $X_2 = (x_{k+1}|\cdots|x_n)$ a $X = (x_1|\cdots|x_n)$. Pokud definujeme $B = X^{-1}AX$, pak B je blokově horní trojúhelníková

$$B = \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline 0 & B_{22} \end{array}\right),$$

kde $B_{11} \in \mathbb{C}^{k\times k}$. Navíc $AX_1 = X_1B_{11}$.

Důkaz. Důkaz tohoto tvrzení je možné najít v knize Watkins (2002, Kapitola 6, Podkapitola 6.1).

□

Matice X je regulární, jelikož sloupce matice X tvoří bázové vektory $\mathbb{C}^n$. Tedy matice B je podobná matici A a podle tvrzení 7 mají stejná vlastní čísla. Navíc vlastní čísla matice B jsou sjednocením spekter matic B_{11} a B_{22}, a tedy je možné úlohu rozdělit na dvě menší a dále hledat vlastní čísla zmíněných matic B_{11} a B_{22}.

Dále uvedeme zmíněné rozklady. Jedním z nich je QR rozklad.

Definice 6 (QR rozklad pro čtvercové matice)**.** *Nechť $A \in \mathbb{C}^{n\times n}$ je matice. Pak QR rozkladem matice A rozumíme rozklad*

$$A = QR,$$

ve kterém sloupce matice $Q \in \mathbb{C}^{n\times n}$ jsou ortonormální a $R \in \mathbb{C}^{n\times n}$ je horní trojúhelníková matice. Je-li navíc matice A regulární, pak matice R je regulární.

Tento rozklad se dá počítat například pomocí Gram-Schmidtova procesu. Při využití v simultánních iteracích (jak bylo vysvětleno v předchozí kapitole), pak sloupce matice Q tvoří požadovanou ortonormální bázi. Dalším rozkladem, který využijeme, je Schurův rozklad.

Tvrzení 11. *Nechť $A \in \mathbb{C}^{n\times n}$ je matice. Pak existuje unitární matice $U \in \mathbb{C}^{n\times n}$ a horní trojúhelníková matice $R \in \mathbb{C}^{n\times n}$ tak, že $R = U^*AU$.*

Důkaz. Toto tvrzení je dokázáno v knize Watkins (2002, Kapitola 5, Podkapitola 5.4).

□

Rovnost $R = U^*AU$ z tvrzení 11 můžeme ekvivalentně přepsat na $A = URU^*$. Tato rovnost se nazývá Schurův rozklad matice A. Prvky vyskytující se na diagonále matice R jsou vlastní čísla matice A. Později ukážeme, že posloupnost matic, kterou získáme QR algoritmem, konverguje právě k matici R ze Schurova rozkladu za jistých předpokladů. Jinak řečeno, pokud se nám podaří sestavit posloupnost matic $A,A_1,A_2,\dots$ takovou, že matice A_{j-1} a A_j jsou svázány unitární transformací a tato posloupnost konverguje k horní trojúhelníkové matici, pak to je přesně matice R ze Schurova rozkladu.

Požadovaný QR algoritmus odvodíme ze simultánních iterací provedených nad celým prostorem $\mathbb{C}^n$. Mějme tedy matici $A \in \mathbb{C}^{n\times n}$ splňující předpoklady tvrzení 5. Tedy máme $k \in \{1,\dots,n-1\}$ takové, že $|\lambda_k| > |\lambda_{k+1}|$ pro nějaká vlastní čísla λ_k a λ_{k+1} matice A. Označme

$$\mathcal{S}_k = span\left\{q_1^{(0)},\ldots,q_k^{(0)}\right\},$$

kde $q_1^{(0)},\ldots,q_k^{(0)}$ tvoří bázi startovního prostoru $\mathcal{S}_k$,

$$\mathcal{S} = span\left\{q_1^{(0)},\ldots,q_n^{(0)}\right\},$$

kde $q_1^{(0)},\ldots,q_n^{(0)}$ je doplněná báze prostoru $\mathcal{S}_k$ na bázi prostoru $\mathbb{C}^n$,

$$\mathcal{T}_k = span\left\{v_1,\ldots,v_k\right\} \text{ a } \mathcal{U}_k = span\left\{v_{k+1},\ldots,v_n\right\},$$

kde $v_1,\ldots,v_n$ jsou vlastní vektory matice A.

Na konci kapitoly 1.3 jsme ukázali, že kdykoliv provádíme simultánní iterace pro prostor $\mathcal{S}_k$, pak se automaticky provádějí i simultánní iterace na podprostorech $\mathcal{S}_1,\ldots,\mathcal{S}_{k-1}$ a tedy má smysl provádět simultánní iterace pro celé $\mathcal{S}$. Bez újmy na obecnosti můžeme startovní bázi prostoru $\mathcal{S}$ volit eukleidovskou. Pokud bychom převedli unitární transformací matici A na Hessenbergovu matici, tak díky tvrzení 8 máme podmínku $\mathcal{S}_k \cap \mathcal{U}_k = \{0\}$ splněnou. Nyní zapíšeme simultánní iterace pro prostor $\mathcal{S}$ v maticové formě. Začínáme se startovní bází $e_1,\ldots,e_n$ a tedy označme $\hat{Q}_0 = I_n$. První krok simultánních iterací je vynásobení každého bázového vektoru maticí A. To můžeme maticově reprezentovat jako

$$A\hat{Q}_0 = B_1.$$

Dalším krokem je výpočet ortonormální báze sloupcového prostoru matice B_1. To můžeme provést pomocí QR rozkladu, tedy maticově

$$B_1 = \hat{Q}_1\hat{R}_1.$$

Tím získáme novou bázi, která je tvořena sloupci matice $\hat{Q}_1$. Obecně v m-tém kroku máme maticově

$$\begin{aligned} A\hat{Q}_{m-1} &= B_m \\ B_m &= \hat{Q}_m\hat{R}_m. \end{aligned}$$

Stejně jako výše sloupce matice $\hat{Q}_m = \left(\begin{array}{c|c|c} q_1^{(m)} & \cdots & q_n^{(m)} \end{array}\right)$ tvoří nově získané bázové vektory. Podle tvrzení 5 máme konvergenci podprostoru $span\left\{q_1^{(m)},\ldots,q_k^{(m)}\right\}$ k $\mathcal{T}_k$ pro $m \to \infty$, a tedy pro dostatečně velké M je $span\left\{q_1^{(M)},\ldots,q_k^{(M)}\right\}$ dobrou aproximací prostoru $\mathcal{T}_k$. Dle lemmatu 9 je $\mathcal{T}_k$ A-invariantní, jelikož $\mathcal{T}_k$ je lineární obal vlastních vektorů. Provedeme-li podobnostní transformaci matice A maticí $\hat{Q}_M = \left(\begin{array}{c|c|c} q_1^{(M)} & \cdots & q_n^{(M)} \end{array}\right)$ dostaneme podle tvrzení 10 blokově horní trojúhelníkovou matici

$$A_M = \hat{Q}_M^* A \hat{Q}_M = \begin{pmatrix} A_{11}^{(M)} & A_{12}^{(M)} \\ 0 & A_{22}^{(M)} \end{pmatrix}.$$

Provedená transformace je dokonce unitární, jelikož matice $\hat{Q}_M$ je unitární, a tedy můžeme využít v transformaci vztah $\hat{Q}_M^{-1} = \hat{Q}_M^*$.

Poznamenejme, že nulový blok obsahuje pro naše M ve skutečnosti čísla blízká nule, protože máme pouze aproximaci prostoru $\mathcal{T}_k$. Tento blok konverguje k nulovému v limitě. Pokud bychom o matici A navíc předpokládali, že její vlastní čísla

splňují $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$, pak by matice $\hat{Q}_m^* A \hat{Q}_m$ konvergovala k horní trojúhelníkové pro $m \to \infty$. To je důsledkem tvrzení 10 a automatického provádění simultánních iterací na podprostorech nižších dimenzí. Získali jsme konvergenci posloupnosti matic $A, A_1, A_2, \ldots$ k horní trojúhelníkové matici, následující matici v posloupnosti získáme z předchozí unitární transformací, a tedy dostáváme konvergenci zmíněné posloupnosti k matici R ze Schurova rozkladu(tvrzení 11), která obsahuje vlastní čísla na diagonále.

Celý postup můžeme interpretovat jako hledání vhodné báze, ve které uvidíme vlastní čísla matice A. Tato vhodná báze je složená z vlastních vektorů matice A a prováděná unitární transformace je změna souřadného systému, taková aby vlastní čísla matice A byla vidět na diagonále. Nyní uvedeme, jak počítat uvedenou posloupnost $A, A_1, A_2, \ldots$ pouze z předchozí matice v posloupnosti. V první iteraci začínáme s eukleidovskou bází a tedy máme

$$A\hat{Q}_0 = AI_n = \hat{Q}_1 \hat{R}_1$$
$$A_1 = \hat{Q}_1^* A \hat{Q}_1 = \hat{Q}_1^* \hat{Q}_1 \hat{R}_1 \hat{Q}_1 = \hat{R}_1 \hat{Q}_1.$$

Na prvním řádku jsme provedli QR rozklad matice A a v druhém jsme za A dosadili $\hat{Q}_1 \hat{R}_1$. Dále jsme využili toho, že matice $\hat{Q}_1$ je unitární, a tak $\hat{Q}_1^* \hat{Q}_1 = I_n$. Zjistili jsme, že matici A_1 můžeme vypočítat změnou pořadí násobení matic z QR rozkladu. Podívejme se na druhou iteraci

$$A\hat{Q}_1 = \hat{Q}_2 \hat{R}_2 \tag{R1}$$
$$A_2 = \hat{Q}_2^* A \hat{Q}_2. \tag{R2}$$

Využijeme-li vztahů z první iterace, tak můžeme druhou iteraci vyjádřit následovně

$$A_1 = \hat{Q}_1^* \hat{Q}_2 \hat{R}_2$$
$$A_2 = \hat{Q}_2^* \hat{Q}_1 A_1 \hat{Q}_1^* \hat{Q}_2.$$

První rovnost plyne z dosazení $A\hat{Q}_1 = \hat{Q}_1 A_1$ do R1 a druhá rovnost z dosazení $A = \hat{Q}_1 A_1 \hat{Q}_1^*$ do R2. Označme $Q_2 = \hat{Q}_1^* \hat{Q}_2$, pak

$$A_1 = Q_2 \hat{R}_2$$
$$A_2 = Q_2^* A_1 Q_2.$$

Jelikož $Q_2^* A_1 = \hat{R}_2$, tak platí

$$A_1 = Q_2 \hat{R}_2$$
$$A_2 = \hat{R}_2 Q_2.$$

První rovnost je QR rozklad matice A_1. Této skutečnosti můžeme nahlédnout pomocí změny báze. Skutečně, matice přechodu od eukleidovské báze k bázi $q_1^{(1)}, \ldots, q_n^{(1)}$ je matice $\hat{Q}_1^*$. A tedy R1 vyjádřená v nové bázi má tvar $A_1 = Q_2 \hat{R}_2$. Pokud budeme vždy pracovat v nejnovější bázi, tak každá iterace je vyjádřena jednoduchým vzorcem

$$A_m = Q_{m+1} \hat{R}_{m+1}$$
$$A_{m+1} = \hat{R}_{m+1} Q_{m+1}.$$

Tomuto postupu generování posloupnosti matic se říká *explicitní QR algoritmus.*

Shrňme tento proces. Začneme s maticí $A \in \mathbb{C}^{n\times n}$, jejíž vlastní čísla splňují $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$. Převedeme matici A unitární transformací na Hessenbergovu matici A_0. Dále iterujeme pro $m = 1,2,3,\ldots$

$$\begin{aligned} A_{m-1} &= Q_m R_m \\ A_m &= R_m Q_m. \end{aligned}$$

První krok je udělat QR rozklad matice A_{m-1} a druhý položit A_m rovno součinu matic z QR rozkladu v opačném pořadí. Na první krok můžeme nahlížet jako na první krok simultánních iterací na A_{m-1} s eukleidovskou startovní bází a na druhý jako na změnu souřadného systému. Za předpokladu $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$ konverguje tvořená posloupnost k horní trojúhelníkové matici ze Schurova rozkladu. Tento fakt jsme diskutovali už dříve, ale uveďme ho jako tvrzení.

Tvrzení 12. *Nechť $A_0 \in \mathbb{C}^{n\times n}$ je Hessenbergova matice, jejíž vlastní čísla splňují $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$. Pak posloupnost matic A_0,A_1,$A_2,\ldots$ z explicitního QR algoritmu konverguje k matici $R \in \mathbb{C}^{n\times n}$, která má vlastní čísla $\lambda_1,\ldots,\lambda_n$ na diagonále a $R = U^* A_0 U$, pro $U \in \mathbb{C}^{n\times n}$ unitární. Navíc poddiagonální prvky $e_{k+1}^* A_m e_k$ konvergují k nule pro $m \to \infty$ lineárně s rychlostí $|\lambda_{k+1}/\lambda_k|$ pro každé $k \in \{1,\ldots,n-1\}$.*

Formální důkaz je příliš dlouhý, proto zde probereme pouze jeho hlavní myšlenku. Z první iterace explicitního QR algoritmu máme $A_1 = Q_1^* A_0 Q_1$, z druhé iterace máme $A_2 = Q_2^* A_1 Q_2$ a tedy po dosazení vyjádření matice A_1 dostaneme $A_2 = Q_2^* Q_1^* A_0 Q_1 Q_2$. Takto můžeme postupovat v každé iteraci, a tak pro m-tou iteraci máme vyjádření

$$A_m = Q_m^* \ldots Q_2^* Q_1^* A_0 Q_1 Q_2 \ldots Q_m.$$

Explicitní QR algoritmus generuje stejnou posloupnost matic jako simultánní iterace nad $\mathbb{C}^n$ se startovní eukleidovskou bází. Pro m-tou iteraci simultánních iterací máme $A_m = \hat{Q}_m^* A_0 \hat{Q}_m$ a tedy $\hat{Q}_m = Q_1 \ldots Q_m$. Matice A_0 splňuje všechny předpoklady tvrzení 5 pro každé $k = 1,\ldots,n-1$ a tedy $\mathcal{S}_k = span\left\{q_1^{(m)},\ldots,q_k^{(m)}\right\}$ konverguje k $\mathcal{T}_k = span\{v_1,\ldots,v_k\}$ pro každé k, kde $\left(\begin{array}{c|c|c} q_1^{(m)} & \cdots & q_n^{(m)} \end{array}\right) = \hat{Q}_m$. Podprostor $\mathcal{T}_k$ je lineární obal prvních k vlastních vektorů matice A_0 a tedy podle tvrzení 9 je A_0-invariantní. Nakonec z tvrzení 10 dostáváme $R = \lim_{m\to\infty} \hat{Q}_m^* A_0 \hat{Q}_m$, kde R je horní trojúhelníková díky konvergenci $\mathcal{S}_k$ k $\mathcal{T}_k$ pro každé $k = 1,\ldots,n-1$. Navíc $\hat{Q}_m$ je unitární pro každé $m \in \mathbb{N}$ a tedy R je matice z tvrzení 11 s vlastními čísly $\lambda_1,\ldots,\lambda_n$ na diagonále.

Důkaz. (Tvrzení 12) Kompletní důkaz je uveden v knize Wilkinson (1965, Kapitola 8).

□

2.3 Implementace explicitního QR algoritmu

V této sekci ukážeme, jak efektivně implementovat explicitní QR algoritmus. Budeme k tomu využívat Givensovy rotace, které jsou zavedeny v knize Watkins

(2002, Kapitola 3, Podkapitola 3.2). Pomocí Givensových rotací můžeme nulovat prvky vektoru. Mějme vektor $x \in \mathbb{C}^n$, který má prvních k složek nenulových, $2 \leq k \leq n$, a chceme vynulovat k-tou složku. Pak existuje matice $G \in \mathbb{C}^{n\times n}$, tak že

$$G^*x = G^* \begin{pmatrix} x_1 \\ \vdots \\ x_{k-1} \\ x_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ r_{k-1} \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

kde $r_{k-1} = \sqrt{|x_{k-1}|^2 + |x_k|^2}$. Definujme $c_{k-1} = x_{k-1}/r_{k-1}$ a $s_{k-1} = x_k/r_{k-1}$, pak matici G nazýváme maticí Givensovy rotace a má následující strukturu

$$G = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & c_{k-1} & -\overline{s}_{k-1} & & \\ & & s_{k-1} & \overline{c}_{k-1} & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix}.$$

Obdobně jako na vektory můžeme Givensovy rotace aplikovat na matice a pomocí nich tak nulovat prvky v matici. Nyní Givensovými rotacemi najdeme QR rozklad matice. Mějme matici $A \in \mathbb{C}^{n\times n}$ v Hessenbergově tvaru a chceme vynulovat prvek na pozici (2,1). Najdeme matici Givensovy rotace G_1, která působí na prvním a druhém řádku a provedeme

$$G_1^*A = \begin{pmatrix} c_1 & -\overline{s}_1 & & \\ s_1 & \overline{c}_1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots \\ a_{21} & a_{22} & \cdots \\ 0 & a_{32} & \\ \vdots & \ddots & \ddots \end{pmatrix} = \begin{pmatrix} r_1 & \tilde{a_{12}} & \cdots \\ 0 & \tilde{a_{22}} & \cdots \\ 0 & a_{32} & \\ \vdots & \ddots & \ddots \end{pmatrix}.$$

Násobení maticí G_1^* způsobí vytvoření nuly na požadovaném místě a změní první a druhý řádek matice A. Pokračujme nalezením matice Givensovy rotace G_2, která vynuluje prvek na pozici (3,2). Dostaneme

$$G_2^*G_1^*A = \\ = \begin{pmatrix} 1 & & & & \\ & c_2 & -\overline{s}_2 & & \\ & s_2 & \overline{c}_2 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix} \begin{pmatrix} r_1 & \tilde{a_{12}} & \tilde{a_{13}} & \cdots \\ 0 & \tilde{a_{22}} & \tilde{a_{23}} & \cdots \\ 0 & a_{32} & a_{33} & \cdots \\ 0 & 0 & a_{43} & \cdots \\ \vdots & & \ddots & \ddots \end{pmatrix} = \begin{pmatrix} r_1 & \tilde{a_{12}} & \tilde{a_{13}} & \cdots \\ 0 & r_2 & \hat{a_{23}} & \cdots \\ 0 & 0 & \hat{a_{33}} & \cdots \\ 0 & 0 & a_{43} & \cdots \\ \vdots & & \ddots & \ddots \end{pmatrix}$$

Násobení maticí G_2^* působí na druhém a třetím řádku, vytvoří nulu na pozici (3,2), jak jsme požadovali, a zároveň ponechá již vytvořenou nulu na pozici (2,1). Pokračujeme nalezením matice G_3,$G_4 \ldots$ a po $(n-1)$ Givensových rotacích dostaneme matici v horním trojúhelníkovém tvaru, označme ji R. Označíme-li nyní

$Q = G_1 G_2 \dots G_{n-1}$, tak máme

$$\begin{aligned} R &= G_{n-1}^* \dots G_1^* A \\ R &= Q^* A \\ A &= QR. \end{aligned}$$

Jelikož matice Givensových rotací $G_1, \dots, G_{n-1}$ jsou unitární, tak i jejich součin Q je unitární, a tedy jsme dostali rozklad matice A na horní trojúhelníkovou matici R a unitární matici Q, což je přesně QR rozklad matice A. Tím máme hotový první krok jedné iterace explicitního QR algoritmu.

Nyní, abychom dokončili iteraci explicitního QR algoritmu, musíme provést

$$\begin{aligned} A_1 &= RQ \\ A_1 &= G_{n-1}^* \dots G_1^* A G_1 \dots G_{n-1} = R G_1 \dots G_{n-1}. \end{aligned}$$

Podívejme se na součin RG_1. Matice G_1 působí na první a druhý sloupec matice R a vytvoří v nulových prvcích matice R nenulový prvek jen na pozici (2,1), ostatní zůstanou nulové. Podobně součin RG_1G_2 vytvoří z nulových prvků nenulový jen na pozici (3,2). Získáváme, že výsledná matice A_1 je opět Hessenbergova. A tedy posloupnost matic tvořených explicitním QR algoritmem je posloupnost matic v Hessenbergově tvaru.

Výhoda použití Givensových rotací spočívá v ušetření výpočetních nákladů. Provedení QR rozkladu obecné matice má výpočetní náročnost $O(n^3)$, a tedy jedna iterace explicitního QR algoritmu, která se skládá z QR rozkladu($O(n^3)$) a násobení matic($O(n^3)$), by měla výpočetní složitost $O(n^3)$. Použijeme-li ovšem výše popsaný postup pro matici v Hessenbergově tvaru, pak jedna iterace explicitního QR algoritmu má výpočetní náročnost $O(n^2)$, jelikož aplikujeme $2(n-1)$ Givensových rotací a cena jedné Givensovy rotace je $O(n)$.

2.4 Explicitní QR algoritmus se shiftem

Zavedení shiftu do explicitního QR algoritmu má dvě výhody. Po matici $A \in \mathbb{C}^{n\times n}$ jsme požadovali, aby její vlastní čísla splňovala podmínku $|\lambda_1| > \dots > |\lambda_n|$ na to, abychom mohli ukázat konvergenci explicitního QR algoritmu k horní trojúhelníkové matici. Zbavit se tohoto předpokladu nám pomůže shift. Druhou výhodou je urychlení konvergence explicitního QR algoritmu. Nyní uvedeme princip duality, který nám pomůže propojit explicitní QR algoritmus s inverzní mocninnou metodou.

Tvrzení 13. *Nechť $A \in \mathbb{C}^{n\times n}$ je regulární matice a $\mathcal{S} \subset \mathbb{C}^n$ podprostor. Označme $B = (A^*)^{-1}$, pak posloupnosti*

$$\begin{aligned} &\mathcal{S},\ A\mathcal{S},\ A^2\mathcal{S}, \dots, \\ &\mathcal{S}^\perp, B\mathcal{S}^\perp, B^2\mathcal{S}^\perp,\ \dots \end{aligned}$$

jsou duální v následujícím smyslu. Platí $(A^m\mathcal{S})^\perp = B^m\mathcal{S}^\perp$.

Důkaz. Požadovaná rovnost prostorů bude zřejmá z následující série ekvivalencí. Zvolme libovolné $v \in (A^m\mathcal{S})^\perp$, pak

$$\begin{aligned} v \in (A^m\mathcal{S})^\perp \quad &\iff \quad v \perp y, \forall y \in A^m\mathcal{S} \quad \iff \quad v \cdot y = 0, \forall y \in A^m\mathcal{S} \\ &\iff \quad v \cdot A^m\tilde{y} = 0, \forall \tilde{y} \in \mathcal{S} \quad \iff \quad (A^*)^m v \cdot \tilde{y} = 0, \forall \tilde{y} \in \mathcal{S} \\ \iff \quad (A^*)^m v \in \mathcal{S}^\perp \quad &\iff \quad v \in (A^*)^{-m}\mathcal{S}^\perp \quad \iff \quad v \in B^m\mathcal{S}^\perp. \end{aligned}$$

V první ekvivalenci používáme definici ortogonálního doplňku, v druhé definici ortogonality dvou vektorů, ve čtvrté vlastnosti standardního skalárního součinu, v páté definici ortogonality dvou vektorů a definici ortogonálního doplňku a v třetí a páté operace s prostory. Podařilo se nám ukázat, že každý vektor z $(A^m\mathcal{S})^\perp$ je také v $B^m\mathcal{S}^\perp$ a naopak. Celkem máme $(A^m\mathcal{S})^\perp = B^m\mathcal{S}^\perp$.

□

Zvolme nyní $\mathcal{S} = span\{e_1, \ldots, e_n\}$, pak posloupnost $\mathcal{S}, A\mathcal{S}, A^2\mathcal{S}, \ldots$ je aplikace simultánních iterací na $\mathcal{S}$ s maticí $A \in \mathbb{C}^{n\times n}$, které jsou ekvivalentní s explicitním QR algoritmem. Pro $k = 1, \ldots, n-1$ máme

$$span\left\{q_1^{(m)}, \ldots, q_k^{(m)}\right\} = A^m span\{e_1, \ldots, e_k\},$$

kde $\left(\begin{array}{c|c|c} q_1^{(m)} & \cdots & q_n^{(m)} \end{array} \right) = \hat{Q}_m$ je matice ze simultánních iterací. Jelikož

$$(span\{e_1, \ldots, e_k\})^\perp = span\{e_{k+1}, \ldots, e_n\},$$

použitím tvrzení 13 dostáváme

$$span\left\{q_{k+1}^{(m)}, \ldots, q_n^{(m)}\right\} = B^m span\{e_{k+1}, \ldots, e_n\}.$$

Speciálně pro $k = n-1$ platí

$$span\left\{q_n^{(m)}\right\} = B^m span\{e_n\} = (A^*)^{-1} span\{e_n\}.$$

Tedy $q_n^{(m)}$ je výsledek inverzní mocninné metody s maticí A^*. V kapitole 1.2 jsme demonstrovali, že na inverzní mocninnou metodu lze uplatnit shift. Speciálně můžeme používat jako shift Rayleighův kvocient (definice 1). Navíc můžeme v každé iteraci pomocí Rayleighova kvocientu spočítat aproximaci vlastního čísla a tuto aproximaci použít jako dynamicky se měnící shift.

Poznámka. Je možné ukázat, že zatím všechny odvozené algoritmy a metody konvergují lineárně s výjimkou invert-shift strategie s aktualizací shiftu pomocí Rayleighova kvocientu, která konverguje kvadraticky. Tedy zavedení shiftu nám urychluje konvergenci explicitního QR algoritmu k jednomu z vlastních čísel.

Mějme nyní matici $A \in \mathbb{C}^{n\times n}$ v Hessenbergově tvaru a její vlastní čísla označme $\lambda_1, \ldots, \lambda_n$. Dále mějme k dispozici $\rho \in \mathbb{C}$, potom podle tvrzení 4 matice $A - \rho I_n$ má vlastní čísla $\lambda_1 - \rho, \ldots, \lambda_n - \rho$. Přeznačme tato vlastní čísla tak, aby platilo

$$|\lambda_1 - \rho| \geq \cdots \geq |\lambda_n - \rho|.$$

Pokud jsou splněny předpoklady tvrzení 12, pak poddiagonální prvky

$$e_{k+1}^*(A_m - \rho I_n)e_k$$

konvergují k nule pro $m \to \infty$ lineárně s rychlostí $(\lambda_{k+1} - \rho)/(\lambda_k - \rho)$. Nejrychleji bude konvergovat k nule prvek $e_n^*(A_m - \rho I_n)e_{n-1}$, jelikož podíl $(\lambda_n - \rho)/(\lambda_{n-1} - \rho)$ je nejmenší. A tedy po dostatečně mnoha iteracích explicitního QR algoritmu na matici $A - \rho I_n$ můžeme prvek $e_n^*(A_m - \rho I_n)e_{n-1}$ považovat z výpočetního hlediska za nulový. Posuneme-li matici A zpět, dostaneme

$$A_m + \rho I_n = \left(\begin{array}{c|c} & a_{1,n} \\ \tilde{A}_m & \vdots \\ & a_{(n-1),n} \\ \hline 0 & a_{n,n} \end{array}\right).$$

Vlastní čísla této matice jsou sjednocením spektra matice $\tilde{A}_m$ a prvku a_{nn}. Tím jsme našli aproximaci a_{nn} vlastního čísla, které bylo nejblíž číslu ρ. V přeznačení to je přesně λ_n, proto máme $\lambda_n \approx a_{nn}$.

Zbývá najít vlastní čísla matice $\tilde{A}_m$ s menším rozměrem, která je také v Hessenbergově tvaru, tuto úlohu můžeme vyřešit stejným postupem. Rekurzivně aplikujeme metodu na $\tilde{A}_m$. Takto najdeme aproximace všech vlastních čísel původní matice A, jelikož po nalezení jednoho vlastního čísla zmenšíme rozměr matice. Procesu zmenšení rozměru matice říkáme *deflace*.

Nyní ovšem vyvstává otázka, jak volit shifty. Z výše uvedeného procesu vidíme, že prvek

$$e_n^*(A_m + \rho I_n)e_n$$

matice $A_m + \rho I_n$ konverguje v průběhu iterací k vlastnímu číslu λ_n. Jako shift ρ můžeme v každé iteraci proto volit právě aktuální prvek $e_n^*(A_m + \rho I_n)e_n$ iterační matice. V každé iteraci se podíl $(\lambda_n - \rho)/(\lambda_{n-1} - \rho)$ zmenšuje, a tedy dostáváme rychlejší konvergenci. Dá se ukázat, že při této volbě shiftu je konvergence prvku $e_n^*(A_m + \rho I_n)e_{n-1}$ k nule dokonce kvadratická. Tuto úpravu explicitního QR algoritmu s volbou shiftu $e_n^*(A_m + \rho I_n)e_n$ nazýváme *explicitní QR algoritmus s Rayleigovým shiftem.*

Shrňme nyní úplně celý postup. Hledáme vlastní čísla matice $A \in \mathbb{C}^{n\times n}$, pokud A není v Hessenbergově tvaru, převeďme ji pomocí unitární transformace na A_0 v Hessenbergově tvaru, jinak označme $A_0 = A$. Pro $m = 1,2,\ldots$ iterujeme

$$\begin{aligned} \rho_{m-1} &= e_n^* A_{m-1} e_n \\ (A_{m-1} - \rho_{m-1} I_n) &= Q_m R_m \\ A_m &= R_m Q_m + \rho_{m-1} I_n. \end{aligned} \tag{R3}$$

V prvním řádku zvolíme shift, v druhém spočítáme QR rozklad matice $A_{m-1} - \rho_{m-1} I_n$ a ve třetím vypočteme A_m ze získaného QR rozkladu. Tento postup opakujeme, dokud prvek $e_n^* A_m e_{n-1}$ matice A_m není dostatečně blízko nule. Zapamatujeme si získanou aproximaci vlastního čísla $e_n^* A_m e_n$ a následně provedeme deflaci. Celý algoritmus opakujeme rekurzivně na matici bez posledního řádku a sloupce. Pokračujeme do té doby, dokud nezískáme aproximace všech vlastních

čísel. Poznamenejme, že platí $A_m = Q_m^* A_{m-1} Q_m$, díky výpočtu

$$\begin{aligned} A_m &= R_m Q_m + \rho_{m-1} I_n \\ A_m - \rho_{m-1} I_n &= Q_m^* \left(A_{m-1} - \rho_{m-1} I_n \right) Q_m \\ A_m - \rho_{m-1} I_n &= Q_m^* A_{m-1} Q_m - \rho_{m-1} Q_m^* Q_m I_n \\ A_m &= Q_m^* A_{m-1} Q_m, \end{aligned}$$

kde do druhé rovnice jsme dosadili vyjádření $R_m = A_{m-1} - \rho_{m-1} I_n$ z R3. Z tohoto důvodu k výpočtu můžeme využít implementaci z kapitoly 2.3.

2.5 Wilkinsonův shift

Explicitní QR algoritmus s Rayleighovým shiftem nekonverguje v případě, když máme $A \in \mathbb{R}^{n \times n}$, která má komplexně sdružená vlastní čísla. Pak totiž celý algoritmus počítá v reálných číslech a k aproximaci komplexního čísla nemůže dokonvergovat. V takovém případě můžeme volit Wilkinsonův dvojitý shift.

Podstatou Wilkonsonova shiftu je vypočítat explicitním vzorcem vlastní čísla podmatice $\hat{A}$ rozměru 2×2. Je-li $A = (a_{ij})_{i,j=1}^n$, pak

$$\hat{A} \equiv \begin{pmatrix} a_{(n-1),(n-1)} & a_{(n-1),n} \\ a_{n,(n-1)} & a_{n,n} \end{pmatrix}.$$

Vlastní čísla této matice můžeme vypočítat pomocí vzorce pro kořeny kvadratické rovnice aplikovaného na charakteristický polynom. Pokud získáme oba kořeny reálné, pak vybereme ten co je bližší prvku a_{nn}, prohlásíme ho za shift a pokračujeme explicitním QR algoritmem s jedním shiftem. Pokud jsou kořeny komplexně sdružené, pak můžeme použít explicitní QR algoritmus s jedním shiftem dvakrát za sebou. Uvažujme, že jsme v m-té iteraci a označme získané kořeny $\rho_{m-1}, \overline{\rho_{m-1}}$. Provedeme

$$\begin{aligned} A_{m-1} - \rho_{m-1} I_n &= Q_m R_m \\ A_m &= R_m Q_m + \rho_{m-1} I_n \\ A_m - \overline{\rho_{m-1}} I_n &= Q_{m+1} R_{m+1} \\ A_{m+1} &= R_{m+1} Q_{m+1} + \overline{\rho_{m-1}} I_n. \end{aligned}$$

Lze dokázat, že matice A_{m+1} je reálná. Pokud ale tento proces implementujeme v nějakém matematickém softwaru, v důsledku zaokrouhlovacích chyb (výpočet probíhá v aritmetice s plovoucí řádovou čárkou) získáme výslednou matici A_{m+1}, která může být komplexní. Přechodu do komplexních čísel se však lze vyhnout. Existuje způsob, jak získat A_{m+1} přímo z matice A_{m-1}, a přitom zůstat v reálné aritmetice. Tento postup pouze stručně uvedeme. Víme, že platí vztah

$$A_{m+1} = Q_{m+1}^* A_m Q_{m+1} = Q_{m+1}^* Q_m^* A_{m-1} Q_m Q_{m+1}.$$

Označme

$$\begin{aligned} M_m \equiv \left(A_{m-1} - \overline{\rho_{m-1}} I_n \right) \left(A_{m-1} - \rho_{m-1} I_n \right) = \\ = A_{m-1}^2 - (\overline{\rho_{m-1}} + \rho_{m-1}) A_{m-1} + \overline{\rho_{m-1}} \rho_{m-1} I_n. \end{aligned}$$

Je zřejmé, že matice M_m je reálná. Pokud provedeme její QR rozklad, dostaneme $M_m = \hat{Q}\hat{R}$. Dá se ukázat, že platí $\hat{Q} = Q_m Q_{m+1}$ a tedy A_{m+1} získáme výpočtem

$$A_{m+1} = \hat{Q}^* A_{m-1} \hat{Q}.$$

Více o Wilkinsonově shiftech viz Watkins (2002) a zde uvedené odkazy.

2.6 Implicitní QR algoritmus

V každé iteraci explicitního QR algoritmu se shiftem počítáme

$$\begin{aligned}(A_{m-1} - \rho_{m-1} I_n) &= Q_m R_m \\ A_m &= R_m Q_m + \rho_{m-1} I_n,\end{aligned}$$

pro nějaký shift $\rho_{m-1} \in \mathbb{C}$. Už jsme ukázali, že není nutné konstruovat matici R_m, jelikož platí $A_m = Q_m^* A_{m-1} Q_m$, a tedy můžeme použít postup z kapitoly 2.3. Ovšem matice Givensových rotací, pomocí kterých získáme matici Q_m, musíme vytvořit z matice $(A_{m-1} - \rho_{m-1} I_n)$, a proto ji musíme explicitně sestrojit. Navíc existují případy, kdy odečtení matice $\rho_{m-1} I_n$ od matice A_{m-1} může způsobit numerickou nestabilitu výpočtu.

Implicitní QR algoritmus je metoda, jak počítat matice Givensových rotací, bez nutnosti sestrojit problematickou matici $(A_{m-1} - \rho_{m-1} I_n)$. Nejprve popíšeme implicitní QR algoritmus a následně ukážeme jeho ekvivalenci s explicitní verzí.

Ukažme si, jak funguje implicitní QR algoritmus s jedním shiftem, přidání více shiftů je poté přirozeným rozšířením této myšlenky. Uvažujme, že máme matici $A \in \mathbb{C}^{n \times n}$ v Hessenbergově tvaru a shift $\rho \in \mathbb{C}$. Potřebujeme sestrojit matice Givensových rotací $G_1, \ldots, G_{n-1}$ z kapitoly 2.3, které splňují

$$R = G_{n-1}^* \ldots G_1^* (A - \rho I_n).$$

Výsledek $\hat{A}$ jedné iterace QR algoritmu pak získáme výpočtem

$$\hat{A} = G_{n-1}^* \ldots G_1^* A G_1 \ldots G_{n-1}.$$

Matice $G_1, \ldots, G_{n-1}$ konstruujeme postupně a pomocí nich ihned provedeme podobnostní transformace, což nám umožní budovat mezivýsledky $A^{(1)}, A^{(2)}, \ldots$

$$\begin{aligned}A^{(1)} &= G_1^* A G_1 \\ A^{(2)} &= G_2^* A^{(1)} G_2 \\ &\vdots \\ \hat{A} = A^{(n-1)} &= G_{n-1}^* A^{(n-2)} G_{n-1}.\end{aligned}$$

Ukažme nyní, jak najít matici Givensovy rotace G_1. Tato matice má za úkol vynulovat prvek a_{21} uvnitř matice $A - \rho I_n$, čímž uskuteční transformaci

$$\begin{pmatrix} a_{11} - \rho \\ a_{21} \end{pmatrix} \to \begin{pmatrix} r_1 \\ 0 \end{pmatrix}.$$

V kapitole 2.3 jsme ukázali, jakou strukturu mají matice Givensových rotací a jak počítat její prvky. V našem případě položme $r_1 = \sqrt{|a_{11}-\rho|^2 + |a_{21}|^2}$, $c_1 = (a_{11}-\rho)/r_1$ a $s_1 = a_{21}/r_1$. Tedy máme

$$G_1 = \begin{pmatrix} c_1 & -\overline{s}_1 & & \\ s_1 & \overline{c}_1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix},$$

která nuluje prvek a_{21} v matici $A-\rho I_n$. Na výpočet r_1,c_1 a s_1 jsme nepotřebovali konstruovat matici $A-\rho I_n$, stačilo z matice A přečíst prvky a_{11} a a_{21}. Proveďme nyní podobnostní transformaci $G_1^* A G_1$, abychom získali mezivýsledek $A^{(1)}$. Násobení maticí G_1^* zleva změní první a druhý řádek matice A a násobení maticí G_1 zprava změní první a druhý sloupec matice $G_1^* A$. Uvědomme si, že násobení maticí G_1^* zleva vynuluje prvek a_{21} pouze v případě $\rho = 0$. Matice $A^{(1)}$ má tvar

$$A^{(1)} = G_1^* A G_1 = \left(\begin{array}{c|cccccc} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & \cdots & \cdots & a_{1,n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & \cdots & \cdots & a_{2,n}^{(1)} \\ \hline a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & \cdots & a_{3,n}^{(1)} \\ 0 & 0 & a_{43}^{(1)} & \ddots & & a_{4,n}^{(1)} \\ \vdots & \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & & a_{n-1,n}^{(1)} & a_{n,n}^{(1)} \end{array}\right).$$

Vzniklý prvek $a_{31}^{(1)}$, kvůli kterému není matice v Hessenbergově tvaru, nazýváme *bulge*.

Dále potřebujeme sestrojit matici Givensovy rotace G_2. Na to bychom ale potřebovali znát prvky $r_{22}^{(1)}, r_{23}^{(1)}$ matice $R_1 = Q_1^*(A-\rho I_n)$, kterou ovšem konstruovat nechceme. Později ukážeme, že $\left(a_{21}^{(1)},a_{31}^{(1)}\right)^T$ je násobkem $\left(r_{22}^{(1)},r_{23}^{(1)}\right)^T$, a tedy k sestrojení G_2 můžeme použít prvky $a_{21}^{(1)}, a_{31}^{(1)}$ z matice $A^{(1)}$. Položme $r_2 = \sqrt{\left|a_{21}^{(1)}\right|^2 + \left|a_{31}^{(1)}\right|^2}$, $c_2 = a_{21}^{(1)}/r_1$ a $s_2 = a_{31}^{(1)}/r_1$. Proto máme

$$G_2 = \begin{pmatrix} 1 & & & & \\ & c_2 & -\overline{s}_2 & & \\ & s_2 & \overline{c}_2 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}.$$

Proveďme podobnostní transformaci matice $A^{(1)}$ maticí G_2 a dostaneme

$$A^{(2)} = G_2^* A^{(1)} G_2 = \left(\begin{array}{cc|cccc} a_{11}^{(2)} & a_{12}^{(2)} & \cdots & \cdots & \cdots & a_{1,n}^{(2)} \\ a_{21}^{(2)} & a_{22}^{(2)} & \cdots & \cdots & \cdots & a_{2,n}^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & \cdots & a_{3,n}^{(2)} \\ \hline 0 & a_{42}^{(2)} & a_{43}^{(2)} & \ddots & & a_{4,n}^{(2)} \\ \vdots & 0 & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & & a_{n-1,n}^{(2)} & a_{n,n}^{(2)} \end{array}\right).$$

Násobením $G_2^* A^{(1)}$ vynulujeme bulge na pozici (3,1), ale násobení matice $G_2^* A^{(1)}$ maticí G_2 zprava vytvoří nový bulge na pozici (4,2). Tím jsme diagonálně posunuli bulge o jednu pozici níž. Nyní sestrojíme matici G_3 pomocí prvků $a_{32}^{(2)}, a_{42}^{(2)}$. Opět platí, že $\left(a_{32}^{(2)}, a_{42}^{(2)}\right)^T$ je násobkem $\left(r_{33}^{(2)}, r_{34}^{(2)}\right)^T$ a provedeme podobnostní transformaci $G_3^* A^{(2)} G_3$. Jako u předchozího kroku se vynuluje bulge na pozici (4,2) a posune se na pozici (5,3). Postup opakujeme, až dosáhneme

$$\hat{A} = A^{(n-1)} = G_{n-1}^* A^{(n-2)} G_{n-1}.$$

To završuje jednu iteraci implicitního QR algoritmu s jedním shiftem. Shrňme tento postup.

Začneme s maticí $A \in \mathbb{C}^{n \times n}$ v Hessenbergově tvaru a shiftem $\rho \in \mathbb{C}$. Pomocí prvků $a_{11} - \rho, a_{21}$ sestavíme matici Givensovy rotace G_1 a provedeme podobnostní transformaci matice A maticí G_1, což znamená $A^{(1)} = G_1^* A G_1$. V matici $A^{(1)}$ se vytvoří bulge na pozici (2,1). Zbylou část procesu nazýváme *bulge chasing*. Z prvků $a_{21}^{(1)}, a_{31}^{(1)}$ sestavíme G_2 a provedeme $A^{(2)} = G_2^* A^{(1)} G_2$. Tím posuneme bulge z pozice (3,2) na pozici (4,3). Pokračujeme tak dlouho, než bulge z matice úplně vyženeme, posledním krokem je $A^{(n-1)} = G_{n-1}^* A^{(n-2)} G_{n-1}$. Právě jsme popsali jednu iteraci *implicitního QR algoritmu s jedním shiftem.*

Poznámka (volba shiftu). Shift můžeme volit úplně stejně jako v kapitole 2.4. V každé iteraci položíme shift roven prvku a_{nn} z matice v předchozí iteraci. Můžeme také využít myšlenky z kapitoly 2.5. V tom případě budeme počítat v každé iteraci vlastní čísla podmatice 2×2 a jako shift zvolíme to, které je nejblíž prvku a_{nn}.

Abychom ukázali ekvivalenci explicitního a implicitního QR algoritmu, musíme ověřit, že matice $G_1, \ldots, G_{n-1}$ v explicitním QR algoritmu jsou stejné jako ty v implicitním QR algoritmu. Z odvození výše víme, že stačí dokázat, že pro každé $i = 1, .., n-2$ existuje $\beta \neq 0$ splňující

$$\begin{pmatrix} a_{i+1,i}^{(i)} \\ a_{i+2,i}^{(i)} \end{pmatrix} = \beta \begin{pmatrix} r_{i+1,i+1}^{(i)} \\ r_{i+2,i+1}^{(i)} \end{pmatrix}.$$

Zformulujeme příslušné tvrzení.

Tvrzení 14. *Givensovy rotace generované v každé iteraci $m = 1, 2, \ldots$ explicitního a implicitního QR algoritmu jsou stejné. Proto jsou explicitní a implicitní QR algoritmy matematicky ekvivalentní.*

Důkaz. Nechť $i \in \{1, .., n-2\}$ je fixní. Podle diskuse před tvrzením stačí dokázat požadovanou rovnost pro nenulové β. Matice G_1 je stejná pro obě verze algoritmu, jelikož jsme G_1 vytvořili z prvků $a_{11} - \rho$ a a_{21} matice $A^{(1)}$. Dále postupujme indukcí. Předpokládejme, že matice $G_1 \ldots, G_i$ jsou shodné pro oba algoritmy. Z implicitního algoritmu máme

$$\begin{aligned} A^{(i)} &= G_i^* \ldots G_1^* A G_1 \ldots G_i \\ A^{(i)} G_i^* \ldots G_1^* &= G_i^* \ldots G_1^* A. \end{aligned}$$

Vynásobme rovnici maticí $(A - \rho I_n)$ zprava

$$A^{(i)} G_i^* \ldots G_1^* (A - \rho I_n) = G_i^* \ldots G_1^* A (A - \rho I_n).$$

Z výpočtu $A\,(A-\rho I_n) = A^2 - \rho A = (A-\rho I_n)\,A$ vidíme, že matice A a $(A-\rho I_n)$ komutují. Připomeňme značení $R_i = Q_i^* \dots Q_1^*\,(A-\rho I_n)$. Dále využijme komutativity zmíněných matic a definici R_i. Dostaneme

$$A^{(i)} R_i = R_i A.$$

Zapišme tuto rovnost v blokovém tvaru

$$\left(\begin{array}{c|c} A_{11}^{(i)} & A_{12}^{(i)} \\ \hline A_{21}^{(i)} & A_{22}^{(i)} \end{array}\right) \left(\begin{array}{c|c} R_{11}^{(i)} & R_{12}^{(i)} \\ \hline 0 & R_{22}^{(i)} \end{array}\right) = \left(\begin{array}{c|c} R_{11}^{(i)} & R_{12}^{(i)} \\ \hline 0 & R_{22}^{(i)} \end{array}\right) \left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right).$$

Bloky $A_{11}^{(i)}, R_{11}^{(i)}$ a A_{11} mají rozměr $i \times i$, $R_{11}^{(i)}$ je horní trojúhelníková a $R_{22}^{(i)}$ je v Hessenbergově tvaru. Z blokového násobení matic plyne rovnost

$$A_{21}^{(i)} R_{11}^{(i)} = R_{22}^{(i)} A_{21}.$$

Podle indukčního předpokladu je matice $A_{21}^{(i)}$ skoro v Hessenbergově tvaru, až na porušení bulgem, a proto má jen dva nenulové prvky v posledním sloupci. Jsou to prvky $a_{i+1,i}^{(i)}$ a $a_{i+2,i}^{(i)}$. Matice A je v Hessenbergově tvaru, a tudíž A_{21} má pouze jeden nenulový prvek $a_{i+1,i}$. Pokud vynásobíme obě strany rovnice, tak získáme nenulový jen poslední sloupec. Přesně

$$r_{ii}^{(i)} \begin{pmatrix} a_{i+1,i}^{(i)} \\ a_{i+2,i}^{(i)} \end{pmatrix} = a_{i+1,i} \begin{pmatrix} r_{i+1,i+1}^{(i)} \\ r_{i+2,i+1}^{(i)} \end{pmatrix}.$$

Prvky $r_{ii}^{(i)}$ a $a_{i+1,i}$ jsou nenulové. Pro $\beta = a_{i+1,i}/r_{ii}^{(i)}$ dostáváme požadovanou rovnost.

□

Poznámka. Na konci důkazu tvrdíme, že prvek $a_{i+1,i}$ v matici A je nenulový. To je ekvivalentní předpokladu, že je matice v úplném Hessenbergově tvaru. Pokud by ovšem matice A nebyla v úplném Hessenbergově tvaru, pak můžeme použít deflaci a úlohu rozdělit na dvě menší podúlohy.

2.7 Implicitní QR algoritmus s násobným shiftem

Nyní zobecníme postup pro M shiftů. Poznamenejme, že musí platit $M \leq n-1$, kde n je rozměr matice, jinak by výpočet neměl smysl. Většinou M volíme malé. V praxi se běžně využívají jeden, dva nebo tři shifty.

Zvolme $\rho_1, \dots, \rho_M \in \mathbb{C}$ a definujme

$$f(A) = (A-\rho_M I_n) \dots (A-\rho_1 I_n).$$

Na vytvoření bulge potřebujeme pouze první sloupec $f(A)$, označme ho

$$x \equiv f(A) e_1.$$

Ze skutečnosti, že $A - \rho_i$ jsou pro $i = 1, \ldots, M$ Hessenbergovy matice, plyne

$$(A - \rho_1 I_n)e_1 = \begin{pmatrix} * \\ * \\ 0 \\ 0 \\ \vdots \end{pmatrix}, \quad (A - \rho_1 I_n)(A - \rho_1 I_n)e_1 = \begin{pmatrix} * \\ * \\ * \\ 0 \\ \vdots \end{pmatrix},$$

kde hvězdičky jsou nenulová komplexní čísla. Z výše uvedeného je zřejmé, že x má prvních $M + 1$ prvků nenulových a zbylé prvky jsou nulové. Nyní najdeme M matic Givensových rotací $\tilde{G}_1, \ldots \tilde{G}_M$, které postupně nulují prvky vektoru x. (Na tento proces můžeme využít také jednu matici Householderovy reflexe.) Pak platí

$$\tilde{G}_1^* \ldots \tilde{G}_M^* x = \begin{pmatrix} * \\ 0 \\ \vdots \end{pmatrix}.$$

Označme $G_1 = \tilde{G}_M \ldots \tilde{G}_1$. Matici G_1 využijeme k vytvoření bulge. Stačí provést $A^{(1)} = G_1^* A G_1$.

Ilustrujme na matici 6×6, jak vypadá bulge pro $M = 2$ a $M = 3$:

$$A^{(1)} = \begin{pmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ \times & * & * & * & * & * \\ \times & \times & * & * & * & * \\ & & & * & * & * \\ & & & & * & * \end{pmatrix}, \qquad A^{(1)} = \begin{pmatrix} * & * & * & * & * & * \\ * & * & * & * & * & * \\ \times & * & * & * & * & * \\ \times & \times & * & * & * & * \\ \times & \times & \times & * & * & * \\ & & & & * & * \end{pmatrix}$$

Hvězdičky znázorňují strukturu matice před podobnostní transformací a křížky prvky, které přibyly vlivem transformace. Vlevo je případ $M = 2$ a vpravo případ $M = 3$.

Dále postupujeme procesem bulge chasing. Postupně hledáme jednotlivé matice $G_2, \ldots, G_{n-1}$ a pro $j = 2, \ldots, n - 1$ provádíme transformace

$$A^{(j)} = G_j^* A^{(j-1)} G_j.$$

Uvědomme si, že matice $G_2, \ldots G_{n-1}$ jsou (stejně jako G_1) složeny ze součinu M Givensových rotací. Poslední matice $\hat{A} = A^{(n-1)}$ je opět v Hessenbergově tvaru. Tím je dokončena jedna iterace implicitního QR algoritmu s M shifty. Můžeme zvolit nové shifty $\rho_1^{(2)}, \ldots, \rho_M^{(2)}$ a celý proces znovu opakovat.

Už jsme si ukázali, jak můžeme volit shifty pro $M = 1$ a $M = 2$, viz kapitoly 2.4 a 2.5. Obecně lze volit jako shifty posledních M diagonálních prvků matice A. Jelikož je M většinou malé, tak další možností je rozšíření myšlenky Wilkinsonova shiftu. Najdeme vlastní čísla podmatice rozměru $M \times M$, pro $M > 2$ například implicitním QR algoritmem s jedním shiftem. Ty pak použijeme jako shifty. Existují i jiné komplexnější shiftovací strategie, viz články David a Watkins (2006) a Vandebril a Watkins (2012). Vliv volby shiftovací strategie budeme diskutovat v následující kapitole, která se bude zabývat numerickými experimenty.

Na závěr uveďme, že bylo pozorováno, že přidání shiftu rozumné velikosti ($M < 6$) do implicitního QR algoritmu pozitivně ovlivní jeho konvergenční vlastnosti a celkovou rychlost výpočtu při implementaci na počítači. Bohužel přidání shiftů ztíží analýzu konvergence viz článek Parlett a Kahan (1968). Zatím se nepodařilo plně dokázat konvergenci implicitního QR algoritmu pro úspěšné shiftovací strategie.

3. Numerické experimenty

V předchozích kapitolách se nám podařilo odvodit implicitní QR algoritmus s vícenásobným shiftem, v literatuře známý také pod označením Francisův algoritmus. Tento algoritmus jsem naprogramoval v prostředí MATLAB a jeho implementaci popíšeme níže. V této kapitole otestujeme zmíněnou implementaci na testovacích maticích různých rozměrů a pokusíme se ukázat vlastnosti konvergence v závislosti na velikosti shiftu a samotné volbě shiftů. V tomto textu se zaměříme na demonstraci experimentů na několika vybraných testovacích maticích, nicméně je nutné zdůraznit, že algoritmus byl důkladně ověřen i na mnohem rozsáhlejším souboru matic. Experimenty byly prováděny na počítači s procesorem Intel i5-4460 s výkonem 3,20 GHz ve verzi R2021b prostředí MATLAB. Všechny zdrojové kódy a experimenty je možné najít na stránce `https://github.com/CirbusJ/Bakalarska-prace-Francisuv-algoritmus`.

3.1 Implementace Francisova algoritmu

Implementace je založena na poznatcích z knihy Aurentz a kol. (2018). Provedl jsem dvě implementace. Tyto implementace jsou identické, až na volbu shiftovací strategie. Jednou je algoritmus implementován s Wilkinsonovým shiftem (nalezení vlastních čísel podmatice $M \times M$) a po druhé volím shifty jako M posledních diagonálních prvků, kde M je velikost shiftu. Stručně je popišme. Vstupní parametry zahrnují matici, pro kterou provádíme hledání vlastních čísel, a požadovanou velikost shiftu. Další vstupní parametry jsou určeny pro rekurzivní volání algoritmu.

Při prvním zavolání algoritmus nastaví data jako maximální povolený počet iterací, než dojde k deflaci, a toleranci deflace. Dále pomocí vestavěné funkce `hess` převede vstupní matici na Hessenbergovu matici.

Poté algoritmus zkontroluje rozměr matice. Pokud je matice 1×1 nebo 2×2, tak spočte vlastní čísla explicitním vzorcem. V případě 1×1 položí aproximaci přímo rovnou tomuto číslu a v případě 2×2 využije vzorec pro počítání kořenů kvadratické funkce. Než algoritmus začne samotný proces vytvoření bulge a následný bulge chasing, zkontroluje, jestli je rozměr matice menší než šest, v takovém případě pak algoritmus nastaví velikost shiftu rovnou jedné. Používání moc velkého shiftu na matice malých rozměrů může působit komplikace s konvergencí.

Nyní algoritmus zvolí počáteční shift. Volbu shiftovací strategie provedeme před voláním algoritmu použitím jedné z implementací. Pomocí for cyklu provádíme proces vytvoření bulge a následný bulge chasing, přičemž po každém provedení obou procesů algoritmus zkontroluje, jestli není nějaký z poddiagonálních prvků v absolutní hodnotě menší než parametr tolerance deflace. Pokud nastane situace, kdy je jeden z poddiagonálních prvků v absolutní hodnotě menší než tolerance deflace, pak algoritmus provede deflaci a rekurzivně zavolá sám sebe na příslušné podmatice. V opačném případě algoritmus najde nový shift podle zvolené shiftovací strategie a vrátí se na začátek for cyklu.

Tímto postupem algoritmus najde všechna vlastní čísla vstupní matice. Výstupem algoritmu je tedy celé spektrum vstupní matice.

Poznámka. Procesy vytvoření bulge a bulge chasing jsou implementovány podob-

nostními transformacemi maticemi Givensových rotací, které sestrojíme stejně jako v kapitole 2.7. V knize Aurentz a kol. (2018) tyto matice nazývají core transformations.

Poznámka. Implementace s Wilkinsonovým shiftem na výpočet vlastních čísel podmatice $M \times M$ využívá implementaci s jedním shiftem zvoleným jako poslední diagonální prvek.

3.2 Testovací matice

Zde uvedeme matice, na kterých budeme provádět experimenty.

Matice s předepsanými vlastními čísly Uvažujme matici $A \in \mathbb{C}^{n\times n}$ s předepsanými vlastními čísly. Matici A sestrojíme pomocí podobnostní transformace. Vezmeme matici $D \in \mathbb{C}^{n\times n}$, která má na diagonále požadovaná vlastní čísla, a provedeme podobnostní transformaci náhodnou maticí S stejného rozměru. Náhodnou matici S získáme příkazem `rand(n)` z MATLABu. Pak stačí položit $A = S^{-1}DS$. Náhodná matice S je s pravděpodobností blízko jedné regulární, proto se jedná o podobnostní transformaci. Z tvrzení 7 máme rovnost spekter matic A a D.

Matice s citlivými vlastními čísly Matice uvedené v této sekci mají citlivá vlastní čísla. Využijeme sadu testovacích matic, které jsou vestavěné v MATLABu. Matice získaná příkazem `gallery('grcar',n)` je $n \times n$ Toeplitzovská s jedničkami na diagonále a na třech naddiagonálách a mínus jedničkami na poddiagonále.

Příkazem `gallery('lesp',n)` dostaneme matici rozměru $n \times n$, která je tridiagonální s reálnými vlastními čísly, která jsou rovnoměrně rozdělená. Citlivost jejích vlastních čísel roste exponenciálně pro záporná vlastní čísla.

3.3 Experiment 1

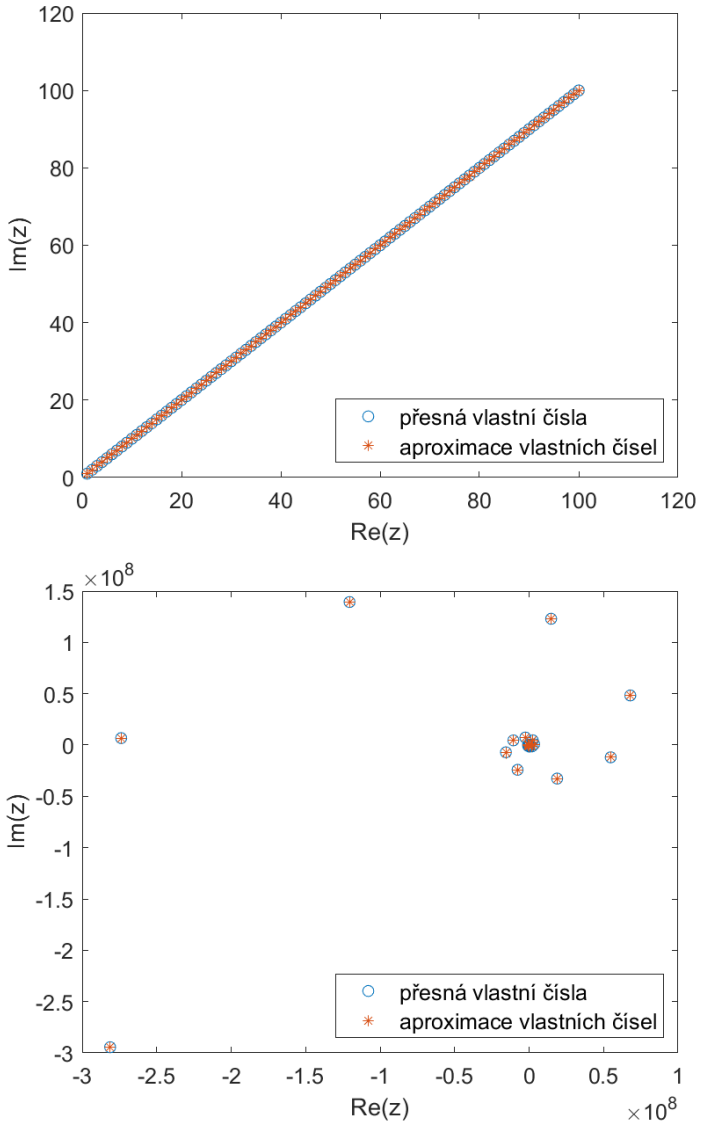
V tomto experimentu ověříme, že algoritmus skutečně najde vlastní čísla zadané matice.

Příklad (Matice s předepsanými vlastními čísly). Uvažujme matici $A \in \mathbb{R}^{n\times n}$ s předepsanými vlastními čísly. Zvolme rozměr matice $n = 100$ a vlastní čísla rovnoměrně rozdělená na intervalu [1,100], tedy $\lambda_1 = 1, \lambda_2 = 2, \ldots, \lambda_{100} = 100$. Aplikujme na matici A implementaci Francisova algoritmu s jedním shiftem a parametrem tolerance deflace nastaveným na $tol = 10^{-6}$.

Algoritmus spolehlivě najde aproximace $\mu_1, \ldots, \mu_{100}$ všech vlastních čísel. Pokud se podíváme na chybu získaných aproximací $|\lambda_j - \mu_j|$ pro $j = 1, \ldots, 100$, zjistíme, že všechna vlastní čísla jsou aproximována s přesností řádově stejnou jako zvolená tolerance $tol = 10^{-6}$. Poznamenejme, že přesná hodnota je závislá na náhodně vygenerované matici S. Existují náhodné matice S, pro které je chyba řádově větší, ale pořád dostáváme kvalitní aproximace. Přesnějších aproximací můžeme dosáhnout zmenšením parametru tolerance deflace.

Stejný výsledek dostaneme i pro matice s libovolnými rovnoměrně rozloženými vlastními čísly. Například $\lambda_1 = 2, \lambda_2 = 4, \ldots, \lambda_{100} = 200$ nebo s komplexními

$\lambda_1 = 1+i, \ldots, \lambda_{100} = 100+100i$, viz Obrázek 3.1 nahoře. Dokonce i pro matice s vlastními čísly $\lambda_1 = 1{,}2, \lambda_2 = (1{,}2)^2, \ldots, \lambda_{100} = (1{,}2)^{100}$ nebo $\lambda_1 = (1+1{,}1i), \lambda_2 = (1+1{,}1i)^2, \ldots, \lambda_{50} = (1+1{,}1i)^{50}$, které mají nerovnoměrně rozložená vlastní čísla, platí stejný závěr. Pro srovnání viz Obrázek 3.1 dole.

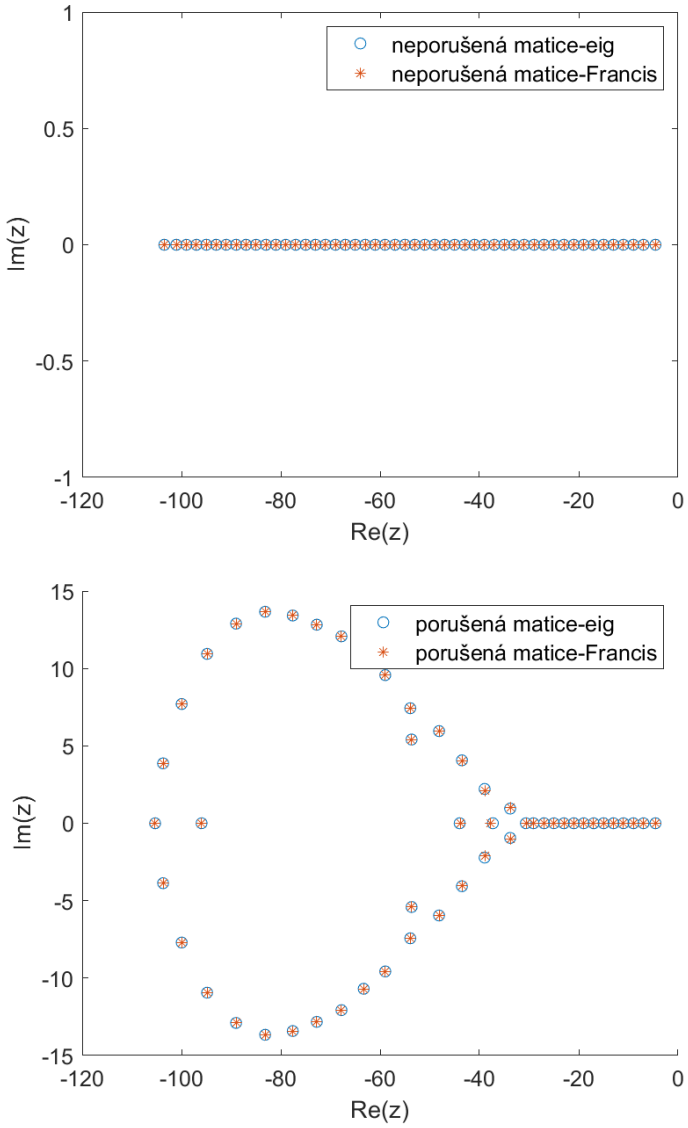


Obrázek 3.1: Přesná vlastní čísla a jejich aproximace spočtené Francisovým algoritmem pro matice s předepsaným spektrem $\lambda_1 = 1+i, \ldots, \lambda_{100} = 100+100i$ (nahoře); $\lambda_1 = (1+1{,}1i), \lambda_2 = (1+1{,}1i)^2, \ldots, \lambda_{50} = (1+1{,}1i)^{50}$ (dole).

Příklad (Matice s citlivými vlastními čísly). Otestujeme algoritmus na maticích `gallery('grcar',n)` a `gallery('lesp',n)`. Zde nemáme k dispozici přesná vlastní čísla. Proto budeme srovnávat získané aproximace naší implementací s aproximacemi z vestavěné funkce `eig` v MATLABu. Opět využijeme implementaci Francisova algoritmu s jedním shiftem a parametrem tolerance deflace nastaveným na $tol = 10^{-6}$.

Pro obě zmíněné matice algoritmus najde velice dobré aproximace. Pro volbu rozměru matic $n = 50$ a $n = 100$ je chyba získaných aproximací $\mu_1, \ldots, \mu_n$ vůči aproximacím $\sigma_1, \ldots, \sigma_n$ z funkce `eig` řádově 10^{-5}, tedy $|\sigma_j - \mu_j| \leq 10^{-5}$

pro $j = 1, \ldots, n$. Pokud matice `gallery('grcar',n)` a `gallery('lesp',n)` porušíme perturbací $E = 10^{-6}$`rand(n)`, dostaneme velice rozdílná spektra oproti neporušeným maticím. Ovšem aproximace $\mu_1, \ldots, \mu_n$ velice dobře odpovídají aproximacím $\sigma_1, \ldots, \sigma_n$, opět platí $|\sigma_j - \mu_j| \leq 10^{-5}$ pro $j = 1, \ldots, n$. A tedy naši implementaci Francisova algoritmu je vhodné použít i na matice s citlivými vlastními čísly. Srovnání pro matici `gallery('lesp',n)` vidíme na Obrázku 3.2.



Obrázek 3.2: Vlastní čísla aproximovaná vestavěnou funkcí `eig` a Francisovým algoritmem pro matici `gallery('lesp',50)` bez perturbace (vlevo), s perturbací (vpravo).

Z výše uvedených experimentů vidíme, že algoritmus opravdu najde spolehlivé aproximace vlastních čísel. Na závěr se zmíníme, že schopnost algoritmu hledat vlastní čísla byla otestována na mnohem větším vzorku matic.

3.4 Experiment 2

V tomto experimentu se pokusíme ukázat vliv volby shiftovací strategie. Budeme porovnávat Wilkinsonův shift (nalezení vlastních čísel podmatice $M \times M$) a zobecněný Rayleighův shift (posledních M prvků na diagonále). Zmíněné shiftovací strategie pro velikost shiftu jedna splývají, proto v této části budeme pracovat s velikostí shiftu dva. Tolerance deflace bude pro obě implementace nastavena na hodnotu $tol = 10^{-6}$.

Příklad (Matice s předepsanými vlastními čísly). Uvažujme matici $A \in \mathbb{R}^{n \times n}$ s předepsanými vlastními čísly. Začněme s případem $n = 100$ a vlastními čísly $\lambda_1 = 1, \lambda_2 = 2, \ldots, \lambda_{100} = 100$. Označme $\mu_1, \ldots, \mu_{100}$ aproximace získané pomocí implementace se zobecněným Rayleighovým shiftem a $\nu_1, \ldots, \nu_{100}$ aproximace získané pomocí implementace s Wilkinsonovým shiftem. Oba algoritmy najdou spolehlivé aproximace splňující $|\lambda_j - \mu_j| \leq 10^{-6}$ a $|\lambda_j - \nu_j| \leq 10^{-6}$ pro $j = 1, \ldots, 100$.

V následující tabulce uvedeme celkový počet iterací potřebný k nalezení všech aproximací. Jednou iterací zde rozumíme vytvoření bulge a následný bulge chasing. Poznamenejme, že jednotlivé iterace nejsou z výpočetního hlediska ekvivalentní, jelikož se provádějí na různých velikostech matic v důsledku rekurzivní povahy implementací.

	1.	2.	3.	4.	5.	6.
Rayleigh	227	344	268	235	213	285
Wilkinson	108	107	108	108	103	108

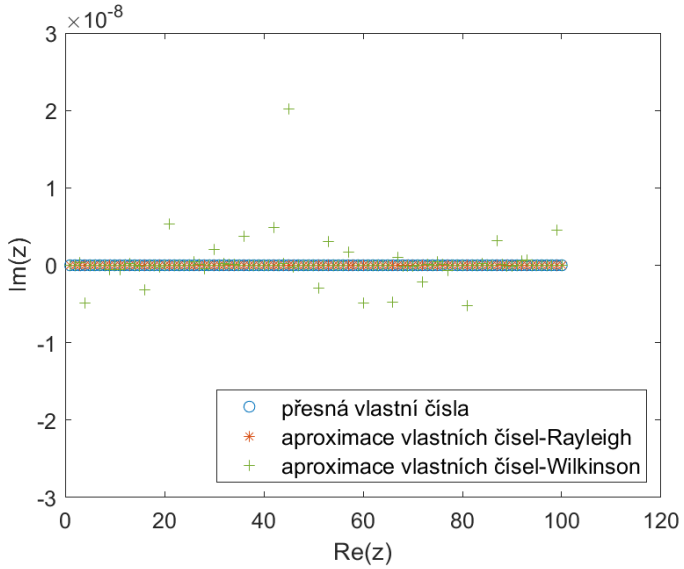
Tabulka 3.1: Celkový počet iterací pro různé shiftovací strategie a 6 testovacích matic s předepsaným (rovnoměrně rozloženým) spektrem.

Šestkrát jsme použili implementace na matici s vlastními čísly $\lambda_1 = 1, \lambda_2 = 2, \ldots, \lambda_{100} = 100$, ale pokaždé byla jiná náhodná matice, kterou jsme provedli podobnostní transformaci. Z tabulky je zřejmé, že Wilkinsonův shift je pro uvažovanou úlohu výhodnější shiftovací strategie, jelikož tato implementace nalezla aproximace v menším počtu iterací. Toto pozorování je lehce vysvětlitelné. Počítáním vlastních čísel podmatice 2×2 můžeme očekávat, že dostaneme přesnější aproximace skutečných vlastních čísel. Algoritmus tak konverguje rychleji. To nám poskytne ještě lepší aproximace v dalším kroku.

Nevýhodou Wilkinsova shiftu v obecnějším případě je právě nutnost počítat vlastní čísla podmatic. Pro velikost shiftu dva můžeme použít explicitní vzorec. Pro shift větší velikosti však musíme využít nějaký jiný způsob. V naší implementaci používáme pro tento účel Francisův algoritmus s jedním shiftem. Tím ovšem zvyšujeme výpočetní náročnost každé iterace.

Další problém může nastat, pokud jsou hledaná vlastní čísla reálná. Potom bychom rádi dostali i reálné aproximace. Ovšem při využití Wilkinsonova shiftu může ojediněle dojít k sestrojení komplexních shiftů v důsledku nahromadění zaokrouhlovacích chyb v podmatici, ze které shifty počítáme. Tím se celý výpočet dostane do komplexní aritmetiky, která je výpočetně náročnější. Výsledné aproximace pak také obsahují nenulovou komplexní složku. Tato imaginární část je

sice velmi malá, ale i přesto je to nežádoucí vlastnost. Numerický experiment, při kterém došlo k tomuto jevu, ilustruje Obrázek 3.3. Zatímco Rayleighův shift je vždy reálný (neboť aproximuje reálná vlastní čísla), pro Wilkinsonův shift to neplatí.



Obrázek 3.3: Přesná vlastní čísla a shifty počítané dvěma strategiemi. Vidíme, že Wilkinsonův shift je někdy komplexní.

Příklad (Náhodná matice). Použijme nyní naše algoritmy na náhodnou matici $S \in \mathbb{R}^{100\times 100}$ vygenerovanou příkazem `randn(100)`. Označme $\mu_1, \ldots, \mu_{100}$ aproximace získané pomocí implementace se zobecněným Rayleighovým shiftem a $\nu_1, \ldots, \nu_{100}$ aproximace získané pomocí implementace s Wilkinsonovým shiftem. Jelikož neznáme přesná vlastní čísla generovaných náhodných matic k porovnání přesnosti, použijeme aproximace $\sigma_1, \ldots, \sigma_{100}$ získané z vestavěné funkce `eig`.

Pro obě implementace dostáváme chybu vlastních čísel řádově stejnou jako je zvolená tolerance. Platí $|\sigma_j - \mu_j| \leq 10^{-6}$ a $|\sigma_j - \nu_j| \leq 10^{-6}$ pro $j = 1, \ldots, 100$. Srovnání aproximací vidíme na Obrázku 3.4.

Opět se podíváme na potřebný počet iterací na nalezení celého spektra. Iteracemi rozumíme stejný proces jako v předchozím příkladu 3.4. Postupně jsme

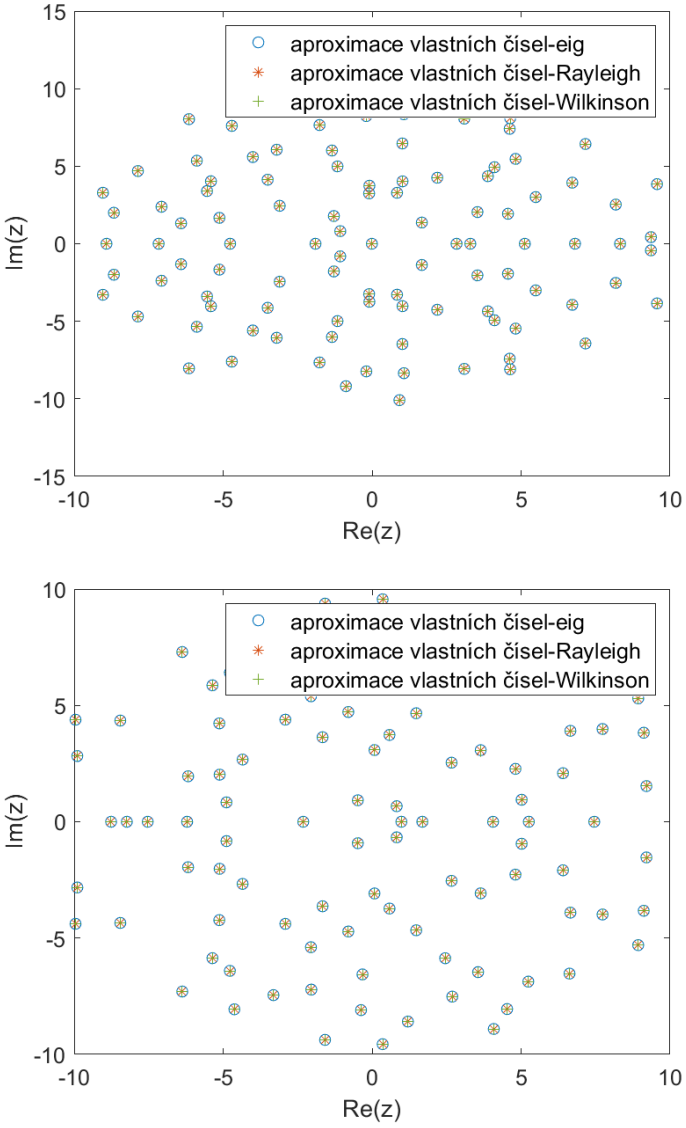
	1.	2.	3.	4.	5.	6.
Rayleigh	16 783	1 782	3 284	3 221	4 878	3 664
Wilkinson	157	153	160	158	149	155

Tabulka 3.2: Celkový počet iterací pro různé shiftovací strategie a 6 náhodných testovacích matic.

generovali náhodné matice a na každou použili obě shiftovací strategie. Tento postup jsme provedli šestkrát. Z výsledné tabulky je vidět, že pomocí Wilkinsonovy shiftovací strategie se nám podařilo získat aproximaci celého spektra za značně menší počet iterací.

Tak razantní rozdíl v počtu celkových iterací je způsoben následující skutečností. Náhodné matice, které generujeme, jsou reálné, ale obsahují komplexně

sdružené páry vlastních čísel. Celý výpočet s Rayleighovým zobecněným shiftem probíhá v reálné aritmetice, a tedy konvergence je velmi pomalá, protože používané shifty nejsou dobrými aproximacemi skutečných vlastních čísel. Naopak implementace s Wilkinsonovým shiftem (díky výpočtu vlastních čísel podmatice 2×2) do iterací zavede komplexní shifty, a proto tato implementace konverguje rychleji. Nevýhodou je opět přechod do komplexní aritmetiky, která je výpočetně náročnější.



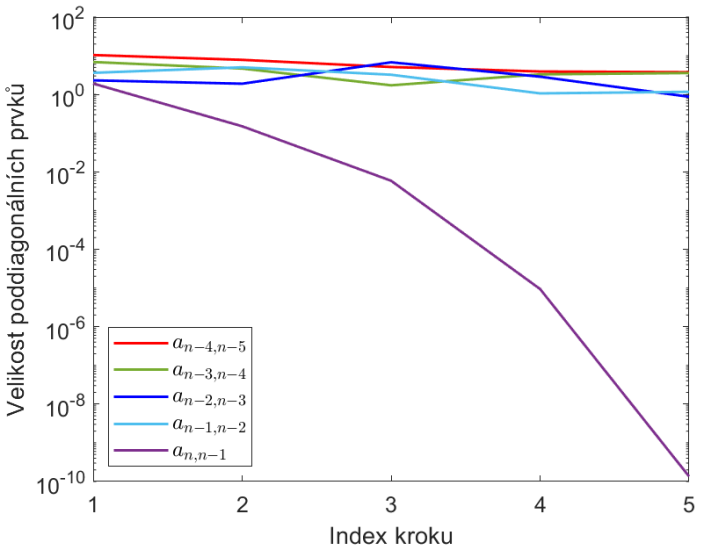
Obrázek 3.4: Příklady spekter náhodných matic a jejich spočtené aproximace.

3.5 Experiment 3

V tomto posledním experimentu otestujeme vliv velikosti shiftu na konvergenci poddiagonálních prvků iterační matice k nule. Budeme používat implementaci s Wilkinsonovým shiftem pro různé velikosti shiftů a sledovat velikost posledních pěti poddiagonálních prvků, než dojde k první deflaci. Parametr tolerance

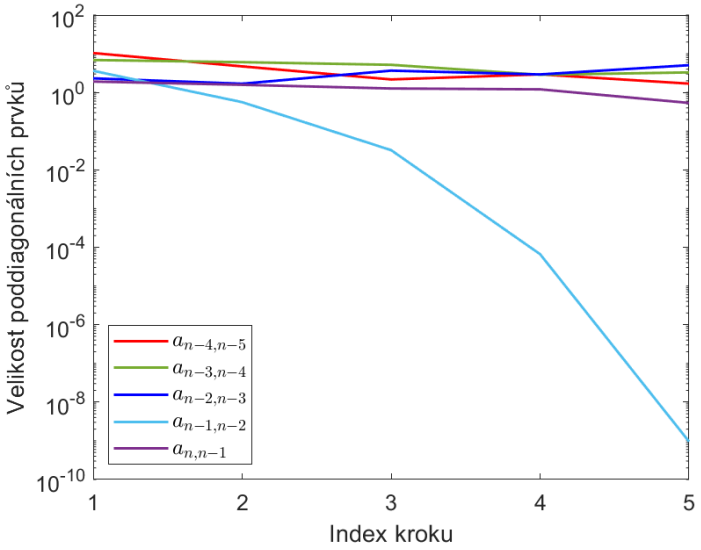
deflace nastavíme na hodnotu $tol = 10^{-6}$.

Příklad (Matice s předepsanými vlastními čísly). Mějme $A \in \mathbb{R}^{n\times n}$ s vlastními čísly $\lambda_1 = 1, \lambda_2 = 2, \ldots, \lambda_{100} = 100$, získanou podobnostní transformací náhodnou maticí. Označme $a_{n-4,n-5}$, $a_{n-3,n-4}$, $a_{n-2,n-3}$, $a_{n-1,n-2}$ a $a_{n,n-1}$ posledních pět poddiagonálních prvků matice A a sledujme, co se s nimi bude dít, než dojde k deflaci.



Obrázek 3.5: Konvergence poddiagonálních prvků pro jeden shift.

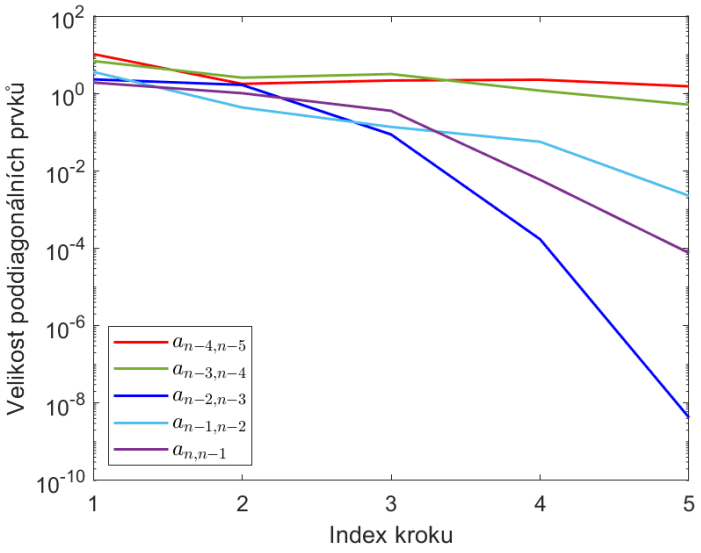
Obrázek 3.5 je podle našeho očekávání. Zvolili jsme jeden shift a úplně poslední poddiagonální prvek konverguje k nule. Ve chvíli, kdy dosáhne požadované tolerance, dojde k deflaci.



Obrázek 3.6: Konvergence poddiagonálních prvků pro dva shifty.

Pro dva shifty bychom očekávali, že dva poddiagonální prvky půjdou k nule, ale v obrázku 3.6 vidíme, že konverguje pouze jeden. Tentokrát to je ale prvek

$a_{n-1,n-2}$. Nejspíš je tato skutečnost způsobena tím, že jsme zvětšením shiftu přidali aproximaci dominantnějšího vlastního čísla.



Obrázek 3.7: Konvergence poddiagonálních prvků pro tři shifty.

Obrázek 3.7 je podle očekávání. Poslední tři poddiagonální prvky konvergují k nule a jeden z nich rychleji než ostatní. Pro větší počet shiftů konvergují různé počty poddiagonálních prvků a bylo by potřeba udělat důkladnější analýzu.

Poznamenejme, že to, kolik a jaké poddiagonální prvky v tomto příkladě konvergují, závisí na náhodně vygenerované matici, pomocí které provádíme podobnostní transformaci. Proces byl proveden mnohokrát a vždy konvergují různé počty prvků.

Na závěr se podívejme na počet iterací potřebný k nalezení spektra. Jednou iterací rozumíme vytvoření bulge a následný bulge chasing. Je důležité si uvědomit, že iterace pro různé velikosti shiftů jsou rozdílně výpočetně náročné. Pro shift velikosti M v jednom kroku bulge chasingu používáme M matic Givensových rotací.

Velikost shiftu	1	2	3	4	5
Počet iterací pro A_1	200	108	96	99	109
Počet iterací pro A_2	198	110	98	111	118

Tabulka 3.3: Celkový počet iterací pro různé velikosti shiftů a dvě testovací matice.

Matice A_1 a A_2 mají obě vlastní čísla $\lambda_1 = 1, \lambda_2 = 2, \ldots, \lambda_{100} = 100$, ale vznikly z různých náhodných matic. Z tabulky 3.3 vidíme, že potřebný počet iterací klesá s rostoucí velikostí shiftu. Ovšem od jisté velikosti začne opět růst. Mějme na paměti, že iterace jsou výpočetně náročnější pro větší shifty, a tedy můžeme usoudit, že velikost shiftu čtyři a víc je pro tento příklad nevhodná. Výpočetní náročnost pro větší shifty ilustrujme na maticích A_3 a A_4, které mají stejná vlastní čísla $\lambda_1 = 1, \lambda_2 = 2, \ldots, \lambda_{200} = 200$, ale jsou získány pomocí různých náhodných matic.

Velikost shiftu	1	2	3	4	5
Počet iterací pro A_3	386	209	185	195	545
Výpočetní čas [s] pro A_3	1,11	1,19	1,48	6,24	8,07
Počet iterací pro A_4	388	212	280	266	496
Výpočetní čas [s] pro A_4	1,03	1,22	1,36	1,49	4,24

Tabulka 3.4: Celkový počet iterací a výpočetní čas pro různé velikosti shiftů a dvě testovací matice.

Z tabulky 3.4 je nyní zřejmé, že jednotlivé iterace pro různé velikosti shiftu jsou opravdu rozdílně výpočetně náročné.

Podobně jako matice z předchozího příkladu se chová většina matic, které jsme v celé experimentální části uvedli (a i jiné). Velikost shiftu, od které se začne zvyšovat počet iterací, je pro všechny testované matice tři nebo čtyři. Z čehož můžeme vyvodit, že se nevyplatí pro naše implementace a shiftovací strategie používat větší velikost shiftu než čtyři.

Závěr

V této práci jsme se v kapitolách 1 a 2 zaměřili na odvození implicitního QR algoritmu s násobnými shifty na základě literatury. V průběhu tohoto odvození jsme uvedli řadu výsledků, které nám posloužily k důkladnému vysvětlení problému a poskytly nám hladký přechod až k požadovanému implicitnímu QR algoritmu. Při odvozování se nám podařilo zavést do problému shift, který urychlil rychlost konvergence QR algoritmu. V závěru kapitoly 2 jsme postup s jedním shiftem zobecnili na implicitní QR algoritmus s násobnými shifty a uvedli dvě shiftovací strategie, se kterými jsme experimentovali v kapitole 3.

V numerických experimentech jsme ověřili účinnost vlastní implementace implicitního QR algoritmu s násobnými shifty na různé testovací matice. Pokusili jsme se ukázat rozdíly plynoucí z volby shiftovacích strategií a také velikosti shiftu.

V prvním experimentu jsme zkoušeli, zda algoritmus opravdu umí najít vlastní čísla. Studovali jsme vliv parametru pro detekci nulového prvku na přesnost získaných aproximací. Podařilo se nám ukázat, že algoritmus spolehlivě najde dobré aproximace vlastních čísel různých matic, mezi nimiž jsou i matice s citlivými vlastními čísly.

V druhém experimentu jsme porovnávali dvě shiftovací strategie, a to Wilkinsonův shift a zobecněný Rayleighův shift. Ukázali jsme, že i přes vyšší výpočetní náročnost je Wilkinsonův shift výhodnější strategií pro většinu matic.

Ve třetím experimentu jsme se zaměřili na vliv velikosti shiftu na konvergenci poddiagonálních prvků matice a výpočetní náročnost celého algoritmu. Na uvedených příkladech jsme ukázali rostoucí výpočetní náročnost pro zvyšující se velikost shiftu. Zjistili jsme, že větší velikost shiftu působí značné ztížení analýzy numerického chování algoritmů. Bylo by třeba provést více experimentů a důkladnější analýzu celého problému.

Seznam použité literatury

Aurentz, J. L., Mach, T., Robol, L., Vandebril, R. a Watkins, D. S. (2018). *Core-Chasing Algorithms for the Eigenvalue Problem*. Fundamentals of Algorithms. SIAM, Philadelphia. ISBN 978-1-611975-33-8.

David, R. J. A. a Watkins, D. S. (2006). Efficient Implementation of the Multishift QR Algorithm for the Unitary Eigenvalue Problem. *SIAM Journal on Matrix Analysis and Applications*, **28**(3), 623–633. doi: 10.1137/050630787.

Golub, G. H. a Van Loan, C. F. (1996). *Matrix Computations*. Third Edition. The Johns Hopkins University Press, Baltimore. ISBN 0-8018-5414-8.

Parlett, B. N. a Kahan, W. (1968). On the convergence of a practical QR algorithm. In *IFIP Congress*. URL `https://api.semanticscholar.org/CorpusID:42668376`.

Vandebril, R. a Watkins, D. S. (2012). A Generalization of the Multishift qr Algorithm. *SIAM Journal on Matrix Analysis and Applications*, **33**(3), 759–779. doi: 10.1137/11085219X.

Watkins, D. S. (2002). *Fundamentals of Matrix Computations*. Second Edition. John Wiley and Sons, New York. ISBN 0-471-21394-2.

Watkins, D. S. (2008). The QR Algorithm Revisited. *SIAM Review*, **50**(1), 133–145.

Wilkinson, J. (1965). *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford. ISBN 9780198534037.