Numeričko rješavanje linearnih matričnih jednadžbi

Šestan, Jasmina

Master's thesis / Diplomski rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:559029

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-10-20**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Jasmina Šestan

NUMERIČKO RJEŠAVANJE LINEARNIH MATRIČNIH JEDNADŽBI

Diplomski rad

Voditelj rada:
doc. dr. sc. Zvonimir Bujanović

Zagreb, 2016

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Sadržaj

Sadržaj **iii**

Uvod **1**

1 Sylvesterova i Lyapunovljeva jednadžba **2**
1.1 Linearne matrične jednadžbe i linearni sustav 2
1.2 Jedinstvenost rješenja 4
1.3 Analitičke metode rješavanja 5

2 Motivacija i primjene **8**
2.1 LTI sustav 8

3 Algoritmi za matrice manjih dimenzija **11**
3.1 Schurova forma 11
3.2 Bartels–Stewartov algoritam 12
3.3 BS algoritam za Sylvesterovu jednadžbu 13
3.4 BS algoritam za Lyapunovljevu jednadžbu 18

4 Struktura matrica velikih dimenzija **22**
4.1 Rijetko popunjene matrice 22
4.2 Pozitivno semidefinitne matrice niskog ranga 24

5 Algoritmi za matrice velikih dimenzija **27**
5.1 ADI iteracije 27
5.2 Projekcijske metode 34
5.3 Primjer 40

Bibliografija **42**

Uvod

U raznim područjima matematike i primijenjenih znanosti možemo se susresti s linearnim matričnim jednadžbama. U teoriji dinamičkih sustava od posebne su važnosti Sylvesterova i Lyapunovljeva linearna matrična jednadžba.

Problem rješavanja Sylvesterove matrične jednadžbe sastoji se u pronalaženju matrice X takve da je $AX + XB = C$, pri čemu su zadane realne kvadratne matrice A, B i C. Posebnu vrstu ove jednadžbe čini Lyapunovljeva jednadžba kod koje je $B = A^T$. Cilj ovog diplomskog rada je dati pregled osnovnih rezultata o numeričkom rješavanju ovakvih jednadžbi, te napraviti usporedbu nekoliko glavnih algoritama. Implementacija algoritama će se napraviti pomoću biblioteka numpy i scipy u Pythonu.

U Poglavlju 1 pokazat ćemo kako linearne matrične jednadžbe možemo prikazati kao sustav linearnih jednadžbi i vidjet ćemo pod kojim uvjetima Sylvesterova jednadžba ima jedinstveno rješenje. Nakon toga, u Poglavlju 2, dajemo motivaciju za proučavanje ovih vrsta jednadžbi i opisujemo njihovu ulogu u analizi stabilnosti linearnih dinamičkih sustava.

U ostatku rada opisujemo neke od postojećih algoritama za pronalaženje rješenja matričnih jednadžbi i na njima je najveći naglasak u ovom radu. Najprije ćemo u Poglavlju 3 opisati algoritme koji su primijenjivi samo na matrične jednadžbe kod kojih su sve uključene matrice malih dimenzija. Podjela matrica na one malih dimenzija i one velikih dimenzija uvelike ovisi o arhitekturi računala na kojem se rješavaju navedene matrične jednadžbe. No, u radu ćemo matricama malih dimenzija smatrati one koje su veličine najviše nekoliko tisuća.

Teže je riješiti matrične jednadžbe velikih dimenzija zato što porastom dimenzija rastu prostorna i vremenska složenost algoritama za rješavanje. Međutim, velike matrične jednadžbe koje se javljaju u praksi obično imaju specijalnu strukturu koja nam olakšava izračunavanje rješenja. U Poglavlju 4 ćemo opisati neke tipične strukture, a u Poglavlju 5 iterativne metode razvijene za njihovo rješavanje.

Poglavlje 1

Sylvesterova i Lyapunovljeva jednadžba

U ovom poglavlju definirat ćemo Sylvesterovu i Lyapunovljevu jednadžbu i pokazat ćemo kako ih možemo prikazati kao sustav linearnih jednadžbi. Vidjet ćemo pod kojim uvjetima navedene jednadžbe imaju jedinstveno rješenje i kakva svojstva ono ima. Na kraju ćemo proučiti neke od analitičkih metoda rješavanja te vidjeti zašto su nam potrebne numeričke metode.

1.1 Linearne matrične jednadžbe i linearni sustav

Definicija 1.1.1. *Neka su* $A_i \in \mathbb{R}^{p\times q}$, $B_i \in \mathbb{R}^{r\times s}$, $C \in \mathbb{R}^{p\times s}$, $i = 1, 2, \ldots, k$. *Jednadžba oblika*

$$\sum_{i=1}^{k} A_i X B_i = C$$

zove se linearna matrična jednadžba, a matrica $X \in \mathbb{R}^{q\times r}$ *njezino rješenje.*

Matrice A_i i B_i nazivamo matricama koeficijenata matrične jednadžbe, a matricu C ćemo ponekad navoditi kao matricu s desne strane jednadžbe.

Naveli smo u Uvodu da su od posebnog zanimanja linearne matrične jednadžbe u kojima je $k = 2$.

Definicija 1.1.2. *Neka su* $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{m\times m}, C \in \mathbb{R}^{n\times m}$. *Jednadžba oblika*

$$AX + XB = C$$

zove se ***Sylvesterova matrična jednadžba****.*

Lyapunovljeva jednadžba je poseban slučaj Sylvesterove jednadžbe (kad je $B = A^T$).

Definicija 1.1.3. *Neka su $A, C \in \mathbb{R}^{n\times n}$. Jednadžba oblika*

$$AX + XA^T = C$$

*zove se **Lyapunovljeva matrična jednadžba**.*

Za analizu linearnih matričnih jednadžbi pokazuje se korisnim definirati tzv. Kroneckerov produkt matrica i operator vektorizacije.

Definicija 1.1.4. *Za matrice $A \in \mathbb{R}^{m\times n}$ i $B \in \mathbb{R}^{p\times q}$ **Kroneckerov produkt** matrica $A = [a_{ij}]$ i B (u oznaci $A \otimes B$) definiran je s*

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp\times nq}.$$

Operator vektorizacije je linearni operator koji matricu preslikava u vektor u kojem su stupci matrice nanizani uzastopno jedan ispod drugog. Dajemo precizniju definiciju.

Definicija 1.1.5. *Operator vektorizacije matrice $A \in \mathbb{R}^{m\times n}$, u oznaci $vec(A)$, definiran je s*

$$vec(A) = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix},$$

pri čemu su $a_1, a_2, \ldots, a_n$ stupci matrice A.

Uzmimo Sylesterovu jednadžbu $AX + XB = C$, pri čemu su a_i, b_i, c_i, x_i redom stupci matrica A, B, C, X. Lako se vidi da za i-ti stupac matrice C vrijedi

$$c_i = Ax_i + Xb_i = Ax_i + \sum_{j=1}^{m} b_{ji}x_j.$$

Dakle, Sylvesterova jednadžba se može zapisati kao $mn \times mn$ linearni sustav

$$\begin{bmatrix} A + b_{11}I & b_{21}I & \cdots & b_{m1}I \\ b_{12}I & A + b_{22}I & \cdots & b_{m2}I \\ \vdots & & \ddots & \vdots \\ b_{1m}I & b_{2m}I & \cdots & A + b_{mm}I \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}.$$

Time smo dokazali sljedeću propoziciju.

Propozicija 1.1.6. *Sylvesterova jednadžba $AX+XB = C$ je ekvivalentna linearnom sustavu*

$$[(I_m \otimes A) + (B^T \otimes I_n)]vec(X) = vec(C).$$

Analogno, Lyapunovljeva jednadžba $AX + XA^T = C$ se može zapisati kao linearni sustav

$$[(I_m \otimes A) + (A \otimes I_n)]vec(X) = vec(C).$$

Na ovaj način možemo pokazati da se svaka linearna matrična jednadžba može zapisati kao običan sustav linearnih jednadžbi čije su nepoznanice elementi matrice X. Međutim, zbog specijalne strukture tih linearnih sustava, postoji zasebna teorija koja proučava postojanje i svojstva njihovih rješenja, kao i specijalizirani algoritmi koji ih izračunavaju.

1.2 Jedinstvenost rješenja

Teorem 1.2.1. *(Jedinstvenost rješenja Sylvesterove jednadžbe)*
Neka su $\lambda_1, \lambda_2, \ldots, \lambda_n$ svojstvene vrijednosti matrice $A \in \mathbb{R}^{n\times n}$ i $\mu_1, \mu_2, \ldots, \mu_m$ svojstvene vrijednosti matrice $B \in \mathbb{R}^{m\times m}$.
Tada Sylvesterova jednadžba $AX + XB = C$ ima jedinstveno rješenje X ako i samo ako je

$$\lambda_i + \mu_j \neq 0,$$

za svaki $i = 1, 2, \ldots, n$ i $j = 1, 2, \ldots, m$.

Drugim riječima, Sylvesterova matrična jednadžba ima jedinstveno rješenje ako i samo ako A i $-B$ nemaju zajedničke svojstvene vrijednosti.

Dokaz. Iz Propozicije (1.1.6) slijedi da Sylvesterova jednadžba ima jedinstveno rješenje ako i samo ako je $P = (I_m \otimes A) + (B^T \otimes I_n)$ regularna matrica.

Uzmimo jednu proizvoljnu svojstvenu vrijednost λ_i matrice A i jednu proizvoljnu svojstvenu vrijednost μ_j matrice B. Tada za neke $x \neq 0$ i $y \neq 0$ vrijedi:

$$Ax = \lambda_i x$$
$$By = \mu_j y.$$

Pronađimo svojstvene vrijednosti matrice P. Za to će nam biti potreban iskaz $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$ koji se lako može dokazati koristeći definiciju Kroneckerovog produkta. Uočimo još da je $(y \otimes x) \neq 0$ za $x, y \neq 0$.

$$\begin{aligned} P(y \otimes x) &= (I_m \otimes A + B^T \otimes I_n)(y \otimes x) \\ &= (I_m \otimes A)(y \otimes x) + (B^T \otimes I_n)(y \otimes x) \\ &= (I_m y) \otimes (Ax) + (B^T y) \otimes (I_n x) \\ &= y \otimes (\lambda_i x) + (\mu_j y) \otimes x \\ &= (\lambda_i + \mu_j)(y \otimes x) \end{aligned}$$

Dakle, svojstvena vrijednost matrice P je jednaka zbroju jedne svojstvene vrijednosti matrice A i jedne svojstvene vrijednosti matrice B.

Budući da je determinanta matrice jednaka umnošku svojstvenih vrijednosti, slijedi da je P regularna (to jest, da Sylvesterova jednadžba ima jedinstveno rješenje) ako i samo ako vrijedi $\lambda_i + \mu_j \neq 0$ za svaki $i = 1, 2, \ldots, n$ i $j = 1, 2, \ldots, m$. □

Analogno, Lyapunovljeva jednadžba ima jedinstveno rješenje ako i samo ako A i $-A$ nemaju zajedničkih svojstvenih vrijednosti:

Korolar 1.2.2. *(Jedinstvenost rješenja Lyapunovljeve jednadžbe)*
Neka su $\lambda_1, \lambda_2, \ldots, \lambda_n$ svojstvene vrijednosti matrice $A \in \mathbb{R}^{n\times n}$. Tada Lyapunovljeva jednadžba $AX + XA^T = C$ ima jedinstveno rješenje ako i samo ako vrijedi $\lambda_i + \lambda_j \neq 0$ za svaki $i, j = 1, 2, \ldots, n$.

Često se u primjenama proučava Lyapunovljeva jednadžba u kojoj je matrica C simetrična, to jest vrijedi $C = C^T$. Ako Lyapunovljeva jednadžba ima jedinstveno rješenje X i ako je C simetrična matrica, onda je i X simetrična matrica. Naime, ako transponiramo Lyapunovljevu jednadžbu $AX + XA^T = C$ dobivamo $AX^T + X^TA^T = C^T = C$, a budući da je X jedinstveno rješenje, slijedi $X = X^T$.

1.3 Analitičke metode rješavanja

Budući da su Sylvesterova i Lyapunovljeva jednadžba od velikog interesa, postoji puno metoda za njihovo rješavanje, kako numeričkih tako i analitičkih. U nastavku su navedene neke analitičke metode, odnosno eksplicitni izrazi za rješenja Sylvesterove i Lyapunovljeve jednadžbe.

Integral eksponencijalnih funkcija matrica:

Uvedimo najprije pojam stabilne matrice s kojim se često susrećemo u analizi linearnih matričnih jednadžbi.

Definicija 1.3.1. *Kažemo da je kvadratna matrica **stabilna** ako svaka njezina svojstvena vrijednost λ ima negativan realni dio, $Re(\lambda) < 0$.*

Ako su matrice koeficijenata A i B Sylvesterove jednadžbe $AX + XB = C$ stabilne, onda se rješenje može napisati kao integral eksponencijalnih funkcija matrica.

Propozicija 1.3.2. *Neka je $AX + XB = C$ Sylvesterova jednadžba takva da su spektri od A i $-B$ disjunktni i neka su matrice A i B stabilne. Tada je jedinstveno rješenje Sylvesterove jednadžbe dano sa:*

$$X = -\int_0^\infty e^{At} C e^{Bt} dt.$$

Dokaz. Pretpostavimo da su A i B stabilne matrice. Pokazat ćemo u Propoziciji 2.1.2 da tada vrijedi $\lim_{t\to\infty} e^{At} = 0$ i $\lim_{t\to\infty} e^{Bt} = 0$. Tvrdimo da je

$$X = -\int_0^\infty e^{At}Ce^{Bt}dt$$

rješenje Sylvesterove jednadžbe. Naime, parcijalnom integracijom dobivamo:

$$\frac{d}{dt}(e^{At}Ce^{Bt}) = Ae^{At}Ce^{Bt} + e^{At}Ce^{Bt}B$$
$$e^{At}Ce^{Bt}\Big|_0^\infty = \int_0^\infty Ae^{At}Ce^{Bt}dt + \int_0^\infty e^{At}Ce^{Bt}Bdt$$

Zbog $\lim_{t\to\infty} e^{At} = 0$ i $\lim_{t\to\infty} e^{Bt} = 0$ slijedi

$$C = AX + XB.$$

Dakle, X je rješenje Sylvesterove jednadžbe, a budući da matrice A i B nemaju zajedničkih svojstvenih vrijednosti slijedi da je ono i jedinstveno. □

Napomenimo da je ovaj izraz za rješenje Sylvesterove jednadžbe bitan kao teorijski rezultat, ali nije pogodan za numeričko računanje pa se ne primjenjuje u praksi pri pronalaženju rješenja matričnih jednadžbi.

Krivuljni integral:
Neka Sylvesterova jednadžba $AX + XB = C$ ima jedinstveno rješenje. Ono je tada dano sa:

$$X = \frac{1}{2\pi i}\int_\Gamma (A - zI)^{-1}C(B + zI)^{-1}dz$$

pri čemu je Γ krivulja koja sadrži sve svojstvene vrijednosti matrice A, ali nijednu svojstvenu vrijednost od B. Detalji o ovoj metodi mogu se naći u [1].

Inverz matrice A:
Jameson je u svom članku [6] predložio rješavanje Sylvesterove i Lyapunovljeve jednadžbe pomoću inverza matrice A ili B. Ukratko opisujemo metodu za Lyapunovljevu jednadžbu $AX + XA^T = C$.
Definiramo niz matrica $Q_k,\ k = 1, 2, \ldots, n$ na sljedeći način:

$$Q_{-1} = 0, \quad Q_0 = C, \qquad Q_k = AQ_{k-1} - Q_{k-1}A^T + AQ_{k-2}A^T.$$

Tada se može pokazati da je za karakteristični polinom matrice A dan sa $k_A(\lambda) = det(\lambda I - A) = \lambda^n + c_1\lambda^{n-1} + \ldots + c_n$ i za matricu P definiranu kao $P = A^n - c_1A^{n-1} + \ldots + (-1)^n c_n I$, rješenje Lyapunovljeve jednadžbe dano sa:

$$X = P^{-1}(Q_n - c_1Q_{n-1} + \ldots + (-1)^n c_n Q_0).$$

Kad se tek počelo proučavati linearne matrične jednadžbe, metode za rješavanje su se oslanjale na gornje analitičke izraze. Međutim, vidjelo se da te metode nisu pogodne za numeričko računanje, čak i za male dimenzije matrica A, B i C. Na primjer, za zadnju metodu numerički eksperimenti pokazuju da je za pogodno odabrane matrice veličine 14×14 greška prilikom izvršavanja metode na računalu velika gotovo koliko i samo rješenje (vidi [3]). Iz tog razloga razvile su se razne numeričke metode za rješavanje i njima će biti posvećen ostatak rada.

Poglavlje 2

Motivacija i primjene

Sylvesterova i Lyapunovljeva linearna matrična jednadžba su bitne u teoriji linearnih sustava i teoriji upravljanja. Također, javljaju se u problemima rješavanja nelinearnih matričnih jednadžbi, na primjer Riccatijeve jednadžbe. Među ostalim, koriste se i u redukciji modela te u pronalaženju svojstvenih vrijednosti i svojstvenih vektora, a pojavljuju se i u problemima vezanim za adaptivnu optiku i obradu slika. Detalji o ovim problemima, kao i dodatne primjene linearnih matričnih jednadžbi mogu se naći u [9] i referencama unutar istog.

U nastavku ćemo ukratko pojasniti ulogu linearnih matričnih jednadžbi u analizi stabilnosti linearnih dinamičkih sustava.

2.1 LTI sustav

Linearan vremenski invarijantan (LTI) dinamički sustav dan je sustavom linearnih diferencijalnih jednadžbi:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t), \quad x(0) = x_0 \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

pri čemu je $x : \mathbb{R} \to \mathbb{R}^n$ vektorska funkcija stanja, $u : \mathbb{R} \to \mathbb{R}^k$ funkcija ulaza i $y : \mathbb{R} \to \mathbb{R}^p$ funkcija izlaza. Svim funkcijama je varijabla vrijeme.
Matricu $A \in \mathbb{R}^{n\times n}$ zovemo matrica stanja, $B \in \mathbb{R}^{n\times k}$ matrica ulaza, $C \in \mathbb{R}^{p\times n}$ matrica izlaza te $D \in \mathbb{R}^{p\times k}$ matrica prijenosa sustava. Matrice sustava A, B, C, D su konstantne pa zato kažemo da je sustav vremenski invarijantan.

Uvedimo najprije pojam stabilnosti sustava.

Definicija 2.1.1. *Kažemo da je linearni autonomni sustav $\dot{x}(t) = Ax(t)$, $x(0) = x_0$ stabilan ako vrijedi*

$$\lim_{t\to\infty} x(t) = 0.$$

U Poglavlju 1 smo rekli da je matrica stabilna ako svaka njezina svojstvena vrijednost ima negativan realni dio. Pogledajmo kakva je veza između stabilnosti sustava i stabilnosti pripadne matrice.

Propozicija 2.1.2. *Matrica A je stabilna matrica ako i samo ako je linearni autonomni sustav $\dot{x}(t) = Ax(t)$, $x(0) = x_0$ stabilan.*

Dokaz. Lako se pokaže da je rješenje sustava $\dot{x}(t) = Ax(t)$, $x(0) = x_0$ matrica

$$x(t) = e^{At}x_0.$$

Zapišimo matricu A u Jordanovoj formi: $A = XJX^{-1}$, pri čemu je J blok dijagonalna matrica. Tada je $e^{At} = Xe^{Jt}X^{-1}$, a matrica e^{Jt} je također blok dijagonalna.

Uzmimo jedan njezin blok e^{Ft} dimenzije k, gdje je $F = \lambda I + N$ Jordanov blok. Matrica N ima svuda nule osim na prvoj naddijagonali gdje su jedinice, pa je ona nilpotentna matrica indeksa k.

Kako je $e^{Ft} = \sum_{n=0}^{\infty} \frac{F^n t^n}{n!}$, kad uvrstimo $F = \lambda I + N$ vidimo da je e^{Ft} linearna kombinacija matrica $e^{\lambda It}$, $te^{\lambda It}$, $t^2e^{\lambda It}$, $\ldots$, $t^{k-1}e^{\lambda It}$.

Za elemente matrice $e^{\lambda It}$ vrijedi

$$e^{\lambda t} = e^{\alpha t + i\beta t} = e^{\alpha t}e^{i\beta t} = e^{\alpha t}(\cos\beta t + i\sin\beta t).$$

$\lambda = \alpha + i\beta$ je svojstvena vrijednost matrice A koja je stabilna (tj. $\alpha < 0$) pa vrijedi $\lim_{t\to\infty} e^{\lambda t} = 0$. Lako se pokaže da je $\lim_{t\to\infty} t^{\ell}e^{\lambda_i t} = 0$, za svaki $i = 1, 2, \ldots, n$ i za bilo koji $\ell \in \mathbb{N}$, pa slijedi da matrica e^{Ft} konvergira u nul matricu.

Budući da ovaj rezultat vrijedi za svaki dijagonalni blok oblika e^{Ft} matrice e^{Jt}, slijedi $\lim_{t\to\infty} e^{Jt} = 0$, a onda je i $\lim_{t\to\infty} e^{At} = 0$. Analogno se pokaže da vrijedi i obrat. □

Definicija 2.1.3. *Za LTI sustav kažemo da je stabilan ako je pripadni linearni autonomni sustav $\dot{x}(t) = Ax(t)$, $x(0) = x_0$ stabilan.*

Iz prethodne propozicije slijedi sljedeća tvrdnja.

Korolar 2.1.4. *LTI sustav je stabilan ako i samo ako je matrica stanja A stabilna matrica.*

U teoriji linearnih sustava, pa posebno i LTI sustava, bitni su pojmovi kontrolabilnosti (upravljivosti) i opservabilnosti (opažajnosti). Ukratko rečeno, LTI sustav je je kontrolabilan ako pravilnim odabirom ulaza iz bilo kojeg stanja možemo doći u bilo koje stanje u bilo kojem trenutku. Ako možemo u bilo kojem trenutku na temelju izlaza odrediti jedinstveno početno stanje sustava, pretpostavljajući da nema ulaza, onda kažemo da je LTI sustav opservabilan.

Gramijan kontrolabilnosti definiramo sa

$$\mathcal{P} = \int_0^\infty e^{At} B B^T e^{A^T t} dt,$$

a gramijan opservabilnosti sa

$$\mathcal{Q} = \int_0^\infty e^{A^T t} C^T C e^{At} dt.$$

Sljedeći teorem objašnjava ulogu Lyapunovljeve jednadžbe u analizi stabilnosti LTI sustava.

Teorem 2.1.5. *Ako je LTI sustav stabilan, onda gramijani kontrolabilnosti i opservabilnosti postoje i oni su rješenja Lyapunovljevih jednadžbi*

$$A\mathcal{P} + \mathcal{P}A^T = -BB^T$$
$$A^T\mathcal{Q} + \mathcal{Q}A = -C^TC.$$

Dokaz. Iz Korolara 2.1.4 slijedi da je matrica stanja sustava A u jednadžbama LTI sustava stabilna. Analogno dokazu Propozicije 1.3.2 možemo pokazati da matrice $\mathcal{P}$ i $\mathcal{Q}$ postoje i da su rješenja navedenih Lyapunovljevih jednadžbi. □

Pokazat ćemo u Poglavlju 4 da su gramijani $\mathcal{P}$ i $\mathcal{Q}$ uvijek simetrične pozitivno semidefinitne matrice. Nadalje, ako je LTI sustav kontrolabilan, odnosno opservabilan, onda još vrijedi da je gramijan kontrolabilnosti $\mathcal{P}$, odnosno gramijan opservabilnosti $\mathcal{Q}$, pozitivno definitna matrica. Može se pokazati da vrijedi i obrat: ako je sustav stabilan, a $\mathcal{P}$ pozitivno definitna, onda je sustav kontrolabilan. Slično, ako je sustav stabilan, a $\mathcal{Q}$ pozitivno definitna, onda je sustav opservabilan.

Od interesa je još i unakrsni gramijan koji se definira kao

$$\mathcal{X} = \int_0^\infty e^{At} B C e^{At} dt.$$

Za njega se može pokazati da je rješenje Sylvesterove jednadžbe

$$A\mathcal{X} + \mathcal{X}A = -BC$$

ukoliko je matrica A stabilna.

Poglavlje 3

Algoritmi za matrice manjih dimenzija

Budući da se Sylvesterova jednadžba $AX + XB = C$ može zapisati kao linearni sustav, direktan način za njezino rješavanje je primjena Gaussovih eliminacija s parcijalnim pivotiranjem na ekvivalentan linearni sustav $[(I_m \otimes A) + (B^T \otimes I_n)]vec(X) = vec(C)$. Međutim, rješavanje sustava pomoću Gaussovih eliminacija s pripadnom matricom veličine $nm \times nm$ je vremenske složenosti $O(n^3m^3)$ te prostorne složenosti $O(n^2m^2)$.

Dakle, računski je skupo pa treba naći bolje metode ili transformirati sustav u jednostavniji i na njemu primijeniti Gaussove eliminacije. Jedan od najčešće korištenih algoritama za matrične jednadžbe manjih dimenzija je Bartels–Stewartov algoritam čija je glavna ideja transformacija matrične jednadžbe u jednostavniji oblik.

Na početku ovog poglavlja su definirane posebne forme matrica koje su se pokazale pogodnima za transformaciju matričnih jednadžbi. Nakon toga su opisani koraci Bartels–Stewartovog algoritma i verzije algoritma za Sylvesterovu jednadžbu i simetričnu Lyapunovljevu jednadžbu. Također, prikazane su implementacije tih algoritama u Pythonu.

3.1 Schurova forma

U ovom poglavlju koristit ćemo nekoliko posebnih vrsta matrica pa ovdje navodimo njihove definicije i svojstva. Najprije se podsjetimo da je kompleksna matrica U unitarna ako vrijedi $UU^* = U^*U = I$, a realna matrica Q ortogonalna ako je $QQ^T = Q^TQ = I$.

Teorem 3.1.1. *(Schurova dekompozicija)*
Za svaku $n \times n$ kompleksnu matricu A postoje unitarna matrica $U \in \mathbb{C}^{n\times n}$ i gornje trokutasta matrica $T \in \mathbb{C}^{n\times n}$ takve da vrijedi

$$U^*AU = T,$$

pri čemu su na dijagonali matrice T svojstvene vrijednosti matrice A.
Kažemo da je matrica T ***Schurova forma*** *matrice A.*

Budući da realna matrica može imati kompleksne svojstvene vrijednosti, matrice U i T iz gornjeg teorema mogu biti kompleksne i za realnu matricu A. Da to izbjegnemo, na sličan način definiramo realnu Schurovu formu koja nije nužno prava trokutasta matrica, već blok gornje trokutasta matrica s blokovima veličine najviše 2×2.

Teorem 3.1.2. *(Realna Schurova dekompozicija)*
Za svaku $n \times n$ realnu matricu A postoje ortogonalna matrica $Q \in \mathbb{R}^{n\times n}$ i blok gornje trokutasta matrica $R \in \mathbb{R}^{n\times n}$ takve da vrijedi

$$Q^T A Q = R = \begin{bmatrix} R_{[11]} & R_{[12]} & \cdots & R_{[1k]} \\ 0 & R_{[22]} & \cdots & R_{[2k]} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & R_{[kk]} \end{bmatrix}$$

gdje je svaki dijagonalni blok $R_{[ii]}$ dimenzije 1×1 ili 2×2. Trivijalni 1×1 blokovi su realne svojstvene vrijednosti od A, a svojstvene vrijednosti svakog 2×2 bloka su jedan par kompleksno konjugiranih svojstvenih vrijednosti od A.
Kažemo da je R ***realna Schurova forma*** *matrice A.*

Definicija 3.1.3. *Kažemo da je $n \times n$ matrica H u Hessenbergovoj formi (**Hessenbergova matrica**) ako je $H_{ij} = 0$ za $i > j + 1$.*

Teorem 3.1.4. *(Hessenbergova dekompozicija)*
Za svaku $n \times n$ realnu matricu A postoje ortogonalna matrica $Q \in \mathbb{R}^{n\times n}$ i Hessenbergova matrica $H \in \mathbb{R}^{n\times n}$ takve da vrijedi

$$Q^T A Q = H.$$

Dokazi ovih teorema mogu se naći npr. u [3].

3.2 Bartels–Stewartov algoritam

Bartels i Stewart su 1972. godine razvili efikasnu metodu za numeričko rješavanje Sylvesterove jednažbe malih dimenzija koja se, uz neke modifikacije, često koristi i danas. Generalno se Bartels–Stewartov algoritam (BS algoritam) može opisati u četiri koraka:

1. Transformirati matrice A i B u "jednostavniji" oblik koristeći sličnost matrica:

$$\tilde{A} = U^{-1} A U \text{ i } \tilde{B} = V^{-1} B V.$$

2. Izračunati $\tilde{C} = U^{-1} C V$.

3. Riješiti modificirani sustav $\tilde{A}Y + Y\tilde{B} = \tilde{C}$ za Y.

4. Izračunati $X = UYV^{-1}$.

Nameće se pitanje koji je to "jednostavniji" oblik matrice pogodan za rješavanje sustava. Znamo da je lako primjeniti Gaussove eliminacije na trokutastu matricu sustava, pa je zato za "jednostavniji" oblik pogodna Schurova forma. Kada Schurova forma ima kompleksne elemente računski je isplativija transformacija u blok trokutastu realnu Schurovu formu. Za Sylvesterovu jednadžbu u kojoj je velika razlika između dimenzija matrica A i B još je računski bolje transformiranje veće jednadžbe u Hessenbergovu formu, a manje u Schurovu formu.

Neki drugi oblici matrica nam se također mogu nametnuti kao potencijalni dobar izbor za "jednostavniji" oblik, na primjer dijagonalna i Jordanova kanonska forma. U [3] možemo naći razloge zašto se navedene forme trebaju izbjegavati u numeričkom računanju te zašto je baš Schurova forma dobar izbor.

U sljedećem potpoglavlju opisujemo detaljnije Bartels–Stewartov algoritam za Sylvesterovu jednadžbu koristeći realnu Schurovu formu. Također, analizirat ćemo verziju algoritma koji koristi Hessenbergovu formu. Nakon toga, u zadnjem potpoglavlju, opisujemo verziju algoritma za simetričnu Lyapunovljevu jednadžbu.

3.3 BS algoritam za Sylvesterovu jednadžbu

Promatramo Sylvesterovu jednadžbu $AX + XB = C$, pri čemu su $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$, $C \in \mathbb{R}^{n\times m}$. Pretpostavljamo da ona ima jedinstveno rješenje.

Prvi korak Bartels–Stewartovog algoritma za Sylvesterovu jednadžbu je prebaciti matrice A i B^T u realne Schurove forme, to jest dobiti blok gornje trokutaste matrice $H \in \mathbb{R}^{n\times n}$ i $V \in \mathbb{R}^{m\times m}$ oblika

$$H = V^T AV,$$
$$R = U^T B^T U,$$

pri čemu su $V \in \mathbb{R}^{n\times n}$ i $U \in \mathbb{R}^{m\times m}$ ortogonalne matrice.
Ako Sylvesterovu jednadžbu $AX + XB = C$ pomnožimo s lijeve strane s V^T i s desne strane s U dobivamo

$$V^T AXU + V^T XBU = V^T CU.$$

Budući da su U i V ortogonalne matrice, prethodna jednadžba je ekvivalentna jednadžbi

$$V^T AVV^T XU + V^T XUU^T BU = V^T CU.$$

Uz oznake

$$Y = V^T XU \in \mathbb{R}^{n\times m},$$
$$\tilde{C} = V^T CU \in \mathbb{R}^{n\times m},$$

dobivamo modificiranu Sylvesterovu jednadžbu

$$HY + YR^T = \tilde{C} \tag{3.1}$$

koju je bitno lakše riješiti jer su H i R blok gornje trokutaste matrice.

U drugom koraku algoritma izračunava se matrica $\tilde{C}$, a nakon toga u trećem koraku treba izračunati Y iz modificirane Sylvesterove jednadžbe. Jedan od načina rješavanja je koristeći supstituciju unatrag u kojoj redom nalazimo stupce od Y počevši od zadnjeg stupca. U nastavku ćemo detaljnije opisati tu metodu.

Označimo stupce matrica Y i $\tilde{C}$ kao

$$Y = \begin{bmatrix} y_1 & y_2 & \dots & y_m \end{bmatrix}, \qquad \tilde{C} = \begin{bmatrix} \tilde{c}_1 & \tilde{c}_2 & \dots & \tilde{c}_m \end{bmatrix}$$

i elemente matrice R kao

$$R = \begin{pmatrix} r_{ij} \end{pmatrix}, \quad i = 1, 2, \dots, n, \quad j = 1, 2, \dots, m.$$

Za početak razmotrimo slučaj kad je R prava trokutasta matrica.
Raspišimo modificiranu Sylvesterovu jednadžbu (3.1) po stupcima. Uočimo da je R^T donje trokutasta matrica pa za j-ti stupac umnoška YR^T vrijedi

$$\sum_{i=1}^{m} r_{ji} y_i = \sum_{i=j}^{m} r_{ji} y_i.$$

Dakle, za j-ti stupac modificirane Sylvesterove jednadžbe vrijedi:

$$Hy_j + \sum_{i=j}^{m} r_{ji} y_i = \tilde{c}_j.$$

Preciznije, zadnji stupac nepoznate matrice Y izračunamo iz: $Hy_m + r_{mm} y_m = \tilde{c}_m$.
Pomoću zadnjeg stupca izračunamo predzadnji: $Hy_{m-1} + r_{m-1,m-1} y_{m-1} = \tilde{c}_{m-1} - r_{m-1,m} y_m$.
Tako nastavimo sve do prvog stupca.

Dakle, općenita formula za dobivanje j-tog stupca matrice Y pod pretpostavkom da su stupci $y_{j+1}, \dots, y_m$ već izračunati je:

$$\left(H + r_{jj} I_n\right) y_j = \tilde{c}_j - \sum_{i=j+1}^{m} r_{ji} y_i.$$

Uočimo da inverz matrice $(H + r_{jj}I_n)$ postoji za svaki $j = 1, 2, \ldots, m$. Naime, matrice A i H, odnosno B i R, su slične matrice, a slične matrice imaju iste svojstvene vrijednosti. Zato slijedi da je svaka svojstvena vrijednost matrice $(H + r_{jj}I_n)$ jednaka zbroju jedne svojstvene vrijednosti matrice A i vrijednosti r_{jj} koja je svojstvena vrijednost matrice B. Znamo da je taj zbroj različit od nule iz pretpostavke o jedinstvenosti rješenja Sylvesterove jednadžbe, pa slijedi da je matrica regularna.

Vratimo se sada na blok gornje trokutastu matricu R. Iz teorema (3.1.2) znamo da je ona oblika

$$R = \begin{bmatrix} R_{[11]} & R_{[12]} & \cdots & R_{[1k]} \\ 0 & R_{[22]} & \cdots & R_{[2k]} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & R_{[kk]} \end{bmatrix}$$

pri čemu je svaki dijagonalni blok $R_{[pp]}$ ili 1×1 ili 2×2 matrica. Uočimo da je broj dijagonalnih blokova k jednak zbroju $r+c$, pri čemu je r broj realnih svojstvenih vrijednosti matrice A, a c broj kompleksno konjugiranih parova.
Razlikujemo dva slučaja pri rješavanju jednadžbe $HY + YR^T = \tilde{C}$ i to s obzirom na vrstu blok matrice $R_{[pp]}$, za svaki $p = 1, 2, \ldots, k$.

1.slučaj: $R_{[pp]}$ **je** 1×1 **matrica**

Neka je $R_{[pp]}$ 1×1 matrica (to jest, ona je jednaka nekoj svojstvenoj vrijednosti matrice B) i neka se u matrici R nalazi u j-tom retku i j-tom stupcu, to jest $r_{jj} = R_{[pp]}$. Pretpostavimo da su izračunati stupci $y_{j+1}, \ldots, y_m$. Tada j-ti stupac možemo dobiti po gornjoj formuli:

$$y_j = \left(H + r_{jj}I_n\right)^{-1}\left(\tilde{c}_j - \sum_{i=j+1}^{m} r_{ji}y_i\right).$$

Uočimo da nam u ovom slučaju ne smetaju brojevi različiti od nule ispod dijagonala matrica H i R jer za dio HY ionako množimo cijelu matricu H sa stupcem iz Y, a za YR^T nam je bitan samo j-ti stupac matrice R^T, a za njega znamo iz pretpostavke slučaja da su sve vrijednosti $r_{ji} = 0$ za $i < j$, tj. u tom stupcu matrice R su ispod dijagonale samo nule.

2.slučaj: $R_{[pp]}$ **je** 2×2 **matrica**

Neka je blok matrica $R_{[pp]}$ smještena u matrici R na stupcima i redcima $j-1$ i j. Dakle,

$$R_{pp} = \begin{bmatrix} r_{j-1,j-1} & r_{j-1,j} \\ r_{j,j-1} & r_{j,j} \end{bmatrix}.$$

Pretpostavimo ponovno da su izračunati stupci $y_{j+1}, \ldots, y_m$. Sada j-ti stupac umnoška YR^T ne možemo izračunati samo preko poznatih vrijednosti jer se element $r_{j-1,j}$ množi sa stupcem y_{j-1} koji nam još nije poznat. Zato moramo istovremeno izračunati stupce y_{j-1} i y_j, pa moramo proširiti formulu iz 1. slučaja na dva stupca:

$$H\begin{bmatrix} y_{j-1} & y_j \end{bmatrix} + \begin{bmatrix} y_{j-1} & y_j \end{bmatrix} R_{[pp]}^T = \begin{bmatrix} \tilde{c}_{j-1} & \tilde{c}_j \end{bmatrix} - \sum_{i=j+1}^{m} \begin{bmatrix} r_{j-1,i}y_i & r_{ji}y_i \end{bmatrix}. \tag{3.2}$$

Ovo se može preko Kroneckerovog produkta zapisati kao $2n \times 2n$ linearni sustav te riješiti Gaussovim eliminacijama. Dakle, treba riješiti sustav

$$\Big(I_2 \otimes H + R_{[pp]} \otimes I_n\Big) vec\Big([y_{j-1} \;\; y_j]\Big) = vec\Big([d_{j-1} \;\; d_j]\Big),$$

pri čemu je $[d_{j-1} \;\; d_j]$ jednak desnoj strani jednadžbe (3.2).

Kao i u prvom slučaju, sustav će biti rješiv uz pretpostavku da Sylvesterova jednadžba ima rješenje zato što je matrica $(I_2 \otimes H + R_{pp} \otimes I_n)$ regularna ako A i $-B$ nemaju zajedničkih svojstvenih vrijednosti.

U zadnjem koraku algoritma potrebno je iz matrice Y dobiti rješenje početne Sylvesterove jednadžbe, to jest treba izračunati $X = VYU^T$.

Napomena 3.3.1. *Da smo umjesto realne Schurove forme izabrali Schurovu formu $R = U^* B^T U$ imali bismo trokutastu matricu R, pa bi modificiranu Sylvesterovu jednadžbu bilo lakše riješiti jer ne bismo imali vrijednosti različitih od nule ispod dijagonale (dakle, imali bismo samo prvi slučaj koji je jednostavniji). Međutim, matrice U i R tada mogu biti kompleksne, pa bi zbog toga numerički račun bio skuplji (povećava se potrebni memorijski prostor i broj računskih operacija), a zbog grešaka zaokruživanja bi izračunato rješenje vjerojatno imalo netrivijalnu imaginarnu komponentu. Zbog toga je realna Schurova forma bolji izbor.*

Implementacija

U nastavku dajemo implementaciju BS algoritma u Pythonu, koristeći biblioteke numpy i scipy. U njima su već implementirane najčešće korištene operacije za rad s matricama. Nama će biti potrebne osnovne funkcije poput matričnog umnoška (*numpy.dot*) i funkcija za inicijalizaciju matrica (*numpy.zeros* i *numpy.eye*), ali i kompliciranije poput Schurove dekompozicije (funkcija *scipy.linalg.schur*) i QR dekompozicije (*numpy.linalg.qr*) te rješavača sustava linearnih jednadžbi (*numpy.linalg.solve*).

```
import numpy as np
import scipy as sc

def bs_syl(A,B,C):
    [n, n] = np.shape(A)
    [m, m] = np.shape(B)
    [H, V] = sc.linalg.schur(A, output='real')
    [R, U] = sc.linalg.schur(B.T, output='real')
    D = (V.T).dot(C.dot(U))
    Y = np.zeros((n,m))
    I = np.eye(n)

    if R[m-1, m-2]:
        Y = solve_2(H, R, Y, D, m-1)
    else:
        Y[:, m-1] = np.linalg.solve(H + R[m-1, m-1]*I, D[:, m-1])
    for k in range (m-2, -1, -1):
        if Y[:, k].any():
            k -= 1
        elif k > 0 and R[k, k-1]:
            Y = solve_2(H, R, Y, D, k)
        else:
            suma = 0
            for j in range (k+1, m, 1):
                suma += R[k, j]*(Y[:, j])
            Y[:, k] = np.linalg.solve(H + R[k, k]*I, D[:, k] - suma)
    X = V.dot(Y.dot(U.T))
    return X

#funkcija koja rješava 2n x 2n linearni sustav dobiven pomoću Kroneckerovog produkta (vidi (3.2))
def solve_2(H, R, Y, D, k):
    [n,n] = np.shape(H)
    [m,m] = np.shape(R)
    P = np.kron(np.eye(2), H) + np.kron(R[k-1:k+1, k-1:k+1], np.eye(n))
    [sum1, sum2] = [0, 0]
    for j in range(k+1, m, 1):
        sum1 += R[k-1, j]*(Y[:, j])
        sum2 += R[k, j]*(Y[:, j])
    B = [D[:, k-1] - sum1, D[:, k] - sum2]
    b = np.hstack(B).reshape(2*n, 1)
    x = np.linalg.solve(P, b)
    Y[:, k-1] = x[:n].T
    Y[:, k] = x[n:].T
    return Y
```

Hessenberg–Schurova metoda

Kada je n (dimenzija matrice A) puno veći od m (dimenzija matrice B) efikasnijom se pokazala varijanta Bartels–Stewartovog algoritma u kojoj se matrica A ne dovodi u realnu Schurovu formu, već samo u Hessenbergovu. Do Hessenbergove forme se može doći sa $5/3n^3$ operacija, dok je za Schurovu formu potrebno bar $10n^3$. Naime, algoritam svođenja matrice na Hessenbergovu formu završava u konačno mnogo koraka, dok se algoritam za svođenje na Schurovu formu zaustavlja nakon postignute zadovoljavajuće točnosti konvergencije (on zapravo generira niz matrica koje u limesu imaju traženu formu).

Na početku ove metode matrice A i B transformiramo u

$$H = V^T A V \qquad \text{i} \qquad R = U^T B^T U,$$

pri čemu je H u Hessenbergovoj formi, a R u realnoj Schurovoj formi.
Sljedeći koraci su analogni prethodno opisanom BS algoritmu zato što je realna Schurova matrica posebno i Hessenbergova matrica. Dakle, izračunamo $\tilde{C} = V^T C U$, riješimo modificiranu jednadžbu $HY + YR^T = \tilde{C}$ za Y te izračunamo $X = VYU^T$.

Ovu metodu su 1979. predložili autori Golub, Nash i Van Loan u svom članku [4] u kojem su dali i sljedeću usporedbu vremenske složenosti uz pretpostavku da realna Schurova matrica R ima $m/2$ blokova veličine $2x2$ na dijagonali (to je najgori slučaj):

	Schur	Hessenberg–Schur
1.korak	$10n^3 + 10m^3$	$5/3n^3 + 10m^3$
2.korak	$n^2m + nm^2$	$n^2m + nm^2$
3.korak	$1/2(n^2m + nm^2)$	$3n^2m + 1/2nm^2$
4.korak	$n^2m + nm^2$	$n^2m + nm^2$

Zaključili su da je Hessenberg–Schurova metoda bolja kad je n puno veći od m. Na primjer, za $n = 4m$, Hessenberg–Schurova metoda je oko 3 puta brža od Schurove.

Napomena 3.3.2. *U navedenoj implementaciji BS algoritma iz prethodnog potpoglavlja dovoljno je zamjeniti redak koji izračunava Schurovu formu matrice A retkom*

```
[H, V] = sc.linalg.hessenberg(A, calc_q=True)
```

3.4 BS algoritam za Lyapunovljevu jednadžbu

Za Lyapunovljevu jednadžbu $AX + XA^T = C$ također možemo koristiti opisani Bartels–Stewartov algoritam. Uočimo da sada trebamo samo jednu matricu transformirati u realnu Schurovu formu, pa je izračunavanje rješenja jednostavnije. Za Lyapunovljevu jednadžbu u kojoj je matrica C simetrična (a time je i rješenje simetrično), možemo dodatno pojednostaviti BS algoritam.

1. korak: Izračunamo realnu Schurovu formu matrice A, to jest nađemo blok gornje trokutastu matricu S i ortogonalnu matricu Q takve da je $S = Q^T A Q$.

2. korak: Izračunamo $D = Q^T C Q$.

3. korak: Riješimo modificiranu Lyapunovljevu jednadžbu $SY + YS^T = D$, pri čemu je $Y = Q^T X Q$.

4. korak: Izračunamo $X = QYQ^T$.

Uočimo da su matrice Y i D simetrične jer su i njima unitarno slične matrice X i C simetrične. Zbog toga možemo modificiranu Lyapunovljevu jednadžbu zapisati na sljedeći način:

$$\begin{bmatrix} S_1 & S_2 \\ 0 & S_3 \end{bmatrix} \begin{bmatrix} Y_1 & Y_2 \\ Y_2^T & Y_3 \end{bmatrix} + \begin{bmatrix} Y_1 & Y_2 \\ Y_2^T & Y_3 \end{bmatrix} \begin{bmatrix} S_1^T & 0 \\ S_2^T & S_3^T \end{bmatrix} = \begin{bmatrix} D_1 & D_2 \\ D_2^T & D_3 \end{bmatrix}, \tag{3.3}$$

pri čemu su S_3, Y_3 i D_3 ili 1×1 ili 2×2 matrice ovisno o tome je u realnoj Schurovoj formi S na mjestu S_3 blok matrica dimenzije 1×1 ili 2×2.

Kad pomnožimo blok matrice u prethodnoj jednadžbi dobivamo sljedeće jednadžbe:

$$S_3 Y_3 + Y_3 S_3^T = D_3 \tag{3.4}$$

$$S_1 Y_2 + Y_2 S_3^T = D_2 - S_2 Y_3 \tag{3.5}$$

$$S_1 Y_1 + Y_1 S_1^T = D_1 - Y_2 S_2^T - S_2 Y_2^T. \tag{3.6}$$

Ako je $S_3 \in \mathbb{R}$ (tj. 1×1 matrica) onda rješenje jednadžbe (3.4) dobivamo kao

$$Y_3 = \frac{D_3}{2S_3},$$

a ako je $S_3 = \begin{bmatrix} s_1 & s_2 \\ s_3 & s_4 \end{bmatrix}$ onda možemo, uz oznake $Y_3 = \begin{bmatrix} y_1 & y_2 \\ y_2 & y_3 \end{bmatrix}$ i $D_3 = \begin{bmatrix} d_1 & d_2 \\ d_2 & d_3 \end{bmatrix}$, prvu jednadžbu svesti na sljedeći sustav linearnih jednadžbi:

$$\begin{bmatrix} s_1 & s_2 & 0 \\ s_3 & s_1 + s_4 & s_2 \\ 0 & s_3 & s_4 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} d_1/2 \\ d_2 \\ d_3/2 \end{bmatrix}.$$

Jednadžba (3.5) je zapravo Sylvesterova jednadžba koju možemo riješiti Bartels–Stewartovim algoritom opisanim u prethodnom poglavlju. U slučaju kad je S_3 skalar možemo dodatno pojednostaviti tako da linearni sustav $(S_1 + S_3 I)Y_2 = D_2 - S_2 Y_3$ riješimo Gaussovim eliminacijama, a ne BS algoritmom.

Jednadžba (3.6) je manja Lyapunovljeva simetrična jednadžba koja ima dimenzije $(n-1)\times(n-1)$ ili $(n-2)\times(n-2)$, ovisno o veličini blok matrice S_3. Nju rješavamo na isti način kao početnu Lyapunovljevu jednadžbu. Dakle, vraćamo se na 3. korak BS algoritma, ali sada s matricama manjih dimenzija.

Opisani postupak rekurzivno ponavljamo dok blok matrica S_3 ne postane 2×2 matrica s ne-nul elementom ispod dijagonale ili 1×1 matrica.

Uočimo da jednadžbe (3.5) i (3.6) imaju rješenje jer su zadovoljeni uvjeti $\sigma(S_1)\cap\sigma(-S_3^T)=\emptyset$ i $\sigma(S_1)\cap\sigma(-S_1^T)=\emptyset$ zbog pretpostavke o jedinstvenosti rješenja.

Dakle, dobili smo rekurzivni algoritam koji zbog simetričnosti matrica Y i D zahtjeva upola manje koraka od općenitog BS algoritma.

Implementacija

```
def bs_lyap(A,C):
    if (C.T != C).any():
        bs_syl(A, A.T, C)
    [n, n] = np.shape(A)
    [S, Q] = sc.linalg.schur(A, output='real')
    D = (Q.T).dot(C.dot(Q))
    Y = np.zeros((n, n))
    [m, m] = np.shape(D)

    while m > 1:
        if S[m-1, m-2] == 0:
            S1, S2, S3, D1, D2 = create_blocks(S, D, m, 1)
            k = 1
            Y3 = (D[m-1, m-1])/(2*S[m-1, m-1])
            Y2 = np.linalg.solve(S1 + S3*np.eye(m-1), D2 - Y3*S2)

        elif m > 2:
            S1, S2, S3, D1, D2 = create_blocks(S, D, m, 2)
            k = 2
            M = np.zeros((3,3))
            M[0:2, 0:2] = S3
            M[1:3, 1:3] += S3
            d = [D[m-2, m-2]/2, D[m-2, m-1], D[m-1, m-1]/2]
            [y1, y2, y3] = np.linalg.solve(M, d)
            Y3 = [[y1, y2], [y2, y3]]
            Y2 = bs_syl(S1, S3.T, D2 - S2.dot(Y3))

        D = D1 - S2.dot(Y2.T) - Y2.dot(S2.T)
        Y[m-k:m, m-k:m] = Y3
        Y[0:m-k, m-k:m] = Y2
        Y[m-k:m, 0:m-k] = Y2.T
        m = m-k
```

```
    if m == 1:
        Y[0,0] = D/(2*S[0,0])
    X = Q.dot(Y.dot(Q.T))
    return X

#pomoćna funkcija za kreiranje blok matrica kao u (3.3) (samo za lakše praćenje koda)
def create_blocks(S, D, m, k):
    S1 = S[0:m-k, 0:m-k]
    S2 = S[0:m-k, m-k:m]
    S3 = S[m-k:m, m-k:m]
    D1 = D[0:m-k, 0:m-k]
    D2 = D[0:m-k, m-k:m]
    return S1, S2, S3, D1, D2
```

Poglavlje 4

Struktura matrica velikih dimenzija

Algoritmi opisani u prethodnom poglavlju mogu se primijeniti na matrične jednadžbe bilo kojih dimenzija. No, zbog računalnih ograničenja, oni u praksi nisu primijenjivi na matrične jednadžbe velikih dimenzija. Naime, za dovoljno veliki n nije moguće u memoriju pohraniti svih $n \times n$ elemenata matrice, transformirati takvu matricu u pogodniji oblik, niti raditi na njoj algoritme složenosti $O(n^3)$. Zato su za matrične jednadžbe velikih dimenzija razvijene drugačije metode, ovisno o strukturi pripadnih matrica. U ovom poglavlju analiziramo strukture matrica koeficijenata i matrice s desne strane linearnih matričnih jednadžbi koje se često javljaju u praksi.

4.1 Rijetko popunjene matrice

U puno važnih primjena pripadne matrice koeficijenata linearnih matričnih jednadžbi su velikih dimenzija, ali su rijetko popunjene.

Definicija 4.1.1. *Kažemo da je matrica **rijetko popunjena** ako je velika većina elemenata matrice jednaka nuli, a elementi koji nisu nula obično su pravilno raspoređeni po matrici ili čak imaju i pravilno raspoređene numeričke vrijednosti. Matrice koje nisu rijetko popunjene nazivaju se gusto popunjenima.*

Uzmimo proizvoljnu matricu $A \in \mathbb{R}^{n\times n}$ i vektor $v \in \mathbb{R}^n$. Tada je dobivanje umnoška Av prostorne i vremenske složenosti $O(n^2)$. Preciznije, potrebno je n^2 lokacija u memoriji i potrebno je $n(2n-1)$ operacija (jer n puta moramo pomnožiti n-člani redak s n-članim stupcem, a svako takvo množenje zahtijeva n množenja i n-1 zbrajanja). Primjetimo da je za rijetko popunjenu matricu A broj potrebnih operacija puno manji jer ima malo ne-nul elemenata.

U mnogo primjena se pojavljuju rijetko popunjene matrice reda n koje trebaju samo $O(n)$ mjesta u memoriji, a i umnožak s vektorom se može izvesti u linearnoj složenosti.

Prostorna složenost je također manja jer se u memoriju zapisuju samo elementi koji nisu nula. Postoji više metoda i podatkovnih struktura kojima se efikasno postiže i koristi takav zapis. U nastavku opisujemo neke od tih metoda koje se često koriste.

Spremanje rijetko popunjenih matrica

Rijetko popunjene matrice se ne spremaju kao obične matrice u obliku dvodimenzionalnih polja, već se elementi koji nisu nula spreme u neki oblik jednodimenzionalnog polja.

Dictionary of keys (DOK) je struktura u kojoj su ne-nul elementi matrice spremljeni u rječnik. Ključevi rječnika su parovi *(redak, stupac)*, a vrijednost koja je pridružena ključu (i, j) je ne-nul element matrice na i-tom retku i j-tom stupcu.

List of lists (LIL) sprema po jednu listu za svaki redak u kojoj je za svaki ne-nul element zapisan indeks stupca i vrijednost elementa.

Coordinate list (COO) je struktura koju čine trojke *(redak, stupac, element)*. Tipično je lista sortirana po indeksu retka pa po indeksu stupca.

Navedene strukture su pogodne za konstrukciju i efikasnu izmjenu matrice, no nisu pogodne za matrične operacije. Zato se općenito matrica konstruira u nekoj od navedenih struktura, a onda se konvertira u strukturu koja je pogodnija za rad s matricama. Takve strukture su opisane u nastavku.

Compressed sparse row (CSR) struktura sprema matricu u obliku tri jednodimenzionalna niza. Prvi niz čine ne-nul elementi, čitajući po redcima (od prvog do zadnjeg) s lijeva na desno. Drugi niz sadrži indekse stupaca elemenata redom kojim se pojavljuju u prvom nizu. Treći niz sadrži informacije o tome koliko svaki redak ima ne-nul elemenata. Prvi član niza je uvijek nula, a i-ti član niza je zbroj prethodnog člana i broja ne-nul elemenata u (i-1)-vom retku matrice. Ova stuktura omogućava laku podijelu matrice po redcima i brz pristup traženom retku te efikasno množenje matrice s vektorom. Također, efikasno se mogu implementirati aritmetičke operacije s matricama u CSR strukturi.

Ukoliko je potrebno raditi operacije nad stupcima matrice (kao što smo vidjeli u Bartels–Stewartovom algoritmu) pogodnije je matricu zapisati u *Compressed sparse column* (CSC) strukturi. Ona je analogna CSR strukturi samo što elemente čitamo po stupcima (od vrha do dna stupca - od prvog do zadnjeg stupca) i sažimamo indekse stupaca, a ne redaka. CSC struktura, kao i CSR struktura, podržava efikasno množenje matrice s vektorom i aritmetičke operacije s matricama.

Na primjer, matrica

$$\begin{bmatrix} 1 & 3 & 4 \\ 0 & 0 & 5 \\ 2 & 0 & 6 \end{bmatrix}$$

je pomoću CSC prikazana nizovima

$$\begin{aligned} data &= [1, 2, 3, 4, 5, 6] \\ indices &= [0, 2, 0, 0, 1, 2] \\ indptr &= [0, 2, 3, 6] \end{aligned}$$

koji redom predstavljaju niz ne-nul elemenata, niz indeksa redaka u kojima se oni nalaze i niz sažetih indeksa stupaca. U CSR strukturi ista matrica izgleda ovako:

$$\begin{aligned} data &= [1, 3, 4, 5, 2, 6] \\ indices &= [0, 1, 2, 2, 0, 2] \\ indptr &= [0, 3, 4, 6]. \end{aligned}$$

4.2 Pozitivno semidefinitne matrice niskog ranga

Čak i za linearne matrične jednadže u kojima se matrice koeficijenata mogu spremiti u memoriju s linearnom složenosti te efikasno množiti s vektorom, i dalje općenito nisu prikladni algoritmi iz Poglavlja 3 za pronalaženje rješenja. Problem je u tome što matrica rješenja može biti gusto popunjena matrica velike dimenzije koju onda ne možemo efikasno spremiti u memoriju.

Na sreću, u primjenama se često pojavljuju Sylvesterova i Lyapunovljeva matrična jednadžba kojima je matrica na desnoj strani niskog ranga što nam olakšava računanje i spremanje rješenja. Kažemo da je matrica **niskog ranga** ako je njezin rang puno manji od njezinih dimenzija.

Osim što je niskog ranga, matrica s desne strane C je često negativno (semi)definitna matrica, to jest $-C$ je simetrična pozitivno (semi)definitna matrica. Navodimo najprije definicije definitnosti.

Definicija 4.2.1. *Za simetričnu matricu A kažemo da je pozitivno semidefinitna matrica ako vrijedi $x^T Ax \geq 0$, za svaki $x \in \mathbb{R}^n$.*
Ako vrijedi $x^T Ax > 0$, za svaki $x \in \mathbb{R}^n$, $x \neq 0$, onda je A pozitivno definitna matrica.

Prisjetimo se Lyapunovljeve jednadžbe koja se pojavljuje u analizi LTI sustava. Ona je oblika $AX + XA^T = -BB^T$, pri čemu je $A \in \mathbb{R}^{n\times n}$ matrica stanja LTI sustava, a $B \in \mathbb{R}^{n\times k}$ je matrica ulaza. Općenito je broj ulaza jako mali u odnosu na veličinu sustava, pa je desna strana Lyapunovljeve jednadžbe niskog ranga.

Motivirani ovim primjerom posebnu ćemo pažnju u nastavku posvetiti sljedećoj vrsti Lyapunovljeve jednadžbe:

$$AX + XA^T = -BB^T, \tag{4.1}$$

gdje je $A \in \mathbb{R}^{n\times n}$ rijetko popunjena stabilna matrica, a matrica $B \in \mathbb{R}^{n\times k}$ punog ranga k, pri čemu je k puno manji od n.

Napomenimo da su navedene pretpostavke o Lyapunovljevoj jednadžbi tipične u literaturi o matričnim jednadžbama velikih dimenzija i da se pojavljuju i u drugim primjenama, a ne samo u analizi LTI sustava.

Uočimo da je matrica BB^T simetrična pozitivno semidefinitna matrica. To se lako pokaže iz definicije pozitivne semidefinitnosti. Dokažimo da je onda i rješenje jednadžbe (4.1) pozitivno semidefinitna matrica.

Propozicija 4.2.2. *Neka je $AX + XA^T = -BB^T$ Lyapunovljeva jednadžba s pretpostavkama iz* (4.1)*. Jednadžba tada ima jedinstveno simetrično pozitivno semidefinitno rješenje.*

Dokaz. Pokazali smo u Propoziciji 1.3.2 da za stabilnu matricu A postoji

$$P = \int_0^\infty e^{At} BB^T e^{A^T t} dt$$

i ono je rješenje Lyapunovljeve jednadžbe. Također iz stabilnosti matrice A slijedi da su sve njezine svojstvene vrijednosti različite od svojstvenih vrijednosti matrice $-A$. Znači, jednadžba ima jedinstveno rješenje P.

Dokažimo još da je ovo rješenje pozitivno semidefinitno. Neka je $x \in \mathbb{R}^n$. Tada vrijedi:

$$x^T P x = \int_0^\infty x^T e^{At} BB^T e^{A^T t} x dt = \int_0^\infty (B^T e^{A^T t} x)^T (B^T e^{A^T t} x) dt = \int_0^\infty \|B^T e^{A^T t} x\|^2 dt \geq 0.$$

□

Spomenuli smo u Poglavlju 2 da ako je LTI sustav kontrolabilan, onda je rješenje Lyapunovljeve jednadžbe pozitivno definitna matrica. Za takve jednadžbe vrijedi još dodatan rezultat.

Propozicija 4.2.3. *Neka je $AX + XA^T = C$ Lyapunovljeva jednadžba takva da je $A, C \in \mathbb{R}^{n\times n}$ i -C je simetrična pozitivno definitna matrica. Jednadžba ima jedinstveno simetrično pozitivno definitno rješenje ako i samo ako je A stabilna matrica.*

Dokaz. $\Leftarrow$ Analogno dokazu prethodne propozicije. Pritom koristimo činjenicu da je pozitivno definitna matrica regularna te da je e^{At} regularna ako je A stabilna. Ovo posljednje vrijedi zbog $det(e^M) = e^{trM}$.

$\Rightarrow$ Neka je P simetrično pozitivno definitno rješenje Lyapunovljeve jednadžbe i neka je λ svojstvena vrijednost matrice A. Tada je ona svojstvena vrijednost i matrice A^T pa je

$$A^T u = \lambda u, \text{ za neki } u \neq 0.$$

Odavde slijedi $u^*A = \bar{\lambda}u^*$.
Kad pomnožimo Lyapunovljevu jednadžbu slijeva vektorom u^* i zdesna vektorom u, dobivamo:

$$u^*APu + u^*PA^Tu = u^*Cu$$
$$\bar{\lambda}u^*Pu + u^*P\lambda u = u^*Cu$$
$$\bar{\lambda} + \lambda = -\frac{u^*(-C)u}{u^*Pu}.$$

Budući da su $-C$ i P pozitivno definitne matrice, slijedi da je $Re(\lambda) = \frac{\lambda+\bar{\lambda}}{2} < 0$. Dakle, A je stabilna matrica. □

Uočimo da je $rang(-BB^T) = rang(B) = k$ pa je desna strana jednadžbe niskog ranga jer je $k \ll n$. Promotrimo kako to utječe na rang rješenja jednadžbe. Za to nam je potreban pojam numeričkog ranga matrice. Znamo da je rang simetrične, pozitivno semidefinitne matrice jednak broju svojstvenih vrijednosti koje su strogo veće od nule. Numerički rang takvih matrica se definira kao broj svojstvenih vrijednosti matrice koji su strogo veći od neke dovoljno male vrijednosti. Ako su $\lambda_1, \lambda_2, \ldots, \lambda_n$ svojstvene vrijednosti matrice za koje vrijedi

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_\ell \gg \lambda_{\ell+1} \geq \ldots \geq \lambda_n > 0,$$

onda kažemo da je matrica numeričkog ranga ℓ, iako je ranga n. Dakle, pozitivno semidefinitna matrica je niskog numeričkog ranga ako njezine svojstvene vrijednosti brzo padaju.

Može se pokazati da je rješenje Lyapunovljeve jednadžbe (4.1) niskog numeričkog ranga. Dokaz možemo naći u [10], uz dodatni uvjet da je A dijagonalizabilna matrica. Budući da je rješenje X pozitivno semidefinitna matrica niskog numeričkog ranga, možemo ga dobro aproksimirati sa

$$X = X^T \approx ZZ^*,$$

gdje je $Z \in \mathbb{R}^{n\times\ell}$ niskog ranga ℓ. Detalji o načinu i složenosti nalaženja ove aproksimacije mogu se naći u [5].

Zaključujemo da je za navedenu jednadžbu (4.1) pri traženju rješenja X dovoljno naći matricu Z koja je puno manjih dimenzija od rješenja, što nam olakšava spremanje i ubrzava obavljanje matričnih operacija. Umjesto n^2 elemenata matrica X, sada je za reprezentaciju rješenja Lyapunovljeve jednadžbe dovoljno pohranjivati samo $n\ell$ elemenata faktora Z. Matrica X dostupna je samo implicitno, jer produkt $X = ZZ^*$ nikad nećemo efektivno izračunati.

Poglavlje 5

Algoritmi za matrice velikih dimenzija

Uz opisane algoritme kojima se pronalazi točno rješenje matrične jednadžbe, razvili su se i razni iterativni algoritmi za pronalaženje aproksimativnog rješenja. To su algoritmi u kojima ponavljamo iteracije određen broj puta te tako dobivamo niz matrica X_k koje konvergiraju ka rješenju matrične jednadžbe: $\lim_{k\to\infty} X_k = X$.

U ovom će poglavlju biti opisane dvije najčešće korištene iterativne metode za linearne matrične jednadžbe velikih dimenzija, a to su ADI iteracije i projekcijske (Krylovljeve) metode. Metode ćemo opisati i analizirati na primjeru Lyapunovljeve jednadžbe (4.1), ali one se analogno mogu primijeniti i na Sylvesterovu jednadžbu.

5.1 ADI iteracije

Alternating Direction Implicit (ADI) metoda je iterativna metoda predložena za rješavanje paraboličkih i eliptičkih parcijalnih diferencijalnih jednadžbi, no ideja ADI iteracija može se primijeniti i za rješavanje matričnih jednadžbi. U ovom potpoglavlju opisujemo ADI metodu za rješavanje Lyapunovljeve jednadžbe.

Za Lyapunovljevu jednadžbu $AX + XA^T = C$ u kojoj je C simetrična matrica, a matrica A rijetko popunjena stabilna matrica, ADI iteracije definiramo za $i = 1, 2, \ldots$ na sljedeći način:

$$(A + p_i I)X_{i-\frac{1}{2}} = C - X_{i-1}(A^T - p_i I),$$
$$(A + p_i I)X_i = C - X^*_{i-\frac{1}{2}}(A^T - p_i I),$$

pri čemu je $X_0 = 0$, a $p_i \in \mathbb{C}$ su parametri pomaka koji ćemo nazivati **ADI parametrima**. Za njih pretpostavljamo da su ili realne vrijednosti ili su kompleksno konjugirani parovi, no ni u jednom slučaju ne smiju biti svojstvene vrijednosti matrice $-A$.

Motivacija za uvođenje upravo gornjih jednadžbi dolazi iz originalne formulacije ADI za parcijalne diferencijalne jednadžbe i nećemo ih ovdje izvoditi. Uočimo samo da postavljanjem $X_{i-1/2} = X_{i-1} = X_i$ obje jednadžbe prelaze u originalnu Lyapunovljevu jednadžbu. Stoga ADI iteracije zapravo "relaksiraju" rješavanje Lyapunovljeve jednadžbe na uzastopno rješavanje sustava linearnih jednadžbi. Parametri p_i služe za ubrzavanje konvergencije.

Navedeni zapis ADI iteracija samo naglašava da se svaki korak sastoji od dva polukoraka u kojima rješavamo dva linearna sustava s istim matričnim koeficijentima. No, to se može analogno zapisati samo jednim izrazom ako iz prve jednadžbe izrazimo $X_{i-\frac{1}{2}}$ i uvrstimo taj izraz u drugu jednadžbu, raspišemo i pomnožimo slijeva s $(A + p_iI)^{-1}$. Uočimo da inverz matrice $(A + p_iI)$ postoji za svaki $i \geq 1$ jer smo pretpostavili da niti jedan ADI parametar nije svojstvena vrijednost matrice $-A$.

Da olakšamo zapis koristimo sljedeću funkciju:

$$\mathcal{F}_\gamma(z) = (z + \gamma)^{-1}(z - \bar{\gamma}).$$

Tada dobivamo

$$X_i = \mathcal{F}_{p_i}(A)X_{i-1}^*\mathcal{F}_{\bar{p}_i}(A^T) + R_i, \quad i \geq 1,$$

pri čemu je

$$\begin{aligned} R_i &= (A + p_iI)^{-1}C(I - (A^T + \bar{p}_iI)^{-1}(A^T - p_iI)) \\ &= 2Re(p_i)(A + p_iI)^{-1}C(A^T + \bar{p}_iI)^{-1} \\ &= 2Re(p_i)(A + p_iI)^{-1}C(A + p_iI)^{-*}. \end{aligned}$$

Pokažimo da vrijedi sljedeća jednakost:

$$\mathcal{F}_p(A)^* = \mathcal{F}_{\bar{p}}(A^T). \tag{5.1}$$

Lako se pokaže da matrice $(A^T + \bar{p}I)$ i $(A^T - pI)$ komutiraju, pa iz toga slijedi da i matrice $(A^T + \bar{p}I)^{-1}$ i $(A^T - pI)$ komutiraju. Zato vrijedi

$$\mathcal{F}_p(A)^* = ((A + pI)^{-1}(A - \bar{p}I))^* = (A^T - pI)(A^T + \bar{p}I)^{-1} = (A^T + \bar{p}I)^{-1}(A^T - pI) = \mathcal{F}_{\bar{p}}(A^T).$$

Uočimo da zbog simetričnosti matrice C slijedi $R_i = R_i^*$, za svaki $i \geq 1$. Budući da je $X_0 = 0$, indukcijom se lako pokaže, koristeći tvrdnju (5.1), da vrijedi $X_i = X_i^*$, za svaki $i \geq 1$.

Dakle, ADI iteracije možemo jednostavije zapisati kao

$$\begin{aligned} &X_0 = 0, \\ &X_i = \mathcal{F}_{p_i}(A)X_{i-1}\mathcal{F}_{p_i}(A)^* + 2Re(p_i)(A + p_iI)^{-1}C(A + p_iI)^{-*}, \quad i \geq 1. \end{aligned} \tag{5.2}$$

Lako se vidi da je fiksna točka ADI iteracija rješenje Lyapunovljeve jednadžbe i da su iteracije dobro definirane ako niti jedan ADI parametar nije svojstvena vrijednost matrice $-A$.

Sljedeći teorem opisuje pod kojim uvjetima na ADI parametre niz matrica X_i iz ADI iteracija konvergira u rješenje Lyapunovljeve jednadžbe X.

Teorem 5.1.1. *(Teorem o konvergenciji)*
Neka je X rješenje Lyapunovljeve jednadžbe $AX+XA^T = C$ u kojoj je C simetrična matrica. Ako je A dijagonalizabilna matrica, točnije ako postoji matrica T takva da je $T^{-1}AT$ dijagonalna matrica, i ako je X_i matrica dobivena u i-tom koraku ADI iteracija s parametrima $p_1, \ldots, p_i$ za koje vrijedi $\{p_1, \ldots, p_i\} = \{\bar{p}_1, \ldots, \bar{p}_i\}$ te ako je $X_0 = 0$, tada vrijedi:

$$\|X_i - X\|_2 \leq \mu_2^2(T) f(p_1, \ldots, p_i)^2 \|X\|_2,$$

pri čemu je

$$\mu_2(T) = \|T\|_2 \|T^{-1}\|_2$$

i

$$f(p_1, p_2, \ldots, p_i) = \max_{\lambda \in \sigma(A)} \left| \prod_{j=1}^{i} \frac{\lambda - \bar{p}_j}{\lambda + p_j} \right| = \max_{\lambda \in \sigma(A)} \left| \mathcal{F}_{p_1}(\lambda) \ldots \mathcal{F}_{p_i}(\lambda) \right|.$$

Dokaz. Iz jednadžbe (5.2) slijedi

$$X_i - X = \mathcal{F}_{p_i}(A)(X_{i-1} - X)\mathcal{F}_{p_i}(A)^* + R_i - X + \mathcal{F}_{p_i}(A)X\mathcal{F}_{p_i}(A)^*.$$

Suma zadnje tri vrijednosti je nula. Naime,

$$\begin{aligned} &R_i - X + \mathcal{F}_{p_i}(A)X\mathcal{F}_{p_i}(A)^* \\ &\quad = (A + p_i I)^{-1}(2Re(p_i)C - (A + p_i I)X(A^T + \bar{p}_i I) + (A - \bar{p}_i I)X(A^T - p_i I))(A^T + \bar{p}_i I)^{-1} \\ &\quad = (A + p_i I)^{-1}(2Re(p_i)C - 2Re(p_i)AX - 2Re(p_i)XA^T)(A^T + \bar{p}_i I)^{-1} = 0. \end{aligned}$$

Dakle, dobivamo

$$X_i - X = \mathcal{F}_{p_i}(A)(X_{i-1} - X)\mathcal{F}_{\bar{p}_i}(A^T).$$

Budući da je $X_0 = 0$, indukcijom dobivamo

$$X_i - X = -\mathcal{F}_{p_i}(A)\mathcal{F}_{p_{i-1}}(A) \ldots \mathcal{F}_{p_1}(A)X\mathcal{F}_{\bar{p}_1}(A^T) \ldots \mathcal{F}_{\bar{p}_{i-1}}(A^T)\mathcal{F}_{\bar{p}_i}(A^T). \tag{5.3}$$

Definirajmo funkciju

$$\varphi(z) = \mathcal{F}_{p_1}(z) \ldots \mathcal{F}_{p_i}(z) = \prod_{j=1}^{i} \frac{z - \bar{p}_j}{z + p_j}.$$

Pretpostavili smo da je $A = TDT^{-1}$ pri čemu je D dijagonalna matrica svojstvenih vrijednosti od A, pa vrijedi da je $\varphi(A) = T\varphi(D)T^{-1}$. Kada normiramo prethodnu jednadžbu, dobivamo

$$\|\varphi(A)\|_2 \leq \|T\|_2\|\varphi(D)\|_2\|T^{-1}\|_2 = \mu_2(T) \max_{\lambda\in\sigma(A)} |\varphi(\lambda)|.$$

Uočimo da za $\|\varphi(A^T)\|_2$ dobivamo istu nejednakost budući da je spektar matrica A i A^T isti i budući da vrijedi $\{p_1, \ldots, p_i\} = \{\bar{p}_1, \ldots, \bar{p}_i\}$. Dakle, kad izjednačimo norme matrica s lijeve i s desne strane u jednadžbi (5.3), dobivamo

$$\|X_i - X\|_2 \leq \mu_2^2(T) f(p_1, \ldots, p_i)^2 \|X\|_2.$$

□

Jako je bitno dobro odabrati ADI parametre i njihov broj kako bi izvršavanje ADI algoritma bilo uspješno i kako bi uz što manje iteracija dobili dovoljno dobro aproksimativno rješenje. Iz teorema vidimo da broj iteracija N i parametre $p_1, p_2, \ldots, p_N$ moramo odabrati tako da vrijednost funkcije $f(p_1, p_2, \ldots, p_N)$ bude što manja. Dakle, dolazimo do sljedećeg problema:

$$\min_{p_1,p_2,\ldots,p_N} \max_{x\in\sigma(A)} \left| \prod_{i=1}^{N} \frac{x - \bar{p}_i}{x + p_i} \right|, \tag{5.4}$$

pri čemu je p_i ili realna vrijednost ili postoji $j \in 1, 2, \ldots, N$ takav da vrijedi $p_j = \bar{p}_i$, za svaki $i = 1, 2, \ldots, N$.

Eksplicitno rješenje ovog problema je poznato samo za određene slučajeve, na primjer kad su sve svojstvene vrijednosti matrice A realne i negativne. No, za općenitu matricu rješenje ili nije poznato ili postoji, ali je računanje jako skupo. Zato se u praksi pronalaze takozvani suboptimalni ADI parametri $p_1, p_2, \ldots, p_N$ koje je lakše izračunati i za koje je vrijednost funkcije $f(p_1, p_2, \ldots, p_N)$ razumno mala.

Ukratko navodimo jednu od metoda za pronalaženje suboptimalnih parametara koja se pokazala zadovoljavajućom u puno primjena. Gornji problem pojednostavimo na sljedeći način:

$$\min_{p_1,p_2,\ldots,p_N\in\mathcal{E}} \max_{x\in\mathcal{E}} \left| \prod_{i=1}^{N} \frac{x - \bar{p}_i}{x + p_i} \right|,$$

gdje je $\mathcal{E}$ konačan skup aproksimacija ekstremnih svojstvenih vrijednosti matrice A (uočimo da za veliku matricu A ne možemo efikasno izračunati sve njezine svojstvene vrijednosti, nego samo njih nekoliko). Detalji i reference za problem odabira ADI parametara mogu se naći u [2, 7].

LR–ADI iteracije

Kako smo objasnili u Poglavlju 4, često je u problemima rješavanja matričnih jednadžbi velikih dimenzija matrica na desnoj strani pozitivno semidefinitna matrica niskog ranga. No, primjetimo da opisane ADI iteracije ne zahtjevaju posebnu strukturu matrice na desnoj strani jednadžbe te da je svaka iteracija X_i ponovno puna matrica velikog reda n. Zato je razvijena posebna vrsta ADI iteracija koja iskorištava niski rang matrice s desne strane Lyapunovljeve jednadžbe i zbog toga je puno manje složenosti od općenitih ADI iteracija. Nju ćemo nazivati *low rank ADI* (LR–ADI).

Ideja LR–ADI iteracija je zapisati matrice X_i kao $X_i = Z_i Z_i^*$, gdje je Z_i niskog ranga, i definirati izračunavanje matrica Z_i bez eksplicitnog formiranja matrica X_i. U literaturi se ponekad matrica Z_i naziva Cholesky faktor matrice X_i mada se pritom ne zahtijeva da Cholesky faktor bude kvadratna donje-trokutasta matrica. Zbog toga se LR–ADI u literaturi još naziva i Cholesky faktor ADI ili skraćeno CF–ADI.

Proučavamo Lyapunovljevu jednadžbu

$$AX + XA^T = -BB^T,$$

gdje je $A \in \mathbb{R}^{n\times n}$ rijetko popunjena stabilna matrica, a matrica $B \in \mathbb{R}^{n\times k}$ punog ranga k, pri čemu je k puno manji od n. Pretpostavimo još da svi ADI parametri imaju negativan realni dio.

Pokazali smo da ADI iteracije možemo zapisati u sljedećem obliku:

$$\begin{aligned} X_0 &= 0 \\ X_i &= \mathcal{F}_{p_i}(A) X_{i-1} \mathcal{F}_{p_i}(A)^* - 2Re(p_i)(A + p_i I)^{-1} BB^T (A + p_i I)^{-*}, \quad i \geq 1. \end{aligned} \tag{5.5}$$

Propozicija 5.1.2. *Za svaki $i \geq 1$ vrijedi da je X_i iz LR–ADI iteracija pozitivno semidefinitna matrica niskog ranga.*

Dokaz. Pozitivna semidefinitnost matrica X_i, $i \geq 1$ može se pokazati indukcijom koristeći sljedeće tvrdnje:

Svaki ADI parametar p_i ima negativnu realnu vrijednost.

Za svaku regularnu matricu P i pozitivno semidefinitnu matricu A vrijedi da je i matrica P^*AP pozitivno semidefinitna.

Zbroj dviju pozitivno semidefinitnih matrica je pozitivno semidefinitna matrica.

Prisjetimo se sljedećih svojstava ranga:

Za regularne matrice B i C vrijedi $rang(AB) = rang(A)$ i $rang(CA) = rang(A)$.

Za realnu matricu B vrijedi $rang(BB^T) = rang(B^T B) = rang(B) = rang(B^T)$.

Za matrice A i B jednakih dimenzija vrijedi $rang(A + B) \leq rang(A) + rang(B)$.

Odavde slijedi da je $rang(X_i) \leq rang(X_{i-1}) + rang(B)$. Zbog $rang(X_0) = 0$ slijedi $rang(X_i) \leq ik$. Dakle, X_i je niskog ranga ako je i dovoljno mali. □

Prisjetimo se da smo u Poglavlju 4 obrazložili da je i egzaktno rješenje pozitivno semidefinitna matrica niskog numeričkog ranga. Iz prethodne propozicije slijedi da se svaka matrica X_i u ADI iteracijama može prikazati kao $Z_i Z_i^*$, pri čemu je Z_i niskog ranga.

Ako je $X_{i-1} = Z_{i-1} Z_{i-1}^*$, tada iz (5.5) dobivamo sljedeće:

$$\begin{aligned} Z_0 &= 0 \\ Z_i Z_i^* &= \mathcal{F}_{p_i}(A) Z_{i-1} Z_{i-1}^* \mathcal{F}_{p_i}(A)^* - 2Re(p_i)(A + p_i I)^{-1} B B^T (A + p_i I)^{-*} \\ &= \mathcal{F}_{p_i}(A) Z_{i-1} (\mathcal{F}_{p_i}(A) Z_{i-1})^* - 2Re(p_i)(A + p_i I)^{-1} B ((A + p_i I)^{-1} B)^* \\ &= U_i U_i^* + V_i V_i^* \end{aligned}$$

gdje su matrice U_i i V_i definirane na sljedeći način:

$$\begin{aligned} U_i &= \mathcal{F}_{p_i}(A) Z_{i-1}, \quad i \geq 2, \quad U_1 = 0, \\ V_i &= \sqrt{-2Re(p_i)}(A + p_i I)^{-1} B, \quad i \geq 1. \end{aligned}$$

Slijedi da se $Z_i \in \mathbb{C}^{n \times ik}$ može dobiti spajanjem matrica $U_i \in \mathbb{C}^{n \times k(i-1)}$ i $V_i \in \mathbb{C}^{n \times k}$. To jest, matricu Z_i možemo zapisati kao

$$Z_i = \begin{bmatrix} V_i & U_i \end{bmatrix}.$$

Dobivamo sljedeće iteracije:

$$\begin{aligned} Z_1 &= \sqrt{-2Re(p_1)}(A + p_1 I)^{-1} B, \\ Z_i &= \begin{bmatrix} \sqrt{-2Re(p_i)}(A + p_i I)^{-1} B & \mathcal{F}_{p_i}(A) Z_{i-1} \end{bmatrix}, \quad i = 2, 3, \ldots, N. \end{aligned}$$

Ove iteracije su manje računalno zahtjevne od općenitih ADI iteracija. Nema potrebe za spremanjem i računanjem matrice X_i u svakom koraku, već se sve operacije rade na matrici Z_i koja je puno manjih dimenzija od matrice X_i. Uočimo još da se u svakom koraku matrica $Z_{i-1} \in \mathbb{C}^{n \times (i-1)k}$ množi slijeva matricom $\mathcal{F}_{p_i}(A)$ pa broj stupaca koji se treba pomnožiti u svakoj iteraciji raste za broj k. Ovo se može izbjeći korištenjem tzv. Li–Whiteovog trika [7], kojeg ovdje nećemo opisati. Korištenjem tog trika u svakom koraku treba primijeniti $\mathcal{F}_{p_i}(A)$ na točno k stupaca.

Svaki korak ADI iteracija zahtjeva rješavanje $2n$ linearnih sustava, 2 množenja matrica i spremanje jedne matrice veličine $n \times n$. S druge strane, u svakom koraku LR–ADI iteracija

treba riješiti k linearnih sustava i spremiti matrice veličine najviše $n \times kN$, pri čemu je k rang matrice B (odnosno broj stupaca matrice B), a N je broj iteracija.

Ako je A rijetko popunjena matrica za koju se množenje s vektorom i rješavanje sustava $(A + pI)x = v$ može izvesti s linearnom složenosti, onda je izvršavanje ADI iteracija složenosti $O(Nn^2)$ dok je složenost LR–ADI iteracija $O(Nkn)$. Dakle, možemo zaključiti da je za Lyapunovljeve jednadžbe u kojima je desna strana niskog ranga ($k \ll n$) LR–ADI algoritam puno brži od običnih ADI iteracija.

ADI parametre za LR–ADI možemo izabrati na isti način kao i za obične ADI iteracije, rješavanjem minimax problema (5.4) ili pronalaženjem suboptimalnih parametara. Dodatno je za određivanje broja iteracija predloženo da kriterij zaustavljanja može biti $\|V_k\|_2/\|Z_k\|_2 < \varepsilon$, za dovoljno mali ε (vidi [2]).

Koristeći LR–ADI iteracije može se dogoditi da je matrica Z_i kompleksna matrica iako smo odabrali kompleksne ADI parametre. Napomenili smo već da su operacije s kompleksnim matricama računski skuplje od realnog računanja. No, to se može izbjeći posebnom implementacijom algoritma koja kombinira dva uzastopna koraka iteracija u kojima su ADI parametri kompleksno konjugirani par (detalji u [7]).

Implementacija

Slijedi implementacija LR–ADI algoritma u Pythonu. Osim već viđenih funkcija iz scipy i numpy biblioteka, sada još koristimo paket scipy.sparse koji matrice sprema u nekom od formata za rijetko popunjene matrice (npr. funkcija *scipy.sparse.eye*). Operacije s takvim formatom matrica su implementirane u paketu scipy.sparse.linalg. Ovdje ćemo koristiti funkciju *scipy.sparse.linalg.spsolve* koja rješava sustav linearnih jednadžbi u kojem je matrica sustava rijetko popunjena matrica.

```
import numpy as np
import scipy as sc
import scipy.sparse.linalg as sp

def adi(A, B, N):
    #N označava broj koraka ADI algoritma
    p = odaberi_shiftove(A, N)
    [n, n] = np.shape(A)
    I = scipy.sparse.eye(n, format='csc')
    Z = np.sqrt(-2*p[0])*(sp.spsolve(A+p[0]*I, B))

    for i in range(1, N, 1):
        U = sp.spsolve(A+p[i]*I, (A-p[i]*I).dot(Z))
        V = np.sqrt(-2*p[i])*(sp.spsolve(A+p[i]*I, B))
        Z = np.concatenate((V, U), axis=1)
    return Z
```

```
#ovo je jedan od načina odabira ADI parametara
def odaberi_shiftove(A, N):
    [n, n] = np.shape(A)
    v = np.random.uniform(0, 1, (n, 1))
    v1 = v/(sc.linalg.norm(v))
    #Arnoldijev algoritam će biti opisan i implementiran u sljedećem poglavlju
    V, H = Arnoldi(A, v1, N)
    p_adi = sc.linalg.eigvals((V.T).dot(A.dot(V)))
    return p_adi
```

Funkcija *scipy.linalg.norm* računa normu, a funkcija *scipy.linalg.eigvals* svojstvene vrijednosti matrice.

5.2 Projekcijske metode

Ideja projekcijskih metoda je odabrati potprostor $\mathcal{V}_m$ od $\mathbb{R}^n$ dimenzije $m \ll n$, izračunati rješenje X_m reducirane Lyapunovljeve jednadžbe koja se dobije projekcijom matrice koeficijenata na potprostor $\mathcal{V}_m$ te iz rješenja X_m izraziti aproksimativno rješenje originalne Lyapunovljeve jednadžbe.

I dalje proučavamo Lyapunovljevu jednadžbu $AX + XA^T = -BB^T$, gdje je $A \in \mathbb{R}^{n\times n}$ rijetko popunjena stabilna matrica, a matrica $B \in \mathbb{R}^{n\times k}$ punog ranga k, $k \ll n$.

Pretpostavimo da je rješenje X Lyapunovljeve jednadžbe takvo da su svi stupci (pa onda zbog simetričnosti i redci) iz potprostora $\mathcal{V}_m$. Tada vrijedi $X = V_m X_m V_m^T$, za neku matricu $X_m \in \mathbb{R}^{m\times m}$. Uvrstimo ovaj izraz u Lyapunovljevu jednadžbu:

$$AV_m X_m {V_m}^T + V_m X_m {V_m}^T A = -BB^T.$$

Kad pomnožimo prethodnu jednadžbu slijeva s ${V_m}^T$, a zdesna s V_m dobivamo reduciranu Lyapunovljevu jednadžbu:

$$({V_m}^T A V_m)X_m + X_m({V_m}^T A^T V_m) = -{V_m}^T BB^T V_m, \tag{5.6}$$

pri čemu je matrica $V_m \in \mathbb{R}^{n\times m}$ punog ranga $m \ll n$ s ortonormiranim stupcima koji razapinju potprostor $\mathcal{V}_m$.

Zbog $m \ll n$, matrica koeficijenata reducirane Lyapunovljeve jednadžbe (5.6) i matrica na njenoj desnoj strani su malih dimenzija. Zato nju možemo riješiti nekim od algoritama za matrice manjih dimenzija iz Poglavlja 3.

Budući da stupci od X ne moraju nužno biti iz $\mathcal{V}_m$ kako smo pretpostavili, općenito vrijedi $X \approx V_m X_m {V_m}^T$ umjesto egzaktne jednakosti. Uočimo još da je ovo aproksimativno rješenje niskog ranga.

Postavlja se pitanje kakav treba biti potprostor $\mathcal{V}_m$. Takav potprostor mora biti jednostavan za generirati i svojom strukturom mora osigurati dovoljno kvalitetnu informaciju o

objektima koje želimo izračunati. Jedna klasa takvih potprostora su Krylovljevi potprostori, koji su istovremeno i teorijski alat i osnova za mnoge numeričke algoritme.

Definicija 5.2.1. *Neka je $A \in \mathbb{R}^{n\times n}$ i $b \in \mathbb{R}^n$. Krylovljeva matrica reda m je definirana s*

$$K_m = K_m(A, b) = \begin{bmatrix} b & Ab & \dots & A^{m-1}b \end{bmatrix}.$$

Krylovljev potprostor reda m generiran s matricom A i vektorom b dan je s

$$\mathcal{K}_m = \mathcal{K}_m(A, b) = span\Big\{b,\ Ab,\ \dots,\ A^{m-1}b\Big\}.$$

Krylovljeve matrice i potprostori pojavljuju se u modernim iterativnim metodama za rješavanje velikih linearnih sustava ili pronalaženje svojstvenih vrijednosti velikih matrica. Umjesto množenja dviju matrica, potreban račun se želi svesti na umnožak matrice i vektora. Počevši s vektorom b, najprije se izračuna Ab, pa se taj vektor množi s A slijeva da se dobije A^2b i tako dalje. Algoritmi koji rade na ovaj način nazivaju se metodama Krylovljevih potprostora i trenutno su među najuspješnijim metodama u numeričkoj linearnoj algebri.

U nastavku opisujemo dvije metode Krylovljevih potprostora za rješavanje matričnih jednadžbi, no u literaturi se može naći još sličnih metoda. Njihova glavna ideja je izgradnja ortonormirane baze za Krylovljev potprostor i projekcija matrice koeficijenata na potprostor. No, najprije opisujemo Arnoldijev algoritam koji će se koristiti u metodama.

Arnoldijev algoritam

Standardni način za pronalaženje ortonormirane baze Krylovljevog potprostora je koristeći Arnoldijev algoritam. To je iterativni algoritam koji, za općeniti slučaj, u m-tom koraku generira bazu za $\mathcal{K}_m(A, b)$. Osim Arnoldijevog algoritma za pronalaženje baze Krylovljevog potprostora često se koristi i Lanczosov algoritam.

Opisujemo jednu od varijanti Arnoldijevog algoritma.

Za matricu $A \in \mathbb{R}^{n\times n}$ i početni vektor $b \in \mathbb{R}^n$ definiramo

$$v_1 = \frac{b}{\|b\|_2}.$$

U svakom koraku $j = 1, 2, \dots, m$ računamo:

$$\begin{aligned} h_{ij} &= \langle Av_j, v_i\rangle \quad i = 1, 2, \dots, j, \\ w_j &= Av_j - \sum_{i=1}^{j} h_{ij}v_i, \\ h_{j+1,j} &= \|w_j\|_2, \\ v_{j+1} &= \frac{w_j}{h_{j+1,j}}, \end{aligned}$$

s time da izračunavanje vektora v_{j+1} radimo samo ako je $h_{j+1,j} \neq 0$. U suprotnom stajemo s algoritmom jer ne možemo izračunati vektor v_{j+1}.

Ako je algoritam stao u m-tom koraku, dobivamo matricu $V_m = \begin{bmatrix} v_1 & v_2 & \dots & v_m \end{bmatrix} \in \mathbb{R}^{n\times m}$ i Hessenbergovu matricu $H_{m+1,m} \in \mathbb{R}^{(m+1)\times m}$ s elementima h_{ij}.

Propozicija 5.2.2. *Neka su $v_1, v_2, \dots, v_m$ vektori dobiveni Arnoldijevim algoritmom nakon izvršavanja m koraka. Tada oni čine ortonormiranu bazu za Krylovljev potprostor $\mathcal{K}_m(A, v_1)$.*

Dokaz. Uočimo da je konstrukcija vektora $v_1, v_2, \dots, v_m$ analogna konstruiranju ortonormiranih vektora u Gram-Schmidtovom postupku ortogonalizacije. Zbog toga se ortonormiranost vektora $v_1, v_2, \dots, v_m$ može dokazati kao u Gram–Schmidtovom postupku ortogonalizacije.

Da vektori $v_1, v_2, \dots, v_m$ razapinju Krylovljev potprostor $\mathcal{K}_m(A, v_1)$ slijedi iz tvrdnje da se svaki vektor v_j može prikazati u obliku $q_{j-1}(A)v_1$, pri čemu je q_{j-1} polinom stupnja j-1. Ova tvrdnja se lako može pokazati indukcijom. □

Propozicija 5.2.3. *Neka su matrica $V_m = \begin{bmatrix} v_1 & v_2 & \dots & v_m \end{bmatrix} \in \mathbb{R}^{n\times m}$ i Hessenbergova matrica $H_{m+1,m} \in \mathbb{R}^{(m+1)\times m}$ s elementima h_{ij} dobiveni Arnoldijevim algoritmom. Označimo s $H_m \in \mathbb{R}^{m\times m}$ Hessenbergovu matricu koju dobijemo kad iz matrice $H_{m+1,m}$ izbacimo zadnji redak. Tada vrijedi*

$$V_m{}^T A V_m = H_m.$$

Dokaz. Iz algoritma slijedi

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i, \quad j = 1, 2, \dots, m.$$

Matrično se to može zapisati kao

$$AV_m = V_{m+1} H_{m+1,m}.$$

Ovaj se zapis može, zbog činjenice da je u zadnjem retku matrice $H_{m+1,m}$ jedino element $h_{m+1,m}$ različit od nule, preformulirati u

$$AV_m = V_m H_m + w_m e_m{}^T,$$

pri čemu je kao u algoritmu $w_m = v_{m+1} h_{m+1,m}$. Kad ovaj izraz pomnožimo slijeva s $V_m{}^T$, zbog ortonormiranosti skupa $\{v_1, v_2, \dots, v_m\}$ dobivamo

$$V_m{}^T A V_m = H_m.$$

□

Blok Krylovljeva metoda

U ovom potpoglavlju opisujemo prvu metodu predloženu u literaturi za konstrukciju matrice V_m čiji stupci razapinju Krylovljev potprostor. Temelji se na Ritz-Galerkinovom uvjetu da rezidualna greška aproksimativnog rješenja bude bude ortogonalna na Krylovljev potprostor.

Preciznije, za Lyapunovljevu jednadžbu $AX + XA^T = -BB^T$, gdje je $A \in \mathbb{R}^{n\times n}$ rijetko popunjena stabilna matrica, a matrica $B \in \mathbb{R}^{n\times k}$ punog ranga k, $k \ll n$, konstruiramo blok Krylovljev potprostor definiran kao

$$\mathcal{K}_m(A, B) = span\Big\{B, AB, A^2B, \ldots, A^{m-1}B\Big\},$$

tako da za aproksimativno rješenje $V_m X_m {V_m}^T$ i za rezidualnu grešku $R(V_m X_m {V_m}^T) = A(V_m X_m {V_m}^T) + (V_m X_m {V_m}^T)A^T + BB^T$ vrijedi

$$R(V_m X_m {V_m}^T) \perp \mathcal{K}_m(A, B).$$

Za konstrukciju ortonormirane baze za Krylovljev potprostor $\mathcal{K}_m(A, B)$ koristimo blok verziju Arnoldijevog algoritma, zato što matricu A množimo s grupom vektora (to jest, s matricom dimenzija $n \times k$), a ne s jednim vektorom. Blok Arnoldijev algoritam za matrice A i B koristi QR faktorizaciju matrica. Prisjetimo se, pravokutna matrica $B \in \mathbb{R}^{n\times k}$ punog ranga k se može faktorizirati kao $Q_1 R_1$, pri čemu je $Q_1 \in \mathbb{R}^{n\times k}$ ortogonalna matrica, a $R_1 \in \mathbb{R}^{k\times k}$ gornje trokutasta matrica.

U nastavku opisujemo blok Arnoldijev algoritam:
Najprije izračunamo ortogonalnu matricu $Q_1 \in \mathbb{R}^{n\times k}$ iz QR faktorizacije matrice B.
U svakom koraku $j = 1, 2, \ldots, m$ računamo:

$$\tilde{H}_{ij} = {Q_i}^T A Q_j \quad i = 1, 2, \ldots, j,$$

$$W_j = AQ_j - \sum_{i=1}^{j} Q_i \tilde{H}_{ij},$$

izračunamo QR faktorizaciju od $W_j = Q_{j+1}\tilde{H}_{j+1,j}$ (ako je W_j punog ranga).

Algoritam staje u m-tom koraku ako je neka od matrica $\tilde{H}_{im}, i \in \{1, 2, \ldots, m\}$ jednaka nul matrici.

Tada dobivamo niz matrica $Q_1, Q_2, \ldots, Q_m$ takve da stupci matrice $V_m = \begin{bmatrix} Q_1 & Q_2 & \ldots & Q_m \end{bmatrix}$ čine ortonormiranu bazu za Krylovljev potprostor $\mathcal{K}_m(A, B)$. Također, dobivamo matricu $H_{m+1,m} \in \mathbb{R}^{(m+1)k\times mk}$ koju sačinjavaju blok matrice $\tilde{H}_{ij} \in \mathbb{R}^{k\times k}$.

Preciznije,

$$H_{m+1,m} = \begin{bmatrix} \tilde{H}_{11} & \tilde{H}_{12} & \dots & \tilde{H}_{1m} \\ \tilde{H}_{21} & \tilde{H}_{22} & \dots & \tilde{H}_{2m} \\ 0 & \tilde{H}_{32} & \dots & \tilde{H}_{3m} \\ & & \ddots & \\ 0 & 0 & \dots & \tilde{H}_{m+1,m} \end{bmatrix},$$

pri čemu je za svaki $j = 1, 2, \dots, m$ matrica $\tilde{H}_{j+1,j}$ gornje trokutasta jer je dobivena QR faktorizacijom. Uočimo da matrica $H_{m+1,m}$ ima ispod glavne dijagonale još k poddijagonala čiji su elementi različiti od nule (nema pravu Hessenbergovu strukturu), pa ćemo ju nazivati blok Hessenbergovom matricom. Analogno dokazu Propozicije 5.2.3 može se pokazati da vrijedi $AV_m = V_{m+1}H_{m+1,m}$.

Napomenimo da u slučaju da neka od matrica W_j nije punog ranga svejedno možemo nastaviti s algoritmom, ali tada koristimo QR faktorizaciju s pivotiranjem pa u blok Hessenbergovoj matrici $H_{m+1,m}$ dobivamo blok matrice različitih dimenzija.

Nakon što izračunamo matrice V_m i $H_{m+1,m}$ blok Arnoldijevim algoritmom trebamo projicirati matricu koeficijenata Lyapunovljeve jednadžbe na Krylovljev potprostor $\mathcal{K}_m(A, B)$. Kao u Propoziciji 5.2.3 možemo pokazati da i za blok Hessenbergovu matricu H_m dobivenu micanjem zadnjeg retka matrice $H_{m+1,m}$ vrijedi

$$V_m{}^T A V_m = H_m.$$

Dakle, reduciranu Lyapunovljevu jednadžbu (5.6) možemo zapisati kao

$$H_m X_m + X_m H_m{}^T = -V_m{}^T B B^T V_m.$$

Problem nastaje ako reducirana Lyapunovljeva jednadžba nema jedinstveno rješenje. Naime, da bi reducirana jednadžba imala jedinstveno rješenje nije dovoljno da početna jednadžba ima jedinstveno rješenje. Međutim, može se pokazati da ako je matrica koeficijenata A početne Lyapunovljeve jednadžbe takva da je matrica $-(A+A^T)$ pozitivno definitna matrica, tada je matrica H_m stabilna pa dobivena reducirana Lyapunovljeva jednadžba ima jedinstveno rješenje.

Uočimo još da je Ritz–Galerkinov uvjet zadovoljen ako reducirana jednadžba ima jedinstveno rješenje X_m jer za projekciju rezidualne greške aproksimativnog rješenja $X \approx V_m X_m V_m{}^T$ na Krylovljev potprostor $\mathcal{K}_m(A, B)$ vrijedi:

$$V_m{}^T R(X) V_m = V_m{}^T A V_m X_m + X_m V_m{}^T A^T V_m + V_m{}^T B B^T V_m = 0.$$

Implementacija

U implementaciji blok Arnoldijevog algoritma koristimo modificirani Gram-Schmidtov postupak ortogonalizacije jer se to pokazalo učinkovitijim od opisanog načina. Detalje o tom postupku i o različitim verzijama Arnoldijevog algoritma može se naći u [8].

```
def Arnoldi(A, B, m):
    [n, n] = np.shape(A)
    [n, k] = np.shape(B)
    Vm = np.zeros((n, k*m))
    H = np.zeros(((m+1)*k, m*k))
    Qnext, R = np.linalg.qr(B)
    for j in range(1, m+1, 1):
        Vm[:, (j-1)*k:j*k] = Qnext
        Wj = A.dot(Qnext)
        for i in range(1, j+1, 1):
            Qi = Vm[:, (i-1)*k:i*k]
            Hij = (Qi.T).dot(Wj)
            Wj = Wj - Qi.dot(Hij)
            H[(i-1)*k:i*k, (j-1)*k:j*k] = Hij
        Qnext, Hnext = np.linalg.qr(Wj)
        H[j*k:(j+1)*k, (j-1)*k:j*k] = Hnext
    return Vm, H[0:m*k, :]

def Krylov(A, B, m):
    Vm, Hm = Arnoldi(A, B, m)
    D = (Vm.T).dot(B)
    C = -D.dot(D.T)
    Xm = bs_lyap(Hm, C)
    return Vm, Xm
```

Racionalan Krylovljev potprostor

Na kraju opisujemo drugu projekcijsku metodu koja se pokazala efikasnom u numeričkom računanju, a bitna je i zbog svoje veze s ADI iteracijama. Ovom metodom konstruiramo ortogonalnu bazu za racionalni Krylovljev potprostor čiju definiciju dajemo u nastavku.

Definicija 5.2.4. *Neka je $s = \{s_1, s_2, \ldots, s_m\}$ niz parametara koje ćemo nazivati pomacima. Tada za matrice A i B te za niz parametara s, racionalan Krylovljev potprostor definiramo kao*

$$\mathcal{K}_m(A, B, s) = span\Big\{(A + s_1)^{-1}B, (A + s_2)^{-1}(A + s_1)^{-1}B, \ldots, \prod_{i \leq m} (A + s_i)^{-1}B\Big\}.$$

Ortonormiranu bazu za racionalan Krylovljev potprostor možemo generirati blok Arnoldijevim algoritmom kao u opisanoj blok Krylovljevoj metodi. Na analogan način dobivamo aproksimativno rješenje Lyapunovljeve jednadžbe, samo što je sada rezidualna greška aproksimativnog rješenja ortogonalna na racionalni Krylovljev potprostor.

U nastavku iskazujemo teorem koji opisuje vezu racionalnih Krylovljevih potprostora s ADI iteracijama. Dokaz teorema i detalje se može naći u [7].

Teorem 5.2.5. *Neka je Z_m matrica dobivena u m-tom koraku LR-ADI iteracija za Lyapunovljevu jednadžbu $AX + XA^T = -BB^T$ i neka su $\{p_q, p_2, \ldots, p_m\}$ pripadni ADI parametri. Tada stupci matrice Z_m razapinju racionalan Krylovljev potprostor $\mathcal{K}_m(A, B, p)$.*

Pokazalo se da metodom racionalnih Krylovljevih potprostora, uz dobar odabir parametara, postižemo bržu konvergenciju nego metodom blok Krylovljevih potprostora. No, općenito je problem odabira dobrih parametara za metodu racionalnih Krylovljevih potprostora težak kao i u ADI iteracijama.

5.3 Primjer

Na kraju dajemo numerički primjer za usporedbu metoda iz ovog poglavlja. Želimo riješiti Lyapunovljevu jednadžbu $AX+XA^T = -BB^T$ algoritmima koje smo implementirali za ADI iteracije i blok Krylovljevu metodu. Cilj nam je vidjeti koliko iteracija treba svaki od njih da bismo dobili rješenje zadovoljavajuće točnosti.

Za rijetko popunjenu stabilnu matricu A uzet ćemo tridijagonalnu matricu koja na glavnoj dijagonali ima vrijednost -4, na prvoj naddijagonali vrijednost -2.5, a na prvoj poddijagonali vrijednost -0.5. Matricu B biramo tako da njezini elementi budu uniformno distribuirani na $[0, 1]$. Neka te matrice budu manjih dimenzija tako da možemo izračunati egzaktno rješenje Bartels–Stewartovim algoritmom. Na primjer, uzmimo da je matrica A reda $n = 400$, a matrica B dimenzije 400×2.

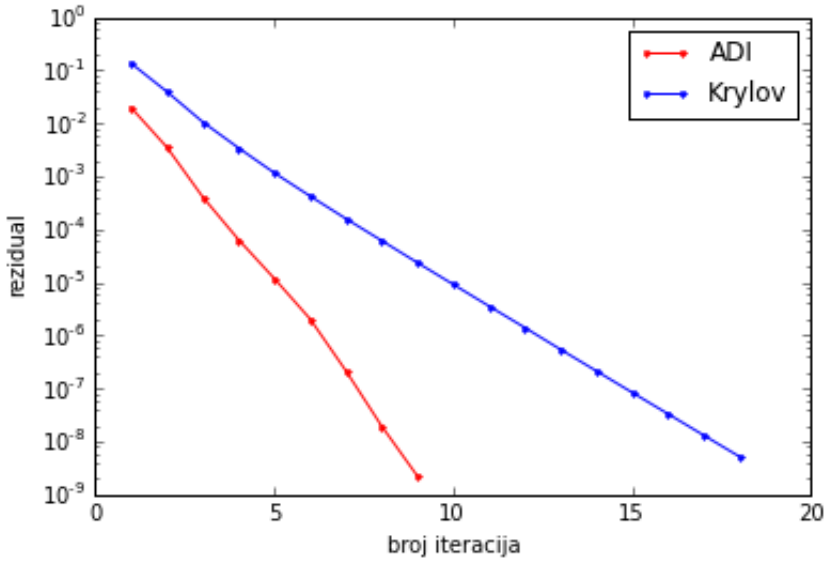
Za ADI parametre uzmimo svojstvene vrijednosti matrice ${V_m}^T A V_m$ gdje je matrica V_m dobivena nakon 20 iteracija Arnoldijevog algoritma (implementirano u potpoglavlju 5.1).

Oba algoritma modificiramo tako da se zaustave kada rezidualna greška $\|AX_m + X_m A^T + BB^T\|$ postane dovoljno mala, recimo manja od $10^{-8}\|B\|^2$. Kako bismo mogli usporediti ova dva algoritma, ispišimo korak u kojem su oni stali i ispišimo dimenzije matrica kojima aproksimiramo rješenje. U slučaju ADI iteracija je to faktor Z, a u slučaju Krylovljeve metode matrica V_m. Također, da bi vidjeli koliko se aproksimativno rješenje razlikuje od egzaktnog, ispišimo $\|X - X_{approx}\|_2$, pri čemu je X rješenje dobiveno BS algoritmom.

Za neku matricu B tada dobivamo sljedeći rezultat:

```
ADI iteracije:
  Broj iteracija:  9
  Dimenzija matrice Z:  (400, 18)
  ||X-X_adi|| =  7.69448051587e-08
Blok Krylovljeva metoda:
  Broj iteracija:  18
  Dimenzija matrice Vm:  (400, 36)
  ||X-X_kry|| =  2.35042619092e-07
```

Za isti primjer dajemo grafički prikaz opadanja rezidualne greške (precizije, rezidual podijeljen s kvadratom norme matrice B) da si možemo predočiti koliko brzo iteracije konvergiraju ka rješenju.



Dakle, vidimo da, iako je dimenzija jednadžbe 400, dovoljno je napraviti aproksimaciju iz potprostora male dimenzije da se dobije izuzetno točno rješenje.

Bibliografija

[1] A. C. Antoulas, *Approximation of large-scale dynamical systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.

[2] D. A. Bini, B. Iannazzo i B. Meini, *Numerical solution of algebraic Riccati equations*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2012.

[3] B. N. Datta, *Numerical methods for linear control systems*, Academic Press, San Diego, 2004.

[4] G. H. Golub, S. Nash i C. Van Loan, *A Hessenberg-Schur method for the problem* $AX + XB = C$, IEEE Trans. Automat. Control **24** (1979), 909–913.

[5] M. Harbrecht, H.and Peters i R. Schneider, *On the low-rank approximation by the pivoted Cholesky decomposition*, Appl. Numer. Math. **62** (2012), 428–440.

[6] A. Jameson, *Solution of the equation* $AX + XB = C$ *by inversion of an* $M \times M$ *or* $N \times N$ *matrix*, SIAM J. Appl. Math. **16** (1968), 1020–1023.

[7] J. R. Li i J. White, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl. **24** (2002), 260–280.

[8] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[9] V. Simoncini, *Computational methods for linear matrix equations*, (Survey article), ožujak 2013.

[10] D. Sorensen i Y. Zhou, *Bounds on eigenvalue decay rates and sensitivity of solutions to Lyapunov equations*, Teh. izv. TR02-07, Rice University, 2002.

Sažetak

U ovom smo radu dali pregled postojećih rezultata u području numeričkog rješavanja linearnih matričnih jednadžbi na primjeru Sylvesterove i Lyapunovljeve jednadžbe. Naveli smo zašto su od interesa u primijenjenoj matematici, posebno u analizi stabilnosti dinamičkih sustava.

Razmotrili smo često korišten Bartels–Stewartov algoritam koji je pogodan za rješavanje jednadžbi malih dimenzija. Za matrične jednadžbe velikih dimenzija razmotrili smo iterativne načine rješavanja: metodu ADI iteracija i projekcijsku Krylovljevu metodu. Pritom smo uzeli u obzir strukture matrica koje se često pojavljuju u primjenama. Implementaciju ovih algoritama smo napravili u Pythonu.

Summary

In this work we reviewed main computational methods for solving linear matrix equations, particularly Sylvester and Lyapunov equations. We explained their role in applied mathematics, especially in stability analysis of linear dynamic systems.

We described Bartels–Stewart algorithm which is frequently used for solving matrix equations of small dimensions. For large-scale matrix equations, we considered special structure of matrices which are often encountered in applications and we described iterative methods for finding solution: ADI method and projection (Krylov) method. We implemented listed algorithms using Python.

Životopis

Rođena sam 22. siječnja 1992. godine u Puli. Nakon završene osnovne škole u Svetom Petru u Šumi, upisala sam klasičnu gimnaziju u Pazinu. Tijekom srednje škole sudjelovala sam na nekoliko državnih natjecanja iz područja matematike, astronomije i informatike te sam tri godine bila polaznica Višnjanske škole astronomije.
2010. godine sam upisala preddiplomski studij matematike na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta u Zagrebu. Preddiplomski studij sam završila 2013. godine. Te godine sam upisala diplomski sveučilišni studij Računarstvo i matematika na istom fakultetu.