Hijerarhijske matrice

Lujo, Marija

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Marija Lujo

HIJERARHIJSKE MATRICE

Diplomski rad

Voditelj rada:
izv. prof. dr. sc. Zvonimir Bujanović

Zagreb, rujan 2022.

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

Uvod

U raznim primjenama često se pojavljuju matrice jako velikih dimenzija. Eksplicitna pohrana takvih matrica u računalu prostorne složenosti $O(n^2)$ u većini slučajeva je neefikasna, a ponekad i nemoguća. Još se veći problemi javljaju ako nad takvim matricama vršimo matrične operacije, primjerice, vremenska složenost množenja dviju matrica dimenzije $n \times n$ je $O(n^3)$. Postoje neka poboljšanja poput „podijeli pa vladaj" Strassenovog algoritma[1], no ni taj algoritam nema složenost manju od $O(n^2)$, što svakako nije dovoljno dobro za velike matrice.

Srećom, neke matrice koje se javljaju u primjenama su rijetko popunjene matrice ili se mogu aproksimirati rijetko popunjenom matricom. Njih dobivamo, npr. diskretizacijom parcijalnih diferencijalnih jednadžbi metodom konačnih elemenata, a pojavljuju se i kao matrica susjedstva u teoriji grafova kod analize društvenih mreža. Ovo su samo neki od primjera gdje se pojavljuju rijetko popunjene matrice. Takve matrice mogu se pohraniti u prostornoj složenosti $O(n)$, a neke matrične operacije nad rijetko popunjenim matricama moguće je efikasnije izvršavati. Međutim, svojstvo rijetkosti u većini slučajeva nije očuvano prilikom standardnih matričnih algoritama, na primjer LU faktorizacije ili inverza matrice.

Tema ovog diplomskog rada su hijerarhijski formati matrica. To je jedan primjer podatkovno rijetke strukture matrica, tj. matrice koja se može pohraniti u prostornoj složenosti manjoj od $O(n^2)$. Takve matrice imaju rekurzivnu blok-strukturu, mogu se efikasno pohraniti, a pri matričnim operacijama im je očuvana struktura. No, ne mogu se sve matrice efikasno pohraniti u hijerarhijskom formatu. Važan je uvjet da se određeni blokovi matrica koje biramo po nekom *uvjetu prihvatljivosti* mogu pohraniti u reprezentaciji niskog ranga, točnije u obliku umnoška dva faktora UV^T, gdje U i V imaju jako mali broj stupaca. Takve matrice prirodno se javljaju u diskretizacijama diferencijalnih i integralnih jednadžbi s obzirom na to da tada znamo u kojim blokovima možemo očekivati dobru reprezentaciju niskog ranga.

Hijerarhijske ili $\mathcal{H}$-matrice prvi je put uveo Wolfgang Hackbusch[2] krajem 90-ih go-

[1] Vidi [2].

[2] Wolfgang Hackbusch (1948.), njemački matematičar, pionir u istraživanju hijerarhijskih struktura matrica.

dina prošlog stoljeća. Hijerarhijske matrice sa oslabljenim uvjetom prihvatljivosti (svaki vandijagonalni blok se može pohraniti u reprezentaciji niskog ranga) nazivaju se HODLR matrice i njima ćemo se najviše baviti u ovom radu. Još neki primjeri hijerarhijske strukture matrica su HSS i $\mathcal{H}^2$-matrice koje se razlikuju od prethodne dvije u tome što se njihovi prihvatljivi vandijagonalni blokovi ne pohranjuju nezavisno već su reprezentacije niskog ranga na svakoj razini ugniježđene.

Primjena hijerarhijskih formata matrica važna je posebno kod diskretizacije diferencijalnih i integralnih jednadžbi. No, to prelazi okvire ovog diplomskog rada. Zadržat ćemo se uglavnom na primjenama kojima je naglasak na linearnoj algebri.

Poglavlje 1

Motivacija i opis ideje

U ovom poglavlju opisat ćemo *rijetko popunjene* (engl. *sparse*) i *podatkovno rijetke* matrice (engl. *data-sparse*). Na kraju ćemo opisati ideju za jedan hijerarhijski format, tzv. HODLR (*Hierarchically Off-Diagonal Low Rank*) format, i to na primjeru inverza tridijagonalnih matrica.

1.1 Rijetko popunjene matrice

Kažemo da je matrica $A \in \mathbb{R}^{n\times n}$ *rijetko popunjena* ako je broj elemenata matrice koji nisu nula mnogo manji od n^2. Matrice koje nisu rijetko popunjene nazivamo *gusto popunjene* (engl. *dense*) matrice. Dok je broj elemenata koji nisu nula gusto popunjene matrice $O(n^2)$, broj je ne-nul elemenata rijetko popunjene $O(n)$. Rijetkost matrice može biti vrlo korisno svojstvo, posebno kod matrica velikih dimenzija. Rijetko popunjene matrice se mogu se u računalu pohraniti na razne načine koji bi uštedili prostor. Jedan od načina je pohrana tri polja: *rows*, *columns* i A. Prvi sadrži indekse redaka, drugi stupaca, a treći elemente matrice koji nisu nula i to tako da uređena trojka $(rows[k], columns[k], A[k])$ jedinstveno određuje položaj i vrijednost elemenata matrice, za svaki $k \in \mathbb{N}, k < nnz(A)$, gdje je $nnz(A)$ broj ne-nul elemenata matrice A. Na ovaj način pohranjuje se $3nnz(A)$, tj. $O(n)$, za razliku od uobičajenih $O(n^2)$ elemenata. Također, neke algoritme poput množenja matrice i vektora je moguće efikasnije implementirati. Složenost će biti $O(n \cdot z)$, gdje je z prosječni broj ne-nul elemenata u svakom retku matrice. Razlikujemo *nestrukturirane* i *strukturirane* rijetko popunjene matrice. Posebna su vrsta strukturiranih matrica *vrpčaste* matrice (engl. *banded matrix*). Matrica A je *vrpčasta* ako postoji $d > 0$ tako da vrijedi $a_{ij} = 0$ za $|i - j| > d$. Zanimljiv će nam slučaj biti $d = 1$, tj. *tridijagonalna* matrica.

Nažalost, svojstvo rijetke popunjenosti se može lako narušiti određenim operacijama. Na primjer, trokutaste matrice dobivene LU faktorizacijom rijetko popunjene matrice obično imaju mnogo manje nula od zadane matrice. Ovo također vrijedi i za inverz rijetko popu-

njene matrice.

Primjer 1.1.1. *Primjer inverza* 8×8 *tridijagonalne matrice. Na ovom primjeru vidimo da inverz takve matrice, koja je rijetko popunjena, nije rijetko popunjena matrica.*

$$A = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \end{bmatrix} \quad A^{-1} = \begin{bmatrix} -1 & 3/2 & 5/2 & -2 & 3/2 & -1/2 & -1/2 & 1/2 \\ 3 & -3 & -5 & 4 & -3 & 1 & 1 & -1 \\ 5 & -5 & -10 & 8 & -6 & 2 & 2 & -2 \\ -8 & 8 & 16 & -12 & 9 & -3 & -3 & 3 \\ 6 & -6 & -12 & 9 & -6 & 2 & 2 & -2 \\ -4 & 4 & 8 & -6 & 4 & -1 & -1 & 1 \\ -8 & 8 & 16 & -12 & 8 & -2 & -3 & 3 \\ 16 & -16 & -32 & 24 & -16 & 4 & 6 & -5 \end{bmatrix}$$

Primjer 1.1.2. *Primjer LU faktorizacije* 6×6 *rijetko popunjene matrice. Gornjetrokutasti i donjetrokutasti faktor nisu rijetko popunjene matrice.*

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1- & -1 & 1 & 0 & 0 & 0 \\ 1 & -1 & 2 & 1 & 0 & 0 \\ 1 & -1 & 2 & 2/3 & 1 & 0 \\ 1 & -1 & 2 & 2/3 & 2/5 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & -1 & -1 & -1 \\ 0 & 0 & -1 & -2 & -2 & -2 \\ 0 & 0 & 0 & 3 & 2 & 2 \\ 0 & 0 & 0 & 0 & 5/3 & 2/3 \\ 0 & 0 & 0 & 0 & 0 & 7/5 \end{bmatrix}$$

Problem se u nekim slučajevima može riješiti *približnom rijetkosti*, tj. A^{-1} se može aproksimirati rijetkom matricom M. U tom slučaju vrijedi

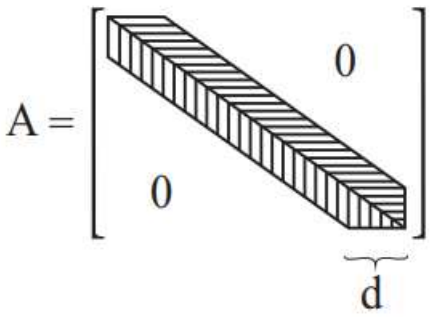
$$||A^{-1} - M|| \leq tol,$$

za maleni $tol > 0$ i neku matričnu normu. Pokazuje se da se inverz tridijagonalne matrice A, ako postoji, može aproksimirati vrpčastom matricom ako je dobro uvjetovana, tj. ako je uvjetovanost matrice A, $\kappa(A) = ||A||_2||A^{-1}||_2$ mala (vidi [1]). No, ni ovo nije uvijek primjenjivo.

Napomena 1.1.3. *Iako se ova točka odnosila na kvadratne rijetko popunjene matrice, analogni opis vrijedi i za pravokutne matrice.*

1.2 Podatkovno rijetke matrice

Kažemo da je matrica $A \in \mathbb{R}^{n \times n}$ *podatkovno rijetka* ako se može prikazati s puno manje od n^2 elemenata. Podatkovno rijetka matrica predstavlja generalizaciju rijetke matrice.



Slika 1.1: Vrpčasta matrica. Izvor: [4].

Najpoznatiji primjer podatkovno rijetke matrice je matrica niskog ranga, tj. matrica ranga $k << n$. Ona se može prikazati u obliku UV^T, $U, V \in \mathbb{R}^{n\times k}$. Na ovaj će način prostorna složenost pohrane matrice biti $O(nk)$. Međutim, postoje i primjeri podatkovno rijetkih matrica, a koje su ranga n i koje nisu rijetke. Inverz tridijagonalne matrice je, ako postoji, regularna matrica, pa je očito punog ranga. Na Primjeru 1.1.1 vidimo da inverz tridijagonalne matrice ne mora biti rijetko popunjena matrica. Pokažimo kako je možemo pohraniti u prostornoj složenosti $O(n \log n)$.

Neka je A tridijagonalna matrica. Sherman-Morrisonova formula[1], koju ćemo koristiti u ovom razmatranju, glasi: ako je $A \in \mathbb{R}^{n\times n}$ regularna matrica i $u, v \in \mathbb{R}^n$, onda je $A + uv^T$ regularna matrica ako i samo ako vrijedi $1 + v^T A^{-1}u \neq 0$. U tom slučaju vrijedi:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + v^T A^{-1}u}. \tag{1.1}$$

Radi jednostavnosti pretpostavimo da je A simetrična i pozitivno definitna[2] tridijagonalna matrica. Matricu A možemo prikazati na sljedeći način:

$$A = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix} - a_{n_1,n_1+1} \begin{bmatrix} e_{n_1} \\ -e_1 \end{bmatrix} \begin{bmatrix} e_{n_1} \\ -e_1 \end{bmatrix}^T, \qquad A_{11} \in \mathbb{R}^{n_1\times n_1}, A_{22} \in \mathbb{R}^{n_2\times n_2},$$

gdje je e_j j-ti jedinični vektor odgovarajuće veličine. Tada prema (1.1) vrijedi:

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 \\ 0 & A_{22}^{-1} \end{bmatrix} + \frac{a_{n_1,n_1+1}}{1 + e_{n_1}^T A_{11}^{-1} e_{n_1} + e_1^T A_{22}^{-1} e_1} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}^T, \tag{1.2}$$

[1] Više o formuli na [7, str. 81-82.].

[2] $A \in \mathbb{R}^{n\times n}$ je pozitivno definitna ako vrijedi $x^T Ax > 0$ za svaki $x \in \mathbb{R}^n \setminus \{0\}$.

gdje je $w_1 = A_{11}^{-1} e_{n_1}$ i $w_2 = -A_{22}^{-1} e_1$. Zaključujemo da su vandijagonalni blokovi A^{-1} ranga 1. Induktivno možemo zaključiti za vandijagonalne blokove dijagonalnih blokova.
Neka imamo sljedeću particiju blokova matrice $B := A^{-1} \in \mathbb{R}^{n\times n}$, uz pretpostavku $n = 2^p$, $p \in \mathbb{N}$:

$$B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$

Matrica B podijeljena je na blokove jednake veličine. Pohranjujemo samo vandijagonalne blokove B_{12} i B_{21} koji su ranga 1 u obliku umnoška UV^T, $U, V \in \mathbb{R}^{n/2\times 1}$. Tako se pohrana vandijagonalnog bloka smanjila na $2 \cdot n/2$ prostornih jedinica, npr. varijabli tipa *double*. Analogno pohranjujemo vandijagonalne blokove podmatrica B_{11} i B_{22} koji su prema (1.2) opet ranga 1. Rekurzivno ponavljamo postupak. Dakle, cijelu matricu B možemo pohraniti sa:

$$2n/2 + 4n/4 + ... + 2^p n/2^p = n(\log n + 1)$$

jedinica prostorne složenosti.

Opisani način pohranjivanja inverza tridijagonalne matrice jedan je primjer strukture tzv. HODLR matrica koje će detaljnije biti opisane u Poglavlju 3. No, neće uvijek biti slučaj da su vandijagonalni blokovi egzaktno niskog ranga. Ono čime ćemo se baviti u idućem poglavlju bit će *aproksimacije niskog ranga* koje će se primjenjivati u mnogo realističnijim slučajevima od navedenog primjera.

Poglavlje 2

Aproksimacije niskog ranga

Aproksimacija niskog ranga je važan problem minimizacije u numeričkoj linearnoj algebri. Problem glasi: za danu matricu A i rang k pronaći matricu $\hat{A}$ ranga k tako da je greška aproksimacije matrice A matricom $\hat{A}$ minimalna.

U ovom poglavlju prvo ćemo opisati osnovu aproksimacije niskog ranga - dekompoziciju na singularne vrijednosti. U nastavku poglavlja dotaknut ćemo se nekih algoritama za navedenu aproksimaciju.

2.1 SVD i najbolja aproksimacija niskog ranga

Teorem 2.1.1 (**SVD - Dekompozicija na singularne vrijednosti**). *Neka je $A \in \mathbb{R}^{m\times n}$ i $m \geq n$. Tada postoje ortogonalne*[1] *matrice $U \in \mathbb{R}^{m\times m}$ i $V \in \mathbb{R}^{n\times n}$ tako da:*

$$A = U\Sigma V^T, \qquad \Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ & 0 & \end{bmatrix} \in \mathbb{R}^{m\times n},$$

i $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$.

Definicija 2.1.2. *Brojeve $\sigma_1, \sigma_2,...,\sigma_n$ nazivamo **singularne vrijednosti** matrice A. Stupce matrice $U = [u_1, ..., u_m]$, nazivamo **lijevi**, a stupce matrice $V = [v_1, ..., v_n]$, **desni singularni vektori** matrice A. Rastav $A = U\Sigma V^T$ nazivamo **dekompozicija na singularne vrijednosti** (SVD) matrice A.*

Napomena 2.1.3. *U slučaju $m < n$, SVD definiramo za matricu A^T.*

[1] Matrica $A \in \mathbb{R}^{n\times n}$ je ortogonalna ako vrijedi $A^TA = I$.

Primjer 2.1.4. *SVD* 3×2 *matrice A.*

$$A = \begin{bmatrix} 3 & 4 \\ 2 & 1 \\ 3 & 2 \end{bmatrix}, \quad U = \begin{bmatrix} -0.6291 & 0.7669 & -0.127 \\ 0.5568 & 0.3306 & -0.762 \\ 0.5424 & 0.5501 & 0.6350 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1.2222 & 0 \\ 0 & 6.4425 \\ 0 & 0 \end{bmatrix}, \quad V = \begin{bmatrix} 0.6982 & 0.7159 \\ -0.7159 & 0.6982 \end{bmatrix}.$$

Napomena 2.1.5. *Može se pokazati da su singularne vrijednosti matrice A kvadratni korijeni svojstvenih vrijednosti matrice* A^TA *i* AA^T*. Obje su matrice simetrične i pozitivno semidefinitne*[2]*, pa su njihove svojstvene vrijednosti nenegativne. Stupci matrice V su svojstveni vektori matrice* A^TA*, a stupci matrice U svojstveni vektori matrice* AA^T*. Ako je A simetrična, tj.* $A = A^T$*, tada su njezine singularne vrijednosti jednake apsolutnim vrijednostima njezinih svojstvenih vrijednosti.*

Napomena 2.1.6. *Singularne vrijednosti su jedinstveno određene matricom A.*

Dekompozicija na singularne vrijednosti ima mnogo primjena. Jedna od njih je računanje ranga matrice. U i V su ortogonalne, pa su i regularne prema definiciji. Dakle, rang matrice $A = U\Sigma V^T$ jednak je rangu matrice Σ, tj. broju singularnih vrijednosti matrice A koji nisu nula.
Ovo su još neka svojstva i primjene SVD-a:

- $||A||_2 = \sigma_1$,
- $||A||_F = \sqrt{\sigma_1^2 + ... + \sigma_n^2}$,
- $||A^{-1}||_2 = 1/\sigma_n$ ako je A regularna,
- *uvjetovanost* regularne matrice A, $\kappa(A) = ||A||_2||A^{-1}||_2$, jednaka je σ_1/σ_n,
- za $A \in \mathbb{R}^{m\times n}, m \geq n$, ako je $rang(A) = r$, tada prvih r stupaca matrice U čini ortonormiranu bazu za $\mathrm{Im}(A)$,
- za $A \in \mathbb{R}^{m\times n}, m \geq n$, ako je $rang(A) = r$, tada zadnjih $n - r$ stupaca matrice V čini ortonormiranu bazu za $\mathrm{Ker}(A)$,
- rješavanje problema najmanjih kvadrata $\min_x ||Ax - b||$,
- kompresija slike,
- najbolja aproksimacija niskog ranga koju ćemo sada obraditi.

[2] $A \in \mathbb{R}^{n\times n}$ je pozitivno semidefinitna ako vrijedi $x^TAx \geq 0$ za svaki $x \in \mathbb{R}^n \setminus \{0\}$.

Neka je $k \leq n$ i neka su:

$$U_k \coloneqq \begin{bmatrix} u_1 & \dots & u_k \end{bmatrix}, \quad \Sigma_k \coloneqq \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix}, \quad V_k \coloneqq \begin{bmatrix} v_1 & \dots & v_k \end{bmatrix}.$$

Definirajmo:

$$\mathcal{T}_k(A) \coloneqq U_k \Sigma_k V_k^T. \tag{2.1}$$

Matrica $\mathcal{T}_k(A)$ očito je ranga k i vrijedi:

$$\|\mathcal{T}_k(A) - A\|_2 = \sigma_{k+1}, \tag{2.2}$$

$$\|\mathcal{T}_k(A) - A\|_F = \sqrt{\sigma_{k+1}^2 + ... + \sigma_n^2}. \tag{2.3}$$

Sljedeći teorem osnova je za daljnje razmatranje.

Teorem 2.1.7 (**Schmidt-Mirsky**). *Za matricu $A \in \mathbb{R}^{m\times n}$ neka je $\mathcal{T}_k(A)$ definiran kao u* (2.1)*. Tada vrijedi:*

$$\|A - \mathcal{T}_k(A)\| = \min\{\|A - B\| : B \in \mathbb{R}^{m\times n} \quad \textit{je ranga} \quad k\}.$$

Ovo vrijedi za svaku unitarno invarijantnu normu $\|\cdot\|$. [3]

Napomena 2.1.8. *Ako vrijedi $\sigma_k > \sigma_{k+1}$, tada je aproksimacija ranga k jedinstvena. Za $\sigma_k = \sigma_{k+1}$ to nije slučaj. Npr. matrica:*

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ima nekoliko najboljih aproksimacija ranga 2*:*

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Pitanje koje se nameće je kako odabrati rang k tako da vrijedi $\|A - \mathcal{T}_k(A)\| \leq \epsilon$, za neki $\epsilon > 0$.

[3] Spektralna norma $\|\cdot\|_2$ i Frobeniusova norma $\|\cdot\|_F$ su unitarno invarijantne.

Numerički rang

Rang matrice definiramo kao broj linearno nezavisnih stupaca ili redaka. Ta dva broja su jednaka. No, može se dogoditi i da nakon samo male promjene (perturbacije) stupci ili retci postanu linearno zavisni, što može stvoriti probleme kod računanja u aritmetici računala. Dakle, prema [8], možemo definirati *numerički rang* matrice A, u oznaci r_ϵ kao broj redaka matrice A koji su linearno nezavisni za bilo koju promjenu E koja je po normi manja ili jednaka ϵ.

$$r_\epsilon = r_\epsilon(A, \epsilon) = \min_{\|E\|_2 \leq \epsilon} rang(A + E) \tag{2.4}$$

Vidjeli smo da je rang matrice A jednak broju singularnih vrijednosti te matrice koji nisu nula. SVD se također može upotrijebiti i za računanje numeričkog ranga. Iz Teorema 2.1.7 i (2.2) vidimo da je najmanja udaljenost matrice A do najbliže matrice ranga k, tj. $\mathcal{T}_k(A)$, jednaka σ_{k+1}. Stoga, slijedi da je numerički rang broj singularnih vrijednosti koje su veće od nekog praga ϵ.

$$\sigma_{r_\epsilon} > \epsilon \geq \sigma_{r_\epsilon+1}. \tag{2.5}$$

Numerički rang ima smisla određivati kod matrice koja ima jasan razmak između singularnih vrijednosti, tj. između relativno velikih i relativno malih singularnih vrijednosti. Ako je kvadratna matrica punog ranga i loše uvjetovana, tj. σ_1/σ_n je dovoljno velik (jer je σ_n malena vrijednost) tada je njezin numerički rang manji od stvarnog ranga.

Primjer 2.1.9. *Primjer matrice punog ranga s gotovo linearno zavisnim retcima.*

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 5 & 2 & 7 \\ 6 & 4 & 9.999 \end{bmatrix}$$

Njene singularne vrijednosti su:

$$\sigma_1 = 15.5434, \qquad \sigma_2 = 1.54398, \qquad \sigma_3 = 0.000333352.$$

Ako bismo uzeli $\epsilon = 10^{-3}$*, njen numerički rang jednak je* 2*. Najbliža matrica ranga* 2 *joj je:*

$$\mathcal{T}_2(A) = \begin{bmatrix} 1.00011 & 2.00011 & 2.99989 \\ 5.00011 & 2.00011 & 6.99989 \\ 5.99989 & 3.99989 & 9.99911 \end{bmatrix}$$

i vrijedi:

$$\|A_2 - A\|_2 = 0.000333352 = \sigma_3.$$

2.2 Algoritmi za aproksimaciju niskog ranga

U ovoj ćemo točki razmotriti najpoznatije algoritme za aproksimaciju niskog ranga zadane matrice. Odabir optimalnog algoritma ovisi o veličini i gustoći matrice. No, moramo imati na umu da ovi algoritmi ne mogu vratiti dobru aproksimaciju zadane matrice ako njene singularne vrijednosti ne opadaju dovoljno brzo.

Algoritmi koji se zasnivaju na SVD-u

Za početak, pogledajmo kako izračunati SVD. U programskom jeziku Python za računanje koristimo rutinu iz paketa numpy.linalg, a koja se oslanja na biblioteku LAPACK . Za matricu $A \in \mathbb{R}^{m\times n}$ i $m \geq n$ složenost računanja SVD-a je $O(mn^2)$.

$\mathcal{T}_k(A)$ se neće pohranjivati eksplicitno, nego kao dva faktora $U_k\Sigma_k$ i V_k. Takva pohrana zahtijeva $(m + n)k$ jedinica pohrane. Velika prednost SVD-a je ta što znajući singularne vrijednosti, lako možemo odrediti željeni rang k za dobru aproksimaciju. To postižemo odbacujući jako male singularne vrijednosti.

Ako matrica nije zadana eksplicitno nego je faktorizirana u obliku $A = BC^T$, aproksimacija niskog ranga preko SVD-a se računa na sljedeći način. Neka je matrica A zadana na sljedeći način:

$$A = BC^T, B \in \mathbb{R}^{m\times K}, C \in \mathbb{R}^{n\times K}, \tag{2.6}$$

gdje je K veći od željenog ranga k, a mnogo manji od m i n. Aproksimacija ranga k se računa sljedećim redoslijedom:

1. Izračunaj reduciranu QR dekompoziciju $B = Q_BR_B$ i $C = Q_CR_C$.
2. Izračunaj $\mathcal{T}_k(R_BR_C^T) = \widetilde{U_k}\Sigma_k\widetilde{V_k}^T$.
3. $U_k = Q_B\widetilde{U_k}$, $V_k = Q_C\widetilde{V_k}$.
4. Vrati $\mathcal{T}_k(A) \coloneqq U_k\Sigma_kV_k^T$.

Algoritam vraća najbolju aproksimaciju ranga k matrice A. Složenost algoritma je $O((m + n)K^2)$. Ovaj postupak najčešće se koristi za kompresiju faktora pri zbrajanju matrica niskog ranga o kojoj će biti riječ u Poglavlju 3.

Napomena 2.2.1. ***QR dekompoziciju*** *definiramo kao prikaz matrice $A \in \mathbb{R}^{m\times n}$ u obliku $A = QR$, gdje je $Q \in \mathbb{R}^{m\times m}$ matrica s ortonormiranim stupcima, a $R \in \mathbb{R}^{m\times n}$ gornjetrokutasta matrica. Za računanje najčešće se koriste Householderove transformacije složenosti $O(mn^2)$ (vidi [16, str. 81.-87.]). Ako pretpostavimo $m \geq n$, tada rastav matrice na $Q \in \mathbb{R}^{m\times n}$ i $R \in \mathbb{R}^{n\times n}$ nazivamo* ***reduciranom QR dekompozicijom****.*

Lanczosov algoritam

Iako je SVD koristan zbog toga što daje najbolju aproksimaciju niskog ranga, za matrice prevelikih dimenzija to računanje postaje preskupo. U mnogim primjenama neće ni biti važna toliko visoka točnost aproksimacije koliko manja složenost algoritma. Također, SVD ne može iskoristiti svojstvo rijetkosti matrice. Lanczosov algoritam, koji će biti opisan u ovoj točki, umjesto transformiranja matrice A koristit će samo operaciju množenja matrice A nekim vektorom. Množenje matrice i vektora je mnogo manje složenosti ako je matrica rijetka. O tome je bilo riječi u Poglavlju 1.

Definicija 2.2.2. *Neka je $A \in \mathbb{R}^{n\times n}$ i $b \in \mathbb{R}^n$. **Krylovljev potprostor** reda m generiran matricom A i vektorom b, u oznaci $\mathcal{K}_m(A, b)$, definiramo kao linearnu ljusku od $\{b, Ab, ..., A^{m-1}b\}$.*

Neka je $A \in \mathbb{R}^{m\times n}$ matrica koju želimo aproksimirati. Lanczosov algoritam računa matrice $U_{K+1} \in \mathbb{R}^{m\times(K+1)}$ i $V_{K+1} \in \mathbb{R}^{n\times(K+1)}$ takve da su njihovi stupci ortonormirane baze za $\mathcal{K}_{K+1}(AA^T, u_1)$ i $\mathcal{K}_{K+1}(A^TA, v_1)$ redom, gdje je $u_1 \in \mathbb{R}^m$ slučajni normirani vektor i $v_1 = A^Tu_1/\|A^Tu_1\|$. Također, ovaj algoritam generirat će skalare α_j i β_j koji će biti poredani u bidijagonalnu matricu:

$$B_K = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_K & \alpha_K \end{bmatrix} \tag{2.7}$$

Ako želimo odbaciti male singularne vrijednosti matrice A i tako izračunati njezin numerički rang k, prema [13] dovoljno je izračunati SVD matrice B_K . Singularne vrijednosti matrice B_K dobra su aproksimacija singularnih vrijednosti matrice A. Singularni vektori matrice B_K kombinirani s matricama U_K i V_K koriste se kao aproksimacija lijevih i desnih singularnih vektora matrice A. Tada vrijedi:

$$\mathcal{T}_k(A) \approx A_K := U_K\mathcal{T}_k(B_K)V_K^T. \tag{2.8}$$

U Algoritmu 1. naveden je pseudokod za Lanczosov algoritam. Vrijednost K biramo na temelju *a priori* procjene ranga zadane matrice. Sljedeća lema daje nam ocjenu greške aproksimacije (2.8).

Lema 2.2.3 ([1, Lema 1.]). *Za aproksimaciju iz Algoritma 1 vrijedi:*

$$\|A_K - A\|_F \leq \sqrt{\sigma_{k+1}(B_K)^2 + ... + \sigma_K(B_K)^2} + \omega_K,$$

gdje je $\omega_K^2 = \|A\|_F^2 - \alpha_1^2 \sum_{j=2}^K(\alpha_j^2 + \beta_j^2)$.

Algoritam 1: Lanczosov algoritam

Ulaz: Matrica $A \in \mathbb{R}^{m\times n}$, vektor $u_1 \in \mathbb{R}^m$ tako da vrijedi $\|u_1\|_2 = 1$, prirodni broj $K \leq \min\{m, n\}$

Izlaz: Matrice $\widetilde{U}_k \in \mathbb{R}^{m\times k}$, $\widetilde{V}_k \in \mathbb{R}^{n\times k}$ koje imaju ortonormirane stupce, dijagonalna matrica $\widetilde{\Sigma}_k \in \mathbb{R}^{k\times k}$ tako da vrijedi $\mathcal{T}_k(A) \approx \widetilde{U}_k\widetilde{\Sigma}_k\widetilde{V}_k^T$

$\widetilde{v} = A^T u_1, \alpha_1 = \|\widetilde{v}\|_2, v_1 = \widetilde{v}/\alpha_1$

for $j = 1,...,K$ **do**

  $\widetilde{u} = Av_j - \alpha_j u_j, \beta_{j+1} = \|\widetilde{u}\|_2, u_{j+1} = \widetilde{u}/\beta_{j+1}$

  $\widetilde{v} = A^T u_{j+1} - \beta_{j+1}v_j, \alpha_{j+1} = \|\widetilde{v}\|_2, v_{j+1} = \widetilde{v}/\alpha_{j+1}$

$U_K = [u_1, ..., u_K], V_K = [v_1, ..., v_K]$

Konstruiraj bidijagonalnu matricu B_K prema (2.7)

Izračunaj SVD od $B_K = \hat{U}\hat{\Sigma}\hat{V}^T$

Odbaci singularne vrijednosti manje od nekog praga, neka je broj preostalih k

$\widetilde{U}_k = U_K\hat{U}(:, 1:k), \widetilde{\Sigma}_k = \hat{\Sigma}(1:k, 1:k), \widetilde{V}_k = V_K\hat{V}(:, 1:k)$

Dokaz. Zbog (2.3), nejednakosti trokuta i očuvanju norme pri množenju s ortogonalnom matricom vrijedi:

$$\|A_K - A\|_F = \|U_K\mathcal{T}_k(B_K)V_K^T - A\|_F$$

$$= \|U_K(\mathcal{T}_k(B_K) - B_K)V_K^T + U_K B_K V_K^T - A\|_F$$

$$\leq \sqrt{\sigma_{k+1}^2 + ... + \sigma_K^2} + \|U_K B_K V_K^T - A\|_F.$$

Zbog ortogonalnosti vrijedi $\|A\|_F^2 = \|B_K\|_F^2 + \|U_K B_K V_K^T - A\|_F^2$, pa možemo ograničiti drugi pribrojnik tako da vrijedi tvrdnja. □

Složenost Algoritma 1. za rijetko popunjene matrice je $O((m + n)K + K^3)$, gdje je $O((m+n)K)$ složenost množenja rijetko popunjene matrice s m redaka i vektora i množenja rijetko popunjene matrice s n redaka i vektora u petlji koja se izvršava K puta. Ovo vrijedi jer smo pretpostavili da je broj ne-nul elemenata u svakom retku matrice konstantan. $O(K^3)$ je složenost SVD-a $K \times K$ matrice.

Randomizirani algoritam

Za algoritam kažemo da je *randomiziran* ako koristi slučajne brojeve. U ovoj točki ukratko ćemo opisati jedan od randomiziranih algoritama za aproksimaciju niskog ranga.

U Lanczosovom se algoritmu za svako množenje matrice i vektora iz memorije dohvaća cijela matrica A, što znatno otežava izvršavanje na računalu. Postoje neka poboljšanja poput blok-Lanczosovog algoritma[4], ali najjednostavnija je alternativa korištenje nekog od randomiziranih algoritama. U Algoritmu 2. nalazi se pseudokod za jedan od njih.
U algoritmu koristimo *Gaussovu matricu*, tj. matricu čiji elementi su slučajni brojevi ge-

Algoritam 2: Randomizirani algoritam za aproksimaciju niskog ranga

Ulaz: Matrica $A \in \mathbb{R}^{m\times n}$, željeni rang k, parametar uzorkovanja $p \geq 0$
Izlaz: Matrice $\widetilde{U}_k \in \mathbb{R}^{m\times k}$, $\widetilde{V}_k \in \mathbb{R}^{n\times k}$ koje imaju ortonormirane stupce, dijagonalna matrica $\widetilde{\Sigma}_k \in \mathbb{R}^{k\times k}$ tako da vrijedi $\mathcal{T}_k(A) \approx \hat{A} := \widetilde{U}_k\widetilde{\Sigma}_k\widetilde{V}_k^T$

1 Generiraj standardnu Gaussovu matricu $\Omega \in \mathbb{R}^{n\times(k+p)}$
2 $Y = A\Omega$
3 Izračunaj reduciranu QR dekompoziciju $Y = QR$
4 $Z = Q^TA$
5 Izračunaj SVD od $Z = \hat{U}\hat{\Sigma}\hat{V}^T$
6 $\widetilde{U}_k = Q\hat{U}(:, 1:k)$, $\widetilde{\Sigma}_k = \hat{\Sigma}(1:k, 1:k)$, $\widetilde{V}_k = \hat{V}(:, 1:k)$

nerirani standardnom Gaussovom distribucijom. Prema [6], slika matrice Q čini dobru aproksimaciju slike matrice A.

Dokažimo korektnost algoritma. Ako je matrica A ranga k, dovoljno je uzeti $p = 0$. Neka je A faktorizirana u obliku BC^T, gdje je $B \in \mathbb{R}^{m\times k}$, a $C \in \mathbb{R}^{n\times k}$. S vjerojatnošću 1 vrijedi da je $C^T\Omega \in \mathbb{R}^{k\times k}$ regularna. Iz tog i $Y = A\Omega = B(C^T\Omega)$ slijedi da Y i B razapinju isti prostor. Tada i Q razapinje isti taj prostor, što znači da je QQ^T ortogonalni projektor na vektorski prostor razapet stupcima matrice B. Zbog toga vrijedi $QQ^T = B$. Iz tog slijedi $\hat{A} = QQ^TA = (QQ^TB)C^T = BC^T = A$.

Kada matrica A nije ranga k, ali je aproksimiramo matricom ranga k, tada je poželjno uzeti parametar $p > 0$. U praksi, ako ne znamo koliki je rang k, p se bira na način da je greška aproksimacije što manja. što je veći p, time je i greška manja. To vrijedi zbog sljedećeg teorema koji nam daje ocjenu greške aproksimacije iz Algoritma 2.

Teorem 2.2.4 ([6, Tm 1.1]). *Neka je $k \geq 2$ i $p \geq 2$ takvi da vrijedi $k + p \leq \min\{m, n\}$. Tada za $\hat{A}$ dobivenu u Algoritmu 2 vrijedi:*

$$\mathbb{E}\|A - \hat{A}\|_2 \leq (2 + \frac{4\sqrt{(k+p)\min\{m,n\}}}{p-1})\sigma_{k+1},$$

gdje je s $\mathbb{E}\|A - \hat{A}\|_2$ označeno očekivanje greške aproksimacije.

[4] Vidi [15].

Složenost Algoritma 2. za rijetko popunjene matrice ovisi o složenosti množenja matrice A i matrice Ω koja ima $k + p$ stupaca i složenosti QR faktorizacije matrice Y reda $n \times (k + p)$. Složenost SVD-a matrice Z zanemarujemo jer pretpostavljamo da je jako malih dimenzija.

Poglavlje 3

Hijerarhijske strukture

U ovom poglavlju opisat ćemo po uzoru na [1] i [11] najčešće hijerarhijske strukture matrica i neke operacije nad njima. Takve strukture smislene su kod matrica čiji vandijagonalni blokovi dopuštaju dobru aproksimaciju niskog ranga. Na kraju poglavlja opisat ćemo implementaciju HODLR matrica i osnovnih operacija s njima u programskom jeziku Python.

3.1 HODLR matrice

Vrlo važan pojam bit će *stablo grupiranja*, pa ćemo ga definirati.

Definicija 3.1.1 (**[11, Def. 1.]**). *Za $n \in \mathbb{N}$, neka je T_I potpuno i uravnoteženo binarno stablo*[1] *visine p čiji čvorovi su podskupovi od $\{1, ..., n\}$. Kažemo da je T_I* ***stablo grupiranja*** *ili* ***klaster stablo*** *ako zadovoljava:*

- *korijen stabla T_I je $I_1^0 := I = \{1, ..., n\}$,*
- *čvorovi na razini l, označeni sa $I_1^l, ..., I_{2^l}^l$ tako da je svaki čvor skup koji se sastoji od nekoliko uzastopnih prirodnih brojeva, čine particiju od I, a iznimno je dozvoljeno da se u particiji nalazi i prazni skup,*
- *čvor I_i^l, $1 \leq l \leq p-1$, ima djecu I_{2i-1}^{l+1} i I_{2i}^{l+1}, a djeca čine particiju svojih roditelja.*

Napomena 3.1.2. *Obično uzimamo da su skupovi koji su djeca istog čvora približno ekvipotentni.*

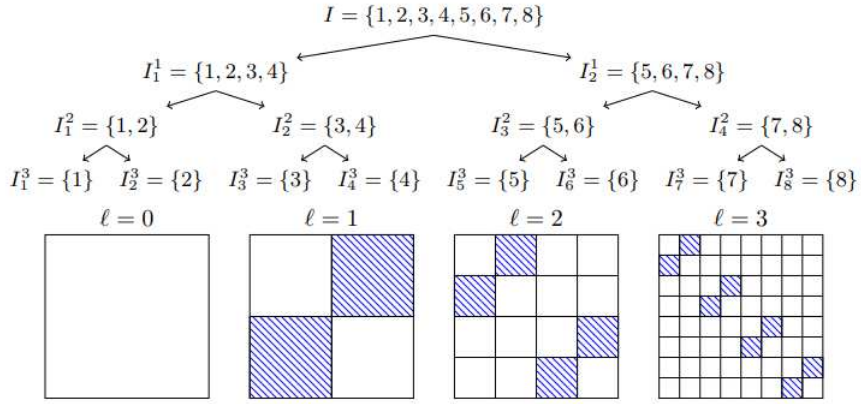
[1] Potpuno binarno stablo je stablo u kojem svaki čvor ima dvoje ili nula djece. Uravnoteženo binarno stablo je stablo u kojem se visine podstabala svakog čvora razlikuju najviše za 1.

HODLR (*Hierarchically Off-Diagonal Low Rank*) matrica je blok-matrica s višerazinskom rekurzivnom strukturom takva da su joj vandijagonalni blokovi niskog ranga. Dakle, za $n \in \mathbb{N}$, HODLR matrica $A \in \mathbb{R}^{n\times n}$ ima sljedeću strukturu: [2]

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

gdje su blokovi A_{12} i A_{21} niskog (numeričkog) ranga, a A_{11} i A_{22} su ponovno HODLR matrice. Rekurzivno particioniranje po blokovima se nastavlja sve do razine koja je određena minimalnom veličinom dijagonalnih blokova $n_{\min} \geq 1$.

Opišimo konstrukciju HODLR matrice. U nastavku teksta oznaka $A(I_i^l, I_j^l)$ označavat će matricu A reduciranu na redove indeksa iz skupa I_i^l i stupce indeksa iz skupa I_j^l gdje je l razina rekurzije na kojoj se trenutačno nalazimo. Također, $A_{ij}^l := A(I_i^l, I_j^l)$. Radi jednostavnosti računanja pretpostavimo $n = 2^p n_{\min}$, za najveću razinu $p \in \mathbb{N}$. Za početak definirajmo skup indeksa redaka i stupaca $I = \{1, ..., n\}$ koji će ujedno biti i korijen stabla grupiranja. Pogledajmo što se događa na svakoj razini.

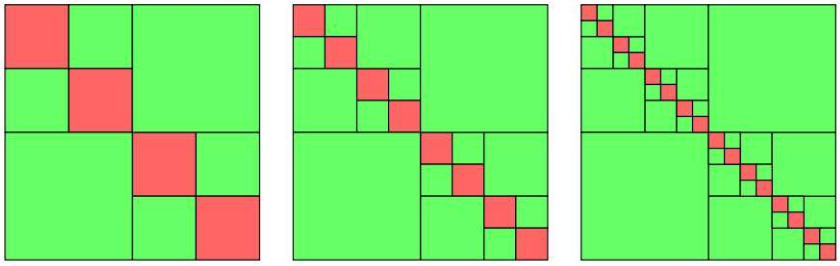


Slika 3.1: Slika prikazuje stablo grupiranja dubine 3 i pripadajuće particioniranje blokova. Blokovi koji su označeni plavom bojom pohranjujemo faktorizirane kako je opisano u (3.1). Izvor:[11].

- Razina $l = 0$. $I^0 := I$ particioniramo na dva podskupa: $I_1^1 = \{1, ..., n/2\}$ i $I_2^1 = \{n/2 + 1, ..., n\}$ i ta dva podskupa su djeca čvora I^0 u stablu grupiranja.

[2] Možemo definirati HODLR strukturu i za pravokutne matrice, ali zbog jednostavnosti, a i s obzirom na to da su neke operacije poput inverza, LU faktorizacije i rješavanja linearnih sustava dozvoljene samo nad kvadratnim matricama, u daljnjem tekstu bavit ćemo se samo njima.

- Razina $l = 1$. Matricu A podijelimo na četiri bloka $A(I_i^1, I_j^1) = A_{ij}^1$, $i, j = 1, 2$. Skupove I_1^1 i I_2^1 particioniramo na ekvipotentne podskupove, tj. na njegovu djecu u stablu grupiranja.
- Razina $l = 2, ..., p-1$. Dijagonalne blokove $A(I_i^l, I_i^l)$, $i = 1, ..., 2^l$, dijelimo na kvadratne blokove veličine $2^{p-l}n_p$. Skupove I_i^l particioniramo na ekvipotentne skupove I_{2i-1}^{l+1} i I_{2i}^{l+1} kao dosad.
- Razina $l = p$. Skupovi $I_1^p, ..., I_{2^p}^p$ listovi su stabla grupiranja. Podijelimo dijagonalne blokove tako da odgovaraju particiji iz prošle razine. Sada su manji blokovi veličine $n_{\min}$. Dijagonalne blokove pohranjujemo kao $n_{\min} \times n_{\min}$ gusto popunjene matrice.



Slika 3.2: Slika prikazuje konstrukciju HODLR matrice za različite dubine rekurzije p. Prikaz za $p = 2, 3, 4$ redom. Izvor:[1].

Pretpostavimo da je svaki vandijagonalni blok $A(I_i^l, I_j^l)$, gdje su I_i^l i I_j^l braća u stablu grupiranja, ranga najviše k. Njih pohranjujemo faktorizirane na sljedeći način:

$$A(I_i^l, I_j^l) = U_i^{(l)}(V_j^{(l)})^T, \qquad U_i^{(l)}, V_j^{(l)} \in \mathbb{R}^{n_l \times k}, \tag{3.1}$$

gdje je n_l veličina bloka na trenutačnoj razini l. Preciznije, $n_l = 2^{p-l}n_{\min}$.

Sada ćemo ocijeniti prostornu složenost ovakve strukture. Primijetimo, na svakoj razini $l > 0$ nalazi se 2^l vandijagonalnih blokova. Oni su pohranjeni kao dvije matrice dimenzije $n_l \times k$. Ako bismo zbrojili prostor za pohranu vandijagonalnih blokova na svakoj razini, dobivamo:

$$\sum_{l=1}^{p} 2k2^l n_l = 2k \sum_{l=1}^{p} 2^l n_l = 2k \sum_{l=1}^{p} 2^l 2^{p-l} n_{\min} = 2k \sum_{l=1}^{p} 2^p n_{\min} =$$

$$2k \sum_{l=1}^{p} n = 2kpn = 2kn \log(n/n_{\min}) \leq 2kn \log n$$

Na razini p potrebno nam je još $2^p n_{\min}^2 = n n_{\min}$ jedinica pohrane za dijagonalne blokove. Ako pretpostavimo $n_{\min} \in O(1)$, tada je ocjena ukupne prostorne složenosti pohranjivanja HODLR matrice $O(kn \log n)$.

Primjer 3.1.3. *Na primjeru* 8×8 *matrice prikazat ćemo konstrukciju HODLR strukture iako se za matrice tako malih dimenzija ta konstrukcija ne koristi. Neka je A tridijagonalna matrica iz Primjera 1.1.1 i* A^{-1} *njezin inverz. U Poglavlju 1. također smo pokazali da su vandijagonalni blokovi matrice* A^{-1} *ranga* 1. *Neka je* $n_{\min} = 1$ *i* $I = \{1, 2, 3, 4, 5, 6, 7, 8\}$ *korijen stabla grupiranja. Slijedi opis konstrukcije HODLR struktrure matrice* A^{-1}.

Razina $l = 0$*:*
Djeca korijena su:

$$I_1^1 = \{1, 2, 3, 4\}, \quad I_2^1 = \{5, 6, 7, 8\}.$$

Razina $l = 1$*:*

$$A(I_1^l, I_2^l) = U_1^{(1)}(V_2^{(1)})^T, \qquad U_1^{(1)} = \begin{bmatrix} -1.73205081 \\ 3.46410162 \\ 6.92820323 \\ -10.39230485 \end{bmatrix}, \qquad V_2^{(1)} = \begin{bmatrix} -0.8660254 \\ 0.28867513 \\ 0.28867513 \\ -0.28867513 \end{bmatrix},$$

$$A(I_2^l, I_1^l) = U_2^{(1)}(V_1^{(1)})^T, \qquad U_2^{(1)} = \begin{bmatrix} -17.23368794 \\ 11.48912529 \\ 22.97825059 \\ -45.95650117 \end{bmatrix}, \qquad V_1^{(1)} = \begin{bmatrix} -0.34815531 \\ 0.34815531 \\ 0.69631062 \\ -0.52223297 \end{bmatrix}.$$

Djeca čvora I_1^1 *su* $I_1^2 = \{1, 2\}$ *i* $I_2^2 = \{3, 4\}$*, a čvora* I_2^1 *su* $I_3^2 = \{5, 6\}$ *i* $I_4^2 = \{7, 8\}$.
Razina $l = 2$*:*

$$A(I_1^2, I_2^2) = U_1^{(2)}(V_2^{(2)})^T, \qquad U_1^{(2)} = \begin{bmatrix} -3.20156212 \\ 6.40312424 \end{bmatrix}, \qquad V_2^{(2)} = \begin{bmatrix} -0.78086881 \\ 0.62469505 \end{bmatrix},$$

$$A(I_2^2, I_1^2) = U_2^{(2)}(V_1^{(2)})^T, \qquad U_2^{(2)} = \begin{bmatrix} -7.07106781 \\ 11.3137085 \end{bmatrix}, \qquad V_1^{(2)} = \begin{bmatrix} -0.70710678 \\ 0.70710678 \end{bmatrix},$$

$$A(I_3^2, I_4^2) = U_3^{(2)}(V_4^{(2)})^T, \qquad U_3^{(2)} = \begin{bmatrix} -2.82842712 \\ 1.41421356 \end{bmatrix}, \qquad V_4^{(2)} = \begin{bmatrix} -0.70710678 \\ 0.70710678 \end{bmatrix},$$

$$A(I_4^2, I_3^2) = U_4^{(2)}(V_3^{(2)})^T, \qquad U_4^{(2)} = \begin{bmatrix} -8.24621125 \\ 16.4924225 \end{bmatrix}, \qquad V_3^{(2)} = \begin{bmatrix} -0.70710678 \\ 0.70710678 \end{bmatrix}.$$

Djeca čvora I_1^2 *su* $I_1^3 = \{1\}$ *i* $I_2^3 = \{2\}$.
Djeca čvora I_2^2 *su* $I_3^3 = \{3\}$ *i* $I_4^3 = \{4\}$.

Djeca čvora I_3^2 *su* $I_5^3 = \{5\}$ *i* $I_6^3 = \{6\}$.
Djeca čvora I_4^2 *su* $I_7^3 = \{7\}$ *i* $I_8^3 = \{8\}$.
Vandijagonalni blokovi na razini 3 *veličine su* $n_{\min} = 1$*, pa se faktoriziraju na trivijalan način. Preostalih* $2^3 = 8$ *blokova pohranjujemo kao* 1×1 *matrice. To su upravo dijagonalni elementi polazne matrice.*

Množenje matrice i vektora

Množenje matrice i vektora $y = Ax$ gdje je $A \in \mathbb{R}^{n\times n}$ HODLR matrica, a $x \in \mathbb{R}^n$ vektor, postižemo rekurzivnim blokovskim množenjem matrice koja je po definiciji već podijeljena na blokove i vektora kojeg dijelimo na blokove. Vektor x podijeljen je na način da dimenzije bloka omogućavaju množenje s određenim blokom matrice.
Na razini $l = 1$ imamo sljedeću situaciju:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 \\ A_{21}x_1 + A_{22}x_2 \end{bmatrix},$$

ili drugačije prikazano:

$$y(I_1^1) = A(I_1^1, I_1^1)x(I_1^1) + A(I_1^1, I_2^1)x(I_2^1),$$
$$y(I_2^1) = A(I_2^1, I_1^1)x(I_1^1) + A(I_2^1, I_2^1)x(I_2^1).$$

Dva su slučaja. Prvi je množenje dijagonalnog bloka, odnosno HODLR matrice s vektorom $A_{11}x_1$ i $A_{22}x_2$ što računamo rekurzivno, a drugi je množenje vandijagonalnog bloka s vektorom $A_{12}x_2$ i $A_{21}x_1$ što, koristeći asocijativnost množenja matrica, računamo na sljedeći način radi manje vremenske složenosti:

$$A(I_1^1, I_2^1)x(I_2^1) = U_1^1((V_2^1)^T x(I_2^1)).$$

Analogno pomnožimo $A(I_2^1, I_1^1)x(I_1^1)$. Na ovaj način za množenje vandijagonalnog bloka veličine n ranga k s vektorom potrebno nam je $4nk$ operacija. Rekurzija se nastavlja do razine $l = p$, gdje je veličina bloka $n_{\min}$. Na toj razini pomnožimo dijagonalni blok s vektorom standardnim algoritmom matričnog množenja.

Označimo broj operacija potrebnih za množenje HODLR matrice i vektora s $c(n)$. Složenost množenja dijagonalnih blokova s vektorom tada je jednaka $c(n/2)$. Za zbrajanje vektora $A_{11}x_1 + A_{12}x_2$ i $A_{21}x_1 + A_{22}x_2$ potrebno nam je n operacija. Imamo sljedeću rekurzivnu relaciju:

$$c(n) = 2c(n/2) + 2 \cdot 4k(n/2) + n.$$

Koristeći Master teorem[3], dobivamo

$$c(n) = (4k + 1)\log(n)n. \tag{3.2}$$

Zbrajanje matrica

Pogledajmo što se događa pri (blokovskom) zbrajanju dvaju HODLR matrica A i B, koje su jednako particionirane, na razini $l = 1$.

$$\begin{bmatrix} A_{11}^1 & A_{12}^1 \\ A_{21}^1 & A_{22}^1 \end{bmatrix} + \begin{bmatrix} B_{11}^1 & B_{12}^1 \\ B_{21}^1 & B_{22}^1 \end{bmatrix} = \begin{bmatrix} A_{11}^1 + B_{11}^1 & A_{12}^1 + B_{12}^1 \\ A_{21}^1 + B_{21}^1 & A_{22}^1 + B_{22}^1 \end{bmatrix}$$

Zbroj vandijagonalnih blokova $A_{12}^1 + B_{12}^1$ računamo na sljedeći način. Radi jednostavnosti pretpostavimo da je rang oba vandijagonalna bloka k. Neka je A_{12}^1 pohranjen kao produkt matrica U_1^l i $(V_2^l)^T$, a B_{12}^1 kao produkt G_1^1 i H_2^1, gdje su $U_1^1, V_2^l, G_1^l, H_2^l \in \mathbb{R}^{n\times k}$. Tada vrijedi:

$$U_1^l(V_2^l)^T + G_1^1(H_2^1)^T = \begin{bmatrix} U_1^l & G_1^1 \end{bmatrix} \begin{bmatrix} V_2^l & H_2^1 \end{bmatrix}^T.$$

Ovim postupkom broj stupaca faktora vandijagonalnih blokova povećao se s k na $2k$, a moglo bi se dogoditi da su ti stupci (skoro) linearno zavisni, odnosno da im je numerički rang manji od broja stupaca. Tada se primjenjuje algoritam za kompresiju faktora (vidi 2.2) . Drugi problem na kojeg nailazimo je prevelik rast ranga vandijagonalnih blokova pri daljnjim zbrajanjima. Tada se primjenjuje kompresija faktora bloka i broj stupaca faktora ograničimo na željenu vrijednost. U nastavku teksta radi jednostavnosti računanja pretpostavit ćemo da je nakon svake kompresije faktora rang vandijagonalnih blokova ograničen sa k. Naravno, pri ovakvom zbrajanju ne dobiva se egzaktni rezultat. Analogno računamo zbroj $A_{21}^1 + B_{21}^1$. Nad dijagonalnim blokovima rekurzivno ponavljamo postupak do razine p, gdje primjenjujemo standardni algoritam zbrajanja matrica.

Izračunajmo broj operacija potrebnih za zbrajanje matrica A i B. Za početak, odredimo broj operacija za zbrajanje vandijagonalnih blokova veličine n. Ona iznosi:

$$c_{LR+LR}(n) = c_{SVD}(nk^2 + k^3).$$

c_{SVD} je generička konstanta koja ovisi o složenosti QR dekompozicije i SVD-a. Na svakoj razini l imamo 2^l vandijagonalnih blokova veličine n_l, pa ukupni broj operacija iznosi:

$$\sum_{l=1}^{p} c_{LR+LR}(n_l) = c_{SVD} \sum_{l=1}^{p} 2^l(k^3 + n_l k^2) \leq$$

[3]Master teorem najbolja je metoda brzog izračuna vremenske složenosti „podijeli pa vladaj" algoritama, a koja je zadana rekurzivnom relacijom. Iskaz i dokaz Master teorema nalazi se u [3, str. 96.-106.].

$$\leq c_{SVD}(2^{p+1}k^3 + \sum_{l=1}^{p} 2^l 2^{p-l} n_{\min} k^2)$$

$$\leq c_{SVD}(2nk^3 + n\log(n)k^2).$$

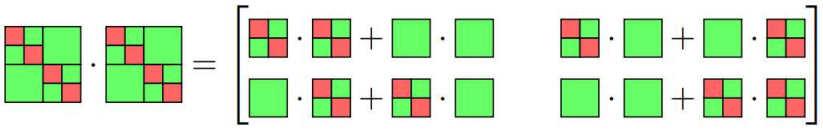
Broj operacija potrebnih za zbrajanje 2^p dijagonalnih blokova na razini p iznosi $2^p n_{\min}^2 = nn_{\min}$. Tada ukupna vremenska složenost zbrajanja dviju HODLR matrica iznosi $O(nk^3 + n\log(n)k^2)$.

Množenje matrica

Označimo sa C umnožak HODLR matrica A i B. Množenje po blokovima izgleda ovako (na razini $l = 1$). Neka je $A_{ij} := A_{ij}^1$ i $B_{ij} := B_{ij}^1$.

$$C = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix} \tag{3.3}$$

Množenje se može prikazati i sljedećom ilustracijom.



Slika 3.3: Slika strukturno opisuje blokovsko množenje HODLR matrica. Izvor: [1].

Kao što se može vidjeti iz (3.3) i slike 3.3, četiri su različita tipa množenja:

1. množenje dviju HODLR matrica veličine $n/2$,
2. množenje dvaju vandijagonalnih blokova,
3. množenje HODLR matrice s vandijagonalnim blokom,
4. množenje vandijagonalnog bloka s HODLR matricom.

Korake 1., 3. i 4. rekurzivno računamo sve do posljednje razine $l = p$ gdje primjenjujemo standardne algoritme množenja matrica. Također, rekurzivno primjenjujemo i kompresiju faktora vandijagonalnih blokova na svakoj razini (za korak 1. i zbrajanje) kako bismo spriječili prevelik rast ranga vandijagonalnih blokova umnoška. Očekivano, rezultat tada ne mora biti egzaktan.

Izračunajmo složenost algoritma množenja matrica veličine n. Neka su $c_{H\cdot H}$, $c_{LR\cdot LR}$, $c_{H\cdot LR}$, $c_{LR\cdot H}$ oznake za broj operacija potrebnih za sve navedenih tipova množenja redom. Neka je c_{H+LR} broj operacija potrebnih za zbrajanje HODLR matrice i matrice niskog ranga, a c_{LR+LR} za zbrajanje dviju matrica niskog ranga (s kompresijom faktora za oba slučaja). Tada je:

$$\begin{aligned} c_{H\cdot H}(n) = 2(&c_{H\cdot H}(n/2) + c_{LR\cdot LR}(n/2) + c_{H\cdot LR}(n/2) \\ &+c_{H\cdot H}(n/2) + c_{H+LR}(n/2) + c_{LR+LR}(n/2)). \end{aligned} \tag{3.4}$$

Broj operacija potrebnih za množenje vandijagonalnih blokova je:

$$c_{LR\cdot LR}(n) = 4nk^2.$$

Broj operacija potrebnih za množenje HODLR matrice i vandijagonalnog bloka (i obratno) ranga k za toliko je puta je veći od množenja matrice s vektorom (vidi (3.2)) jer je postupak sličan.

$$c_{H\cdot LR}(n) = c_{LR\cdot H}(n) = k(4k+1)\log(n)n.$$

Složenost zbrajanja (uz kompresiju faktora) HODLR matrice i matrice niskog ranga jednaka je zbrajanju dviju HODLR matrica:

$$c_{H+LR}(n) = c_{H+H}(n) \in O(nk^3 + n\log(n)k^2).$$

Broj operacija za zbrajanje dviju matrica ranga k iznosi $c_{LR+LR}(n) = k \cdot n$, ali se to zanemaruje.
Ukupan broj operacija (3.4), koji smo izračunali koristeći Master teorem, tada je:

$$c_{H\cdot H}(n) \in O(k^3 n \log n + k^2 n \log^2(n)).$$

Rješavanje linearnih sustava

Rješavanje linearnog sustava $Ax = b$ gdje je A HODLR matrica reda n, a x, $b \in \mathbb{R}^n$ izvršavamo sljedećim redoslijedom:

1. približno izračunamo LU faktorizaciju matrice A,
2. riješimo $Ly = b$, gdje je L donjetrokutasta HODLR matrica,
3. riješimo $Ux = y$, gdje je U gornjetrokutasta HODLR matrica.

Prvo pokažimo kako izračunati korak 2. Blok-matrično zapisano sustav na razini $l = 1$ izgleda ovako:

$$\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \tag{3.5}$$

Ovdje su L_{11} i L_{22} HODLR matrice, a L_{21} je matrica niskog ranga. Računamo sljedećim redoslijedom:

$$L_{11}y_1 = b_1, \tag{3.6}$$

$$c_2 \coloneqq b_2 - L_{21}y_1, \tag{3.7}$$

$$L_{22}y_2 = c_2. \tag{3.8}$$

Koraci (3.6) i (3.8) računaju se rekurzivno do posljednje razine p, gdje primjenjujemo standardne algoritme za rješavanje sustava. Na isti način rješavamo i $Ux = b$.

Izračunajmo složenost rješavanja trokutastih sustava. Računanje (3.7) sastoji se od množenja matrice s vektorom ($2kn$ operacija) i oduzimanja vektora (n operacija). Ukupni broj operacija tada je $(2k + 1)n$. Složenost rješavanja trokutastih sustava stoga zadovoljava sljedeću rekurzivnu relaciju:

$$c(n) = 2c(n/2) + (2k + 1)n.$$

Na razini $l = p$ direktno rješavamo $2^p = n/n_{min}$ sustava reda n_{min}. Rješavanje tih sustava je složenosti $O(n_{min}^3)$.
Ukupna složenost tada iznosi:

$$c(n) \in O(nn_p^2 + kn \log n).$$

Na isti način rješavamo i sustav $LY = B$ (L je HODLR matrica, a B gusta matrica) gdje:

$$L = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}, Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix}, B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$

Prvo rješavamo

$$L_{11}Y_{11} = B_{11}, \qquad L_{11}Y_{12} = B_{12},$$

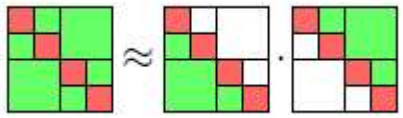
a zatim

$$L_{22}Y_{21} = B_{21} - L_{21}Y_{11}, \qquad L_{22}Y_{22} = B_{22} - L_{21}Y_{12}. \tag{3.9}$$

Primijetimo da (3.9) sadrži matrično HODLR množenje i zbrajanje, a tada koristimo kompresiju faktora vandijagonalnih blokova, pa rezultat uglavnom neće biti egzaktan.
Slično se rješava i sustav $YU = B$, gdje je U gornjetrokutasta HODLR matrica.

LU faktorizacija

Sada ćemo pokazati kako se približna LU faktorizacija računa za HODLR matricu A. L i U su također HODLR matrice. Kao svi algoritmi dosad opisani, i ovaj je rekurzivan.
Na razini $l = 1$ struktura matrica izgleda ovako:



Slika 3.4: Približna LU faktorizacija HODLR matrice. Izvor: [1].

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad L = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}, \quad U = \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix},$$

a iz njih dobivamo jednadžbe:

$$A_{11} = L_{11}U_{11}, \quad A_{12} = L_{11}U_{12}, \quad A_{21} = L_{11}U_{11}, \quad A_{22} = L_{21}U_{12} + L_{22}U_{22}.$$

Računanje se odvija sljedećim redoslijedom:

1. rekurzivno računamo L_{11} i U_{11} faktore od A_{11} (jer su dijagonalni blokovi gornjetrokutaste (donjetrokutaste) matrice opet gornjetrokutaste (donjetrokutaste) matrice),
2. rješavamo $L_{11}U_{12} = A_{12}$,
3. rješavamo $L_{21}U_{11} = A_{21}$,
4. izračunamo faktore L_{22} i U_{22} od $A_{22} - L_{21}U_{12}$.

Ovaj postupak ponavljamo do posljednje razine $l = p$, gdje primjenjujemo standardne algoritme za navedene korake. Primijetimo, na koraku 4. primjenjuje se kompresija faktora pa rezultat ne mora biti egzaktan.

Na sličan način kao za množenje HODLR matrica pokazuje se da složenost LU faktorizacije nije veća od iste.

Invertiranje matrice

Inverz HODLR matrice A je također HODLR matrica koju računamo na sljedeći način:

1. izračunamo LU faktorizaciju matrice A, $A = LU$,
2. izračunamo inverz donjetrokutaste matrice L,
3. izračunamo inverz gornjetrokutaste matrice U,
4. inverz matrice A tada je $A^{-1} = U^{-1}L^{-1}$.

Pokažimo što se događa u 2. koraku. Inverz donjetrokutaste HODLR matrice je donjetrokutasta HODLR matrica i računamo ga na sljedeći način:

$$\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} X_{11} & 0 \\ X_{21} & X_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix},$$

iz čega slijedi:

$$X_{11} = L_{11}^{-1}, \quad X_{22} = L_{22}^{-1}, \quad X_{21} = -X_{22} L_{21} X_{11}.$$

L_{11}^{-1} i L_{22}^{-1} računamo rekurzivno do posljednje razine gdje primjenjujemo Gaussovu metodu za invertiranje matrice. Analogno računamo 3.korak.

Obično izbjegavamo računanje inverza matrice jer algoritam neće dobro raditi za matrice koje su loše uvjetovane. Primjer takve matrice je Hilbertova matrica čiji elementi su definirani s $h_{ij} = 1/(i + j + 1)$. Ako znamo da je algoritam invertiranja moguće numerički stabilno izvesti, postoji efikasan način da ga provedemo.

Transponiranje matrice

Transponiranje HODLR matrice može se napraviti na vrlo jednostavan način.

$$A^T = \begin{bmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{bmatrix}$$

Za vandijagonalne blokove vrijedi:

$$A_{12}^T = (U_{12} V_{12}^T)^T = V_{12} U_{12}^T,$$

$$A_{21}^T = (U_{21} V_{21}^T)^T = V_{21} U_{21}^T.$$

Postupak rekurzivno ponavljamo za dijagonalne blokove, sve do posljednje razine gdje primijenimo standardno transponiranje.

3.2 $\mathcal{H}$-matrice

Particioniranje blokova kakvo srećemo kod HODLR matrica nije uvijek i optimalan izbor. Ono ponekad nije primjenjivo, a ponekad su blokovi prevelikih dimenzija ili prevelikog ranga. Netrivijalno particioniranje blokova koje je prilagođeno situaciji ćemo nazivat ćemo *grupiranje* ili *klasteriranje*. Glavni ciljevi grupiranja blokova matrice $A \in \mathbb{R}^{n \times n}$ su:

1. rangovi vandijagonalnih blokova su mali u odnosu na n,
2. ukupni broj blokova ne smije biti velik.

Spomenut ćemo dvije strategije grupiranja blokova: geometrijsko i algebarsko grupiranje. Oba su motivirana položajem bloka u odnosu na dijagonalu, ali na različite načine. Prvo je motivirano geometrijom samog problema, a drugo grafom incidencije matrice A (ako je A rijetka). $G = (V, E)$ nazivamo *grafom incidencije* matrice A ako je $V = \{1, ..., n\}$ skup indeksa redova i stupaca, a E je definiran na sljedeći način:

$$(i, j) \in E \Leftrightarrow A(i, j) \neq 0.$$

Za potrebe opisa konstrukcije $\mathcal{H}$-matrice bit će nam potrebna sljedeća definicija iz [1].

Definicija 3.2.1. *Za skupove Ω_x, $\Omega_y \subset \mathbb{R}^d$, $d \in \mathbb{N}$, definirajmo:*

$$diam(\Omega_x) \coloneqq \{\max \|x - y\|_2 : x, y \in \Omega_x\},$$

$$dist(\Omega_x, \Omega_y) \coloneqq \{\min \|x - y\|_2 : x \in \Omega_x, y \in \Omega_y\}.$$

Neka vrijedi:

$$\min\{diam(\Omega_x), diam(\Omega_y)\} \leq \eta \cdot dist(\Omega_x, \Omega_y),$$

*za $0 < \eta < 2$. Ovaj uvjet nazivamo **uvjet prihvatljivosti**, a konstantu η **konstanta prihvatljivosti**.*

Kako bismo konstruirali skup prihvatljivih blokova, moramo particionirati skup indeksa I po uzoru na stablo grupiranja iz Definicije 3.1.1. To stablo i uvjet prihvatljivosti koristimo za rekurzivno particioniranje blokova zadane matrice. Na sljedećem primjeru geometrijskog grupiranja pokazat ćemo kako.

Pretpostavimo da želimo naći funkciju $u : \Omega \longrightarrow \mathbb{R}$ tako da je zadovoljena sljedeća integralna jednadžba:

$$\int_0^1 \log|x - y| u(y) dy = f(x), \qquad x \in \Omega = [0, 1],$$

gdje je $f : \Omega \longrightarrow \mathbb{R}$. Neka je $n = 2^p$, za $p \in \mathbb{N}$ i neka je interval $[0, 1]$ podijeljen na manje intervale:

$$\tau_i \coloneqq [(i - 1)/n, i/n], \qquad i = 1, ..., n.$$

Iz Galerkinove diskretizacije[4] s funkcijom $\phi_i(x)$ kojoj je vrijednost 1 ako $x \in \tau_i$, a 0 inače, dobivamo matricu $A \in \mathbb{R}^{n \times n}$:

$$A(i, j) \coloneqq \int_0^1 \int_0^1 \phi_i(x) \log|x - y| \phi_j(y) dy dx = \int_{\tau_i} \int_{\tau_j} \log|x - y| dy dx.$$

[4] Vidi [5, str. 284.].

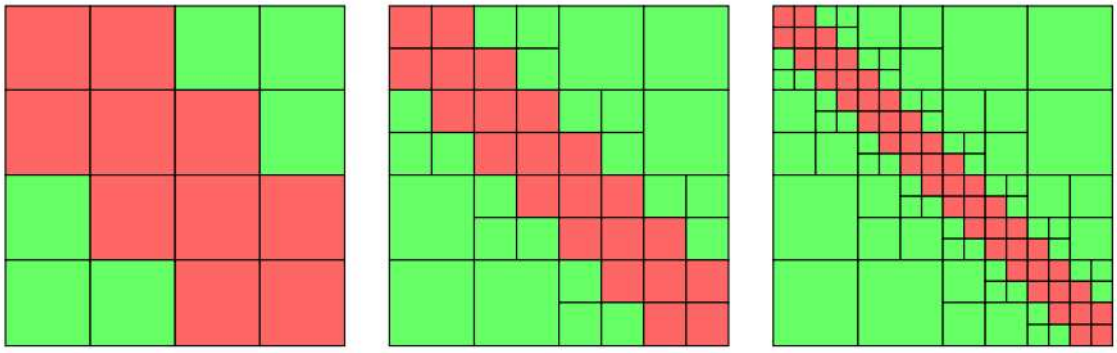
Zbog singulariteta funkcije $\log|x-y|$ za $x = y$ možemo očekivati dobru aproksimaciju niskog ranga samo za one blokove za koje vrijedi da su indeksi svakog elementa dovoljno udaljeni. To nas dovodi do Definicije 3.2.1. Da bismo tu definiciju iskoristili za blokove matrice A, definiramo:

$$\Omega_t := \bigcup_{i \in t} \tau_i \subset \Omega,$$

za skup indeksa $t \subset \{1, ..., n\}$. Stoga, blok (s, t), određen skupom indeksa redaka s i skupom indeksa stupaca t, nazivamo *prihvatljivim* ako za skupove Ω_s i Ω_t vrijedi uvjet prihvatljivosti za neki $\eta > 0$. Intuitivno, to je blok koji je „dovoljno daleko" od dijagonale s obzirom na njegovu veličinu.

Blokove rekurzivno particioniramo koristeći neko stablo grupiranja na sljedeći način. Počevši od korijena stabla $I = \{1, ..., n\}$, provjeravamo je li blok (s, t) prihvatljiv, gdje su s i t čvorovi na istoj razini ili $s = t$. Ako nije prihvatljiv i ako su djeca čvora s s_1 i s_2, a čvora t t_1 i t_2, tada blok (s, t) particioniramo na četiri bloka (s_1, t_1), (s_1, t_2), (s_2, t_1), (s_2, t_2). Provjeravamo dalje za svaki od tih blokova. Ako je blok prihvatljiv, dalje ga ne particioniramo, a u suprotnom ga nastavljamo rekurzivno particionirati. Rekurzija staje kada su čvorovi s i t listovi. Za blokove generirane skupovima indeksa koji su listovi provjerimo jesu li prihvatljivi, no više ne particioniramo blokove koji nisu.

Konstrukcija $\mathcal{H}$ -matrice



Slika 3.5: Particioniranje blokova za različite dubine stabla grupiranja $p = 2, 3, 4$ i konstantu prihvatljivosti $\eta = 1$ (vidi Definiciju 3.2.1). Crvene blokove pohranjujemo kao gusto popunjene matrice, a zelene u reprezentaciji niskog ranga. Izvor: [1].

Hijerarhijske ili $\mathcal{H}$-matrice generalizacija su HODLR matrica. Particija blokova HODLR matrice konstruirana je tako da smo rekurzivno particionirali samo dijagonalne blokove (uključujući i početnu matricu), a vandijagonalne blokove označili smo kao prihvatljive i nismo ih dalje particionirali. Kod $\mathcal{H}$-matrica koristimo općenitiju particiju blokova koja je opisana u prethodnoj točki. Prednosti ovakvog pristupa su veća sloboda u strukturiranju matrice i šira mogućnost primjene. No, konstrukcija same matrice i matričnih algoritama postaje mnogo kompliciranija.

Neka je P proizvoljna particija blokova matrice $A \in \mathbb{R}^{n\times n}$ konstruirana koristeći stablo grupiranja T_I. P dijelimo na uniju dvaju disjunktnih skupova $P = P^+ \cup P^-$. Gdje je

$$P^+ := \{b \in P : b \quad \text{je prihvatljiv}\}, \qquad P^- := P \setminus P^+.$$

Svi blokovi $b \in P^+$ pohranjeni su u reprezentaciji niskog ranga. Dakle, ako je blok $b = (s, t)$, a koji je određen skupom indeksa redaka s i skupom indeksa stupaca t , prihvatljiv, pohranjujemo ga u obliku:

$$A(b) = U_b V_b^T, \qquad U_b \in \mathbb{R}^{|s|\times k_b}, \quad V_b \in \mathbb{R}^{|t|\times k_b},$$

gdje je k_b rang bloka b. Pretpostavimo $k_b \leq k$, za neki $k \in \mathbb{N}$. Blokove iz P^- pohranjujemo kao guste matrice.

Izračunajmo prostornu složenost *stor* ovakve strukture. Za prikaz prihvatljivog bloka $b = (s, t)$ potrebno nam je $k_b(|s| + |t|)$ jedinica prostorne složenosti, gdje je k_b rang tog bloka. Za prikaz bloka (w, z) koji nije prihvatljiv i koji je određen skupom indeksa redaka w i stupaca z potrebno nam je $|w| \cdot |z|$ jedinica prostorne složenosti. Tada je prostorna složenost $\mathcal{H}$-matrice dana s:

$$stor = \sum_{b=(s,t)\in P^+} k_b(|s| + |t|) + \sum_{(w,z)\in P^-} |w| \cdot |z|.$$

Ako za sve neprihvatljive blokove $(w, z) \in P^-$ pretpostavimo $\min\{|w|, |z|\} \leq n_{\min}$, za $n_{\min} \in \mathbb{N}$, tada vrijedi ocjena $|w| \cdot |z| \leq n_{min}(|w| + |z|)$. Dakle, za prostornu složenost *stor* vrijedi:

$$stor \leq \max\{n_{min}, k\} \sum_{(s,t)\in P} (|s| + |t|), \tag{3.10}$$

gdje je k maksimum rangova svih prihvatljivih blokova, kako je i navedeno.

Sada ćemo zbrojiti ukupan broj blokova i ocijeniti prostornu složenost svakog bloka s obzirom na njegovu veličinu. Za svaki čvor s stabla T_I broj svih blokova kojima je s skup indeksa redaka jednak je $card\{t \in T_I : (s, t) \in P\}$, a za svaki čvor t je $card\{s \in T_I : (s, t) \in P\}$ broj svih blokova kojima je t skup indeksa stupaca. Neka je:

$$C_{sp}(P) := \max\{\max_{s\in T_I}(card(\{t \in T_I : (s, t) \in P\}), \max_{t\in T_I}(card(\{s \in T_I : (s, t) \in P\})\}.$$

Ova vrijednost naziva se *konstanta rijetkosti* i označava koliko se puta pribrojnik $|s|$ ili $|t|$ najviše može pojaviti u sumi na desnoj strani nejednakosti (3.10).
Vrijedi:

$$\sum_{(s,t)\in P} |s| \leq C_{sp}(P) \sum_{s\in T_I} |s|,$$

$$\sum_{(s,t)\in P} |t| \leq C_{sp}(P) \sum_{t\in T_I} |t|$$

Također vrijedi ocjena:

$$\sum_{s\in T_I} |s| \leq n(1 + h(T_I)),$$

gdje je $h(T_I)$ visina stabla T_I. Ova ocjena vrijedi jer je ukupan broj elemenata skupova na svakoj od razina stabla uvijek n. Ako pretpostavimo $n_{\min} \leq k$, tada vrijedi ukupna ocjena prostorne složenosti $\mathcal{H}$-matrice:

$$stor \leq 2C_{sp}(P)kn(1 + h(T_I)).$$

Za ograničeni $C_{sp}(P)$ i $h(T_I) \in O(\log n)$ vrijedi ocjena $stor \in O(kn \log n)$ kao za HODLR matrice.

Operacije definirane nad HODLR matricama mogu se proširiti i za $\mathcal{H}$-matrice, pa ih ovdje nećemo posebno opisivati. Detaljno su opisane u [5].

3.3 HSS matrice

HSS (*Hiearchically Semi-Separable*) matrice koriste istu particiju blokova kao HODLR matrice, ali od njih se razlikuju po tome što se vandijagonalni blokovi ne pohranjuju nezavisno jedni od drugih nego su reprezentacije niskog ranga na svakoj razini l ugniježđene. Ovim postupkom smanjuje se i količina memorije potrebne za pohranu matrica. Opišimo konstrukciju HSS matrica.

Neka je vandijagonalni blok u matrici $A \in \mathbb{R}^{n\times n}$ na razini $0 < l < p$ pohranjen u obliku

$$A(I_i^l, I_j^l) = U_i^{(l)} S_{i,j}^{(l)} (V_j^{(l)})^T, \tag{3.11}$$

gdje je $S_{i,j}^{(l)} \in \mathbb{R}^{k\times k}$, a ostale su oznake iste kao za HODLR format (vidi (3.1)). Primijetimo, indekse blokova niskog ranga biramo kao kod HODLR matrica. Na razini l+1 skup indeksa redaka dijelimo na $I_i^l = I_{2i-1}^{l+1} \cup I_{2i}^{l+1}$, a stupaca $I_j^l = I_{2j-1}^{l+1} \cup I_{2j}^{l+1}$.

Važan je uvjet za HSS format egzistencija matrica $X_i^{(l)} \in \mathbb{R}^{2k\times k}$ i $Y_j^{(l)} \in \mathbb{R}^{2k\times k}$ takvih da vrijedi:

$$U_i^{(l)} = \begin{bmatrix} U_{2i-1}^{(l+1)} & 0 \\ 0 & U_{2i}^{(l+1)} \end{bmatrix} X_i^{(l)}, \qquad V_j^{(l)} = \begin{bmatrix} V_{2j-1}^{(l+1)} & 0 \\ 0 & V_{2j}^{(l+1)} \end{bmatrix} Y_i^{(l)}. \tag{3.12}$$

Uvjet (3.12) dopušta nam da faktore $U_i^{(l)}$ i $V_j^{(l)}$, $l = 1, ..., p-1$ rekurzivno dohvatimo iz *baza* $U_i^{(p)}$ i $V_j^{(p)}$. Dakle, za reprezentaciju matrice A moramo pohraniti: dijagonalne blokove na razini p, baze $U_i^{(p)}$ i $V_j^{(p)}$, faktore $S_{i,j}^{(l)}$, $S_{j,i}^{(l)}$ i $X_i^{(l)}$, $Y_j^{(l)}$, $l = 1, ..., p-1$, i ostale blokove koje spremamo kao gusto popunjene matrice. Sada možemo analizirati prostornu složenost ovakve strukture.

Radi jednostavnosti računanja pretpostavimo $n = 2^p n_{\min}$. Neka je $n_l = 2^{p-l} n_{\min}$ veličina bloka na razini l. Na razinama gdje vrijedi $n_l \leq k$ vandijagonalne blokove pohranjujemo kao $n_l \times n_l$ matrice, tj. ne pohranjujemo ih kao (3.11). Neka je $l_c := \lfloor p - \log_2(k/n_{\min}) \rfloor$ najmanja razina na kojoj to vrijedi.

Prvo izračunajmo potrebnu pohranu za $X_i^{(l)}$, $Y_j^{(l)}$ na svakoj razini $l < l_c$. Za svaki l potrebno nam je po 2^{l-1} matrica $X_i^{(l)}$ i $Y_j^{(l)}$. Imamo:

$$2 \sum_{l=1}^{l_c-1} 2^{l-1} 2k^2 \leq 4k^2 2^{l_c} \leq 8k^2 2^p \cdot n_{\min}/k = 8nk.$$

Na sličan način pokazuje se da nam je potrebno $O(nk)$ prostora za pohranu matrica $S^{(l)}$, $l = 1, ..., l_c - 1$. Za pohranu vandijagonalnih blokova na razinama $l = l_c, ..., p$ vrijedi sljedeće:

$$\sum_{l=l_c}^{p} 2^l n_l^2 = 2^p n_{\min}^2 + \sum_{l=l_c}^{p-1} 2^{2p-l} n_{\min}^2 \leq nk + 2^{2p} n_{\min}^2 \sum_{l=l_c}^{p-1} 2^{-l} \leq nk + 2 \cdot 2^{2p} n_{\min}^2 2^{-l_c}$$

$$\leq nk + 2 \cdot 2^{2p} n_{\min}^2 2^{-p} k/n_{\min} = nk + 2 \cdot 2^p n_{\min} = nk + 2n \leq 3nk.$$

Pohrana dijagonalnih blokova na razini p dana je zadnjim pribrojnikom ove sume. Dakle, kada sve zbrojimo, prostorna složenost HSS matrica je $O(nk)$, što je za logaritamski faktor bolje od HODLR matrica.

Efikasni algoritmi nad HSS matricama opisani su u [12]. Na primjer, LU faktorizacija HSS matrice složenosti je $O(n)$.

3.4 $\mathcal{H}^2$-matrice

Ukratko ćemo opisati strukturu $\mathcal{H}^2$-matrica. One su na neki način kombinacija $\mathcal{H}$ i HSS matrica.

Neka je particija blokova P definirana kao za $\mathcal{H}$-matrice. Svaki prihvatljivi blok $b = (s, t) \in P^+$ pohranjujemo u reprezentaciji niskog ranga na sljedeći način:

$$A(b) = U_s S_b V_t^T, \qquad U_s \in \mathbb{R}^{s \times k_s}, \quad V_t \in \mathbb{R}^{t \times K_t}, \quad S_b \in \mathbb{R}^{k_s \times K_t}. \tag{3.13}$$

U_s je baza za skup indeksa redaka s, a V_t za skup indeksa stupaca t. $\mathcal{H}$-matrice sa svojstvom (3.13) nazivamo *uniformne* $\mathcal{H}$-matrice.

Uvedimo još neke restrikcije na matrice U_s i V_t. Prema [1], familiju matrica $(U_s)_{s\in T_I}$ nazivamo *ugniježđenom* ako za svaki $s \in T_I$, koji nije list, sa skupom djece s' postoji matrica $X_{s,s'} \in \mathbb{R}^{k'_s \times k_s}$ takva da:

$$U_s(s', :) = U_{s'} X_{s,s'}.$$

Analogna definicija vrijedi i za $(V_t)_{t\in T_I}$.
Uniformne $\mathcal{H}$-matrice s ugniježđenim familijama $(U_s)_{s\in T_I}$ i $(V_t)_{t\in T_I}$ nazivamo $\mathcal{H}^2$-matrice.

Prostorna složenost $\mathcal{H}^2$-matrice je $O(nk)$, gdje je k maksimalni rang prihvatljivih blokova. Analiza prostorne složenosti nalazi se u [1].

3.5 Implementacija HODLR matrica u Pythonu

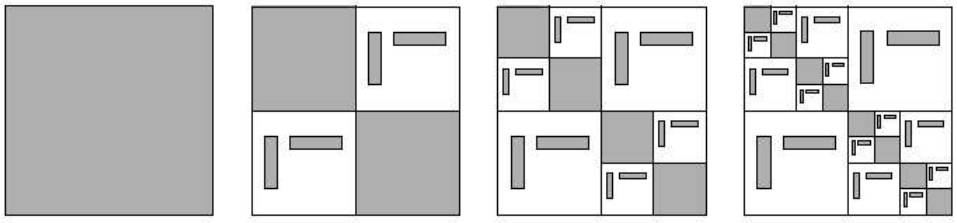
Na poveznici `https://github.com/lumarij/hodlr` nalazi se implementacija strukture kvadratnih HODLR matrica u Pythonu napravljena za potrebe ovog diplomskog rada. Implementirana je po uzoru na već postojeću biblioteku u Matlabu opisanu u [11], a matrične operacije i algoritmi su implementirani kako je opisano u ovom poglavlju. Gdje nam prostor dopusti, prikazat ćemo i neke dijelove koda.

Klasa HODLR

HODLR matrica implementirana je kao klasa sa sljedećim varijablama:

- A11 i A22 instance su HODLR klase i predstavljaju dijagonalne blokove,
- U12 i V12 faktori su vandijagonalnog bloka A_{12},
- U21 i V21 faktori su vandijagonalnog bloka A_{21},
- F je (gusta) matrica pohranjena na posljednjoj razini rekurzije.

Napomena 3.5.1. *Iako smo dosad pohranu vandijagonalnih blokova prikazivali u obliku UV^T, u ovoj implementaciji su prikazani kao produkti $U12 \cdot V12$ i $U21 \cdot V21$, odnosno ti faktori reprezentiraju vandijagonalni blok. SVD u Pythonu vraća vrijednosti U, S, V^T pa će varijable V12 i V21 već reprezentirati V^T .*



Slika 3.6: Prikaz pohrane HODLR matrica u računalu za različite dubine stabla grupiranja ($p = 1, 2, 3, 4$ redom), a koja je određena veličinom minimalnog bloka. Sivi dijagonalni blokovi guste su matrice koje predstavlja varijabla F. Izvor: [11].

Konstruktori HODLR matrica

Aproksimaciju niskog ranga vandijagonalnih blokova HODLR matrice računamo preko SVD-a, Lanczosovog algoritma ili Randomized algoritma opisanih u Poglavlju 2. Kriterij odabira algoritma je gustoća i veličina matrice. Postupak rekurzivno ponavljamo do razine gdje je veličina bloka određena varijablom block_min. Slijedi konstruktor za gusto popunjene matrice.

```
import numpy as np

def __init__(self, A, type=None):
        self.sz  = len(A)
        if self.sz > block_min:
            self.F = []
            tmp = self.sz//2
            self.U12, self.V12 = compress_matrix_svd(A[:tmp, tmp:])
            self.U21, self.V21 = compress_matrix_svd(A[tmp:,:tmp])
            self.A11 = HODLR(A[:tmp,:tmp], type)
            self.A22 = HODLR(A[tmp:,tmp:], type)
        else:
            self.F = A
            self.A11 = []
            self.A22 = []
            self.U12 = []
            self.U21 = []
            self.V12 = []
            self.V21 = []
```

Kod velikih i rijetko popunjenih matrica koristimo posebni konstruktor jer je matrica zadana u scipy . sparse formatu, gdje su podržane drugačije operacije nego kod gusto popunjenih (*dense*) matrica. Biramo želimo li koristiti Lanczosov ili Randomizirani algoritam unosom varijable type. Za blokove manjih dimenzija koristimo SVD, a na dnu rekurzije dijagonalne blokove pohranjujemo kao gusto popunjene matrice.

Ako je matrica posebne strukture koja dopušta da faktore vandijagonalnih blokova pročitamo bez korištenja algoritama za aproksimaciju niskog ranga, tada koristimo posebne konstruktore ovisno o tipu matrice. U nastavku ćemo prikazati faktorizaciju vandijagonalnih blokova jer je za dijagonalne blokove isti postupak.

Tridijagonalne matrice
Vandijagonalni blokovi tridijagonalnih matrica ranga su 1 i njihovi faktori mogu se odmah iščitati iz strukture bloka.

```
self.sz  = len(A)
if type == "tridiagonal":
    self.U12 = np.block([[np.zeros((tmp - 1, 1))], [A[tmp-1, tmp]]])
    self.V12 = np.block([[1, np.zeros(self.sz-tmp-1)]])
    self.U21 = np.block([[A[tmp, tmp-1]],
                         [np.zeros((self.sz-tmp-1, 1))]])
    self.V21 = np.block([[np.zeros(tmp - 1), 1]])
```

Dijagonalne i nul-matrice
Vandijagonalni blokovi takvih su matrica nul-matrice, pa su očito ranga 0. Faktore bloka dimenzije $m \times n$ pohranjujemo kao matrice dimenzija $m \times 0$ i $0 \times n$. Umnožak takve dvije matrice u Pythonu je nul-matrica dimenzija $m \times n$.

```
self.sz  = len(A)
if type == "diagonal" or type == "zeros":
    self.sz  = len(A)
    self.U12 = np.zeros((tmp,0))
    self.V12 = np.zeros((0, self.sz-tmp))
    self.U21 = np.zeros((self.sz-tmp, 0))
    self.V21 = np.zeros((0, tmp))
```

Matrica jedinica
Matrica koja se sastoji samo od jedinica ranga je 1, pa su takvi i njezini vandijagonalni blokovi. Njih se vrlo jednostavno može faktorizirati.

```
self.sz  = len(A)
if type == "ones":
    self.U12 = np.ones((tmp,1))
```

```
self.V12 = np.ones((1, self.sz-tmp))
self.U21 = np.ones((self.sz-tmp, 1))
self.V21 = np.ones((1, tmp))
```

Vrpčaste matrice
Vrpčaste matrice zadane su s gornjom i donjom granicom bandu i bandl koje Python određuje s pomoću funkcije bandwidth. Da budemo u skladu s definicijom u Poglavlju 1, pretpostavimo bandu = bandl. Postupak je sličan faktoriziranju vandijagonalnog bloka tridijagonalne matrice. Rang vandijagonalnih blokova vrpčaste matrice jednak je bandu.

Metode HODLR klase

is_leafnode
Vraća True ako smo došli do dna stabla grupiranja, tj. do posljednje razine rekurzije. U većini će se funkcija pozivati za provjeru jesmo li na dnu rekurzije ili ne.

```
def is_leafnode(self):
    return False if self.A11 else True
```

full
Za zadanu HODLR matricu vraća njezin originalni (gusti) format množenjem faktora vandijagonalnih blokova na svakoj razini rekurzije i rekurzivno ih pohranjujući.

```
def full(self):
     if self.is_leafnode():
       return self.F
     else:
        a11 = self.A11.full()
        a12 = np.matmul(self.U12, self.V12)
        a21 = np.matmul(self.U21, self.V21)
        a22 = self.A22.full()
        sz = self.A11.sz
        A = np.zeros((self.A11.sz + self.A22.sz,
                              self.A11.sz + self.A22.sz ))
        A[:sz, :sz] = a11
        A[:sz, sz:] = a12
        A[sz:, :sz] = a21
        A[sz:, sz:] = a22
        return A
```

hodlr_rank
Metoda vraća najveći rang vandijagonalnog bloka provjeravajući na svakoj razini rekurzije broj stupaca faktora vandijagonalnog bloka U12 i U21 odnosno broj redaka V12 i V21.

```
def hodlr_rank(self):
    if self.is_leafnode():
      return 0
    else:
      return max(self.A11.hodlr_rank(),
            self.A22.hodlr_rank(), len(self.V12), len(self.V21))
```

hodlr_depth
Metoda vraća dubinu stabla grupiranja zadane HODLR matrice.

```
def hodlr_depth(self):
      if self.is_leafnode():
         return 1
      else:
         return 1 + self.A11.hodlr_depth()
```

storage_complexity
Metoda vraća ukupnu prostornu složenost zadane HODLR matrice. Zbog njegove veličine kod ovdje neće biti prikazan.

Implementacija matričnih operacija i algoritama

Množenje matrice i vektora
Funkcija prima HODLR matricu i vektor i vraća vektor koji je njihov umnožak. Postupak prati opis algoritma množenja matrice i vektora iz ovog poglavlja. Zbog veličine kod ovdje neće biti prikazan.

Zbrajanje HODLR matrica
Funkcija hodlr_plus prima dvije HODLR matrice i vraća njihov zbroj u HODLR formatu. Varijablu H pretvaramo u HODLR objekt uz pomoć funkcije copy.deepcopy zbog toga što ćemo tako očuvati vrijednost H1. Funkcija compress_factors smanjuje dimenziju faktora na način kako je to opisano u cjelini 2.2.

```
def hodlr_plus(H1, H2):
   H = copy.deepcopy(H1)
    if H1.is_leafnode():
       H.F = np.add(H1.F, H2.F)
       return H
    else:
       H.A11 = hodlr_plus(H1.A11, H2.A11)
       H.A22 = hodlr_plus(H1.A22, H2.A22)
       H.U12 = np.block([H1.U12, H2.U12])
```

```
    H.U21 = np.block([H1.U21, H2.U21])
    H.V12 = np.block([[H1.V12], [H2.V12]])
    H.V21 = np.block([[H1.V21],[H2.V21]])
    H.U12, H.V12 = compress_factors(H.U12,np.transpose(H.V12))
    H.U21, H.V21 = compress_factors(H.U21, np.transpose(H.V21))
    return H
```

Množenje HODLR matrica

Na (3.3) i Slici 3.3 vidimo da za svaki blok matrice A moramo posebno implementirati množenje zbog različitih vrsta množenja i zbrajanja koji se vrše da bi se pojedini blok izračunao. Prikazat ćemo kod za računanje bloka A_{11} i bloka A_{12} jer se ostala dva računaju analogno.

Blok A_{11}

```
def hodlr_matrix_multiplication(H1, H2):
    H = copy.deepcopy(H1)
    if H1.is_leafnode():
        H.F = np.matmul(H1.F, H2.F)
    else:
        H.A11 = hodlr_matrix_multiplication(H1.A11, H2.A11)
        U = np.matmul(H1.V12, H2.U21)
        U = np.matmul(H1.U12, U)
        H.A11 = hodlr_rank_update(H.A11, U, H2.V21)
```

Funkcija hodlr_rank_update (H,U,V) prima HODLR matricu H i dvije matrice niskog ranga U i V te vraća H+U*V primjenjujući kompresiju faktora pri zbrajanju vandijagonalnih blokova na svakoj razini rekurzije opisanu u cjelini 2.2.

Blok A_{12}

```
H.U12 = np.block([hodlr_times_dense(H1.A11, H2.U12 ), H1.U12] )
H.V12 = np.block([[H2.V12], [dense_times_hodlr(H1.V12, H2.A22)]])
```

Funkcija hodlr_times_dense (H, U) prima HODLR matricu H i (gustu) matricu niskog ranga U i vraća njihov umnožak u gustom formatu. Postupak je sličan kao za množenje HODLR matrice i vektora. Na isti način računamo funkciju dense_times_hodlr (V, H) gdje je V matrica niskog ranga, a H HODLR matrica.

Kada se cijeli algoritam množenja izvrši, preostaje nam još kompresija faktora vandijagonalnih blokova na razini 1.

LU faktorizacija

Funkcija hodlr_lu (H) prima HODLR matricu i vraća gornjetrokutasti i donjetrokutasti faktor u HODLR formatu.

```
def hodlr_lu(H):
    HL = copy.deepcopy(H)
    HU = copy.deepcopy(H)
    if H.is_leafnode():
        HL.F, HU.F = lu(H.F, permute_l=True)
        return HL, HU
    else:
        n = H.sz
        m1 = H.A11.sz
        HL.U12 = np.zeros((m1, 0))
        HL.V12 = np.zeros((0,n-m1))
        HU.U21 = np.zeros((n-m1,0))
        HU.V21 = np.zeros((0, m1))
        HL.A11, HU.A11 = hodlr_lu(H.A11)
        HU.U12 = solve_lower_triangular(HL.A11, H.U12)
        HL.V21 = solve_upper_triangular(HU.A11, H.V21)
        U = copy.deepcopy(H.A22)
        V1 = np.matmul(HL.V21,HU.U12)
        V1 = np.matmul((-1)*HL.U21, V1)
        V2 = HU.V12
        HL.A22, HU.A22 = hodlr_lu(hodlr_rank_update(U,V1,V2))
        return HL,HU
```

Funkcije solve_upper_triangular (HU.A11, H.V21) i solve_lower_triangular (HL.A11, H.U12) rješavaju korake 2 i 3 redom iz opisa približne LU faktorizacije u ovom poglavlju. Za računanje HL.A22 i HU.A22 koristimo funkciju hodlr_rank_update (U, V1, V2) koja rekurzivno računa U+V1*V2, a koju smo koristili i kod množenja matrica.

Implementacija algoritama za aproksimaciju niskog ranga

Za početak, opišimo kako biramo numerički rang vandijagonalnog bloka $A(I_i^l, I_j^l)$ HODLR matrice. Spektralnu normu greške aproksimacije vandijagonalnog bloka ograničavamo s $threshold \cdot \|A(I_i^l, I_j^l)\|_2$, gdje je *threshold* tolerancija greške i po njena pretpostavljena vrijednost je 10^{-12} iako se u kodu može promijeniti u funkciji change_threshold. Time osiguravamo da je spektralna norma greške aproksimacije matrice A ograničena s $O(threshold \cdot \log(n)\|A\|_2)$. Dakle, odbacujemo sve singularne vrijednosti manje od $threshold \cdot \|A(I_i^l, I_j^l)\|_2$.

Numerički rang tada je broj preostalih singularnih vrijednosti i njega koristimo za aproksimaciju niskog ranga.

Aproksimacija niskog ranga koristeći SVD

```
def compress_matrix_svd(A):
    u, s, v = np.linalg.svd(A, full_matrices = True)
    smatrix = np.diagflat(s)
    k = 0
    for i in s:
        if(abs(i) > s[0] * threshold):
            k += 1
    u = np.matmul(u[:,:k], smatrix[:k, :k])
    v = v[:k,:]
    return u, v
```

Funkcija vraća $\mathcal{T}_k(A)$ u obliku dva faktora u i v, gdje je k numerički rang kojeg biramo kako je opisano na početku ove točke.

Kompresija faktora vandijagonalnog bloka

Ovaj algoritam koristi se pri zbrajanju i množenju HODLR matrica. On osigurava da rang vandijagonalnih blokova ne raste prebrzo. Rang aproksimiramo kako smo opisali na početku cjeline. Možemo odrediti i gornju granicu broja stupaca (ili redaka) faktora, ali time se nećemo baviti u ovoj implementaciji. Algoritam je opisan u Cjelini 2.2 .

Lanczosov i Randomizirani algoritam

Algoritmi su opisani u Cjelini 2.2 i kod prati taj opis. Slijedi kod za Randomizirani algoritam.

```
def randomized_algorithm(A, k, p):
    m = A.shape[0]
    n  = A.shape[1]
    O = np.random.normal(size = (n, p+k))
    Y  = A @ O
    q, r = np.linalg.qr(Y, 'reduced')
    Z = np.transpose(q) @ A
    u, s, v = np.linalg.svd(Z)
    rk = 0
    for i in s:
        if(abs(i) > s[0] * threshold):
            rk += 1
```

```
U = np.matmul(q, u[:, :rk])
S = np.diagflat(s)
S = S[:rk, :rk]
V = v[:rk, :]
U = np.matmul(U,S)
return U, V
```

U kodu koristimo operator @ za množenje matrica u scipy.sparse formatu. Na kraju algoritma odbacujemo male singularne vrijednosti da bismo se osigurali od prevelike procjene ranga matrice.

Greška aproksimacije matrice HODLR matricom

Grešku aproksimacije matrice A HODLR matricom H računamo na način da HODLR matricu prvo pretvorimo u full format koristeći metodu full(), a zatim izračunamo spektralnu normu razlike matrice A i H.full(). Nju računa funkcija check_tolerance(A, H.full()).

Poglavlje 4

Primjeri i primjene

4.1 HODLR strukture nekih matrica

U ovoj točki pokazat ćemo primjere matrica čiji vandijagonalni blokovi dopuštaju dobru aproksimaciju niskog ranga. To će posljedično značiti da se takve matrice mogu efikasno pohraniti u HODLR strukturi. To će biti samo mali dio takvih matrica.

Primjer 4.1.1. *Hilbertova matrica $H_n(i, j) = 1/(i + j + 1)$ kanonski je primjer loše uvjetovane matrice. Uvjetovanost joj raste jako brzo u ovisnosti o veličini matrice n. Već smo*

n	$\kappa(H_n)$	n	$\kappa(H_n)$
2	$1.928 \cdot 10^1$	9	$4.932 \cdot 10^{11}$
3	$5.241 \cdot 10^2$	10	$1.603 \cdot 10^{13}$
4	$1.551 \cdot 10^4$	11	$5.231 \cdot 10^{14}$
5	$4.766 \cdot 10^5$	12	$1.713 \cdot 10^{16}$
6	$1.495 \cdot 10^7$	13	$5.628 \cdot 10^{17}$
7	$4.754 \cdot 10^8$	14	$1.853 \cdot 10^{19}$
8	$1.526 \cdot 10^{10}$	15	$6.117 \cdot 10^{20}$

Tablica 4.1: Prikaz rasta uvjetovanosti Hilbertove matrice u ovisnosti o veličini matrice n, za $n \leq 15$. Podaci su preuzeti iz [14].

spomenuli da je numerički rang loše uvjetovane matrice manji od njenog egzaktnog ranga, tako da aproksimacija numeričkim rangom ima smisla. To onda možemo očekivati i za njene vandijagonalne blokove. Iz Tablice 4.2 možemo zaključiti da rang vandijagonalnih blokova neće previše narasti ni za veće dimenzije matrice H_n.

n	k
5	2
10	5
15	7
100	7
500	7
1000	7
2000	7
5000	7
7000	7

Tablica 4.2: HODLR rang k Hilbertove matrice, tj. najveći rang vandijagonalnog bloka na svim razinama, u ovisnosti o n, gdje je rang broj singularnih vrijednosti bloka većih od $10^{-12}\sigma_1$, σ_1 je najveća singularna vrijednost bloka. Odabrali smo $n_{\min} = 1$.

Primjer 4.1.2. *Neka je funkcija* $f : [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \longrightarrow \mathbb{R}$ *definirana na mreži* $\{x_1, ..., x_m\} \times \{y_1, ..., y_n\}$ *tako da vrijedi* $x_{\min} = x_1 \leq x_2 \leq ... \leq x_m = x_{\max}$ *i* $y_{\min} = y_1 \leq y_2 \leq ... \leq y_n = y_{\max}$. *Tada vrijednosti funkcije f možemo rasporediti u matricu* $A(i, j) = f(x_i, y_j)$. *Pretpostavimo da se f može zapisati u obliku* $f(x, y) = g(x)h(y), \quad g : [x_{\min}, x_{\max}] \longrightarrow \mathbb{R}, \quad h : [y_{\min}, y_{\max}] \longrightarrow \mathbb{R}$. *Za takve funkcije kažemo da su* ***separabilne***.
Tada matricu A možemo zapisati u obliku:

$$A = \begin{bmatrix} g(x_1) \\ g(x_2) \\ \vdots \\ g(x_m) \end{bmatrix} \begin{bmatrix} h(y_1) & \dots & h(y_n) \end{bmatrix}.$$

Ovakva matrica očito je ranga 1.

Sada promotrimo slučaj gdje matrični elementi same matrice A nisu vrijednosti separabilne funkcije, ali su takvi elementi njezinih vandijagonalnih blokova. Zgodan primjer (osim tridijagonalne matrice i njezinog inverza ako postoji) je „eksponencijalna" matrica $B_n(i, j) = \exp(\alpha|i - j|/n), \quad \alpha \in \mathbb{R}$. B_n *je regularna i štoviše jako dobro uvjetovana, ali njezini vandijagonalni blokovi su ranga 1. Njih možemo prikazati u obliku* $B_n(i, j) = \exp(\alpha i/n)\exp(-\alpha j/n)$, *za* $i > j$ *i* $B_n(i, j) = \exp(\alpha j/n)\exp(-\alpha i/n)$ *za* $j > i$. *Takva matrica jako je dobar primjer matrice koju možemo prikazati s pomoću HODLR strukture - sama nije niskog ranga, ali njezini vandijagonalni blokovi jesu.*

Generalizacija separabilnih funkcija su ***semi-separabilne*** *funkcije. To su funkcije koje se mogu prikazati kao aproksimacija sume odvojivih funkcija. Dakle, mogu se prikazati u*

obliku:

$$f(x,y) = g_1(x)h_1(y) + g_2(x)h_2(y) + ... + g_k(x)h_k(y) + \epsilon_k(x,y),$$

gdje je $\epsilon_k(x,y)$ greška aproksimacije funkcije f funkcijom $f_k(x,y) = g_1(x)h_1(y)+g_2(x)h_2(y)+ ... + g_k(x)h_k(y)$. Neka je matrica $A(i,j) = f(x_i, y_j)$. Takva matrica ima dobru aproksimaciju matricom ranga k.
Na primjer, takva je funkcija $f_1(x,y) = \sum_{l=1}^{k} \exp(i(x-y)/n) + 10^{-12}$ i matrica $A_1(i,j) = f_1(x_i, y_j)$ može se dobro aproksimirati matricom ranga k.
Funkcija $f_2(x,y) = \sum_{l=1}^{k} \exp(i(|x-y|)/n) + 10^{-12}$ nije semi-separabilna, tako se ni matrica $A_2(i,j) = f_2(x_i, y_i)$ ne može aproksimirati matricom ranga k, ali njeni vandijagonalni blokovi mogu.

Primjer 4.1.3. *Sada ćemo promotriti funkcije čija vrijednost ovisi o udaljenosti između ulaza i neke fiksne točke, npr. ishodišta. Takve funkcije nazivaju se **radijalne bazne funkcije**. U ovom primjeru bavit ćemo se matricama $A \in \mathbb{R}^{n\times n}$ čiji elementi su zadani sa $A(i,j) = \phi(r_{i,j}) = \phi(\|x_i - x_j\|_2)$, gdje je ϕ radijalna bazna funkcija, a x_i i x_j točke na jediničnoj kružnici. Varijablu x_i možemo zapisati kao $x_i = (\cos(\theta_i), \sin(\theta_i)) = \cos(\theta_i) + i\sin(\theta_i)$, gdje je $\theta_i = 2\pi \cdot i/n$, za $i = 0, ..., n-1$. Tada su elementi matrice A zadani sa $A(i,j) = \phi(2|\sin((\theta_i - \theta_j)/2)|)$, $\theta_i, \theta_j \in [0, 2\pi)$.*

Radi jednostavnosti promatrat ćemo numerički rang najvećeg vandijagonalnog bloka matrice A. Pretpostavit ćemo da se singularne vrijednosti unutarnjih vandijagonalnih blokova (kada bismo particionirali dijagonalni blok) matrice A slično ponašaju. Numerički rang određen je brojem singularnih vrijednosti većih od $10^{-12}\sigma_1$.

n	$1+r^2$	$\sqrt{1+r^2}$	$1/(1+r^2)$	$1/(\sqrt{1+r^2})$	$\exp(-r)$	$\exp(-r^2)$	$\log(1+r)$
1024	3	17	20	19	12	19	12
2048	3	17	20	19	12	19	12
4096	3	17	20	19	12	19	12
8192	3	17	20	19	12	19	12

Tablica 4.3: Rang vandijagonalnog bloka matrice $A(i,j) = \phi(\|x_i - x_j\|_2)$ za različite veličine matrice i različite radijalne bazne funkcije. Primjećujemo da se rang vandijagonalnog bloka ne mijenja s veličinom matrice n.

Napomena 4.1.4. *Kada se sama matrica može aproksimirati matricom niskog ranga, njena hijerarhijska (u ovom slučaju HODLR) struktura nam je naizgled nepotrebna. No, kod rješavanja matričnih jednadžbi, npr. koje se nalaze u [10], u iteracijama se gubi prikazivost niskog ranga u obliku faktora s malim brojem stupaca i tada moramo koristiti hijerarhijsku strukturu.*

Primjer 4.1.5. *U drugom poglavlju naveli smo da za aproksimaciju velike i rijetko popunjene matrice koristimo Randomizirani algoritam radi bržeg računanja. U ovom primjeru promatramo slučajnu vrpčastu matricu koja je vrsta strukturirane rijetko popunjene matrice. Iako postoji izravni i jeftiniji način, radi ilustracije za aproksimaciju niskog ranga njezinih vandijagonalnih blokova koristit ćemo Randomizirani algoritam uz $k = band$ i $p = 0$, band je gornja (i donja) granica matrica. Također ćemo pokazati da je najveći rang vandijagonalnih blokova HODLR strukture inverza vrpčaste matrice jednak onom vrpčaste matrice.*

band	k_1	k_2
1	1	1
5	5	5
10	10	10

Tablica 4.4: Najveći rang k_1 vandijagonalnog bloka HODLR strukture vrpčastih matrica reda 1024 kojoj su donja i gornja granica određene varijablom *band* i najveći rang k_2 vandijagonalnih blokova HODLR strukture njezinog inverza uz $n_{\min} = 1$. Elementi vrpčaste matrice su slučajni cijeli brojevi od 1 do 9. Rang matrice određen je brojem singularnih vrijednosti većih od $\sigma_1 10^{-12}$, gdje je σ_1 najveća singularna vrijednost matrice.

band	k_1	k_2
1	1	1
5	5	5
10	10	10

Tablica 4.5: Najveći rang k_1 vandijagonalnog bloka HODLR strukture vrpčastih matrica reda 2048 kojoj su donja i gornja granica određene varijablom *band* i najveći rang k_2 vandijagonalnih blokova HODLR strukture njezinog inverza uz $n_{\min} = 1$. Elementi vrpčaste matrice su slučajni brojevi od 1 do 9. Rang matrice određen je brojem singularnih vrijednosti većih od $\sigma_1 10^{-12}$, gdje je σ_1 najveća singularna vrijednost matrice.

Primjer 4.1.6. *U ovom primjeru bavit ćemo se matricom A_2 iz Primjera 4.1.2 (uz $x_i = i$ i $y_j = j$) i njezinim inverzom. Iako nije rijetko popunjena, vandijagonalne blokove matrice A_2 prvo ćemo aproksimirati s pomoću Randomiziranog algoritma radi ilustracije. Zatim ćemo najveći rang vandijagonalnih blokova usporediti s onim koji dobiven preko SVD-a. Također, izračunat ćemo i najveći rang vandijagonalnih blokova HODLR strukture inverza matrice A_2.*

k	p	r_k
5	0	5
5	2	5
5	5	5
10	0	6
10	2	6
10	5	6

Tablica 4.6: Najveći rang r_k vandijagonalnog bloka HODLR strukture matrice A_2 iz Primjera 4.1.2 reda 512 u ovisnosti o varijabli k koja označava broj pribrojnika u funkciji kojom je generirana matrica A_2 kao i rang kojim želimo aproksimirati vandijagonalni blok i parametru uzorkovanosti p. Rang bloka određen je brojem singularnih vrijednosti većim od $\sigma_1 10^{-12}$ i vrijedi $n_{\min} = 1$. Velike vandijagonalne blokove aproksimiramo randomiziranim algoritmom.

k	r_k	r_{inv}
5	5	187
10	6	206

Tablica 4.7: Najveći rang r_k vandijagonalnih blokova HODLR strukture matrice A_2 iz Primjera 4.1.2 reda 512. Vrijedi sve isto kao u Tablici 4.6 samo što za aproksimaciju niskog ranga vandijagonalnih blokova koristimo SVD. r_{inv} označava najveći rang vandijagonalnih blokova HODLR strukture inverza matrice A_2. Zaključujemo da, iako se matrica A_2 može efikasno pohraniti u HODLR strukturi, za njezin inverz ne vrijedi isto.

4.2 Primjene hijerarhijskih matrica

Osim već nabrojenih primjena, poput omogućavanja manje vremenske složenosti pri matričnim operacijama, LU faktorizaciji i najvažnije, rješavanja linearnih sustava, neke primjene izlaze iz okvira ovog rada. Nabrojat ćemo ih, a reference na neke od njih nalaze se u [1]:

- rješavanje matričnih jednadžbi (vidi [9], [10]),
- diskretizacija integralnih i diferencijalnih jednadžbi,
- rješavanje linearnih sustava proizašlih iz prekondicioniranja metode konačnih elemenata bazirano na $\mathcal{H}$-matricama,

- hijerarhijski format kombiniran s direktnom faktorizacijom rijetko popunjenih matrica,
- računanje svojstvenih vrijednosti,
- aproksimacija jezgre u strojnom učenju,
- klasteriranje u grafovima korištenjem aproksimacija niskog ranga.

Ovo je samo mali dio primjene hijerarhijskih struktura, a u budućnosti će primjena biti još šira.

Bibliografija

[1] J. Ballani i D. Kressner, *Matrices with Hierarchical Low-Rank Structures*, Lecture Notes in Math, sv. 2173, pogl. In Exploiting hidden structure in matrix computations: algorithms and applications, str. 161–209, siječanj 2016.

[2] F. Belić, D. Ševerdija i Ž. Hocenski, *Naivno množenje matrica nasuprot Strassenovog algoritma u više-nitnom okruženju*, Tehnički vjesnik, 18 (3), 309-314. (2011), https://hrcak.srce.hr/71808.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest i C. Stein, *Introduction to Algorithms, 3rd Edition*, The MIT Press, 2009.

[4] T. Došlić, *Numerička matematika*, Građevinski fakultet, Sveučilište u Zagrebu, https://www.grad.unizg.hr/_download/repository/Doslic_T.%3B_Numericka_matematika.pdf.

[5] W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis*, sv. 49, Springer Berlin, Heidelberg, prosinac 2015.

[6] N. Halko, P. G. Martinsson i J. A. Tropp, *Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions*, SIAM Review **53** (2011), br. 2, 217–288.

[7] M. T. Heath, *Scientific Computing: An Introductory Survey, Revised Second Edition*, SIAM-Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2018.

[8] J.Augustinović, *Uvjetovanost linearnog sustava*, Osječki matematički list, 9 (2), 93-105 (2009), https://hrcak.srce.hr/4834.

[9] Daniel Kressner, Stefano Massei i Leonardo Robol, *Low-Rank Updates and a Divide-And-Conquer Method for Linear Matrix Equations*, SIAM Journal on Scientific Computing **41** (2019), br. 2, A848–A876.

[10] S. Massei, D. Palitta i L. Robol, *Solving Rank-Structured Sylvester and Lyapunov Equations*, SIAM Journal on Matrix Analysis and Applications **39** (2018), br. 4, 1564–1590.

[11] S. Massei, L. Robol i D. Kressner, *hm-toolbox: Matlab software for HODLR and HSS matrices*, 2019, `https://arxiv.org/abs/1909.07909`, posjećeno u rujnu 2022.

[12] Z. Sheng, P. Dewilde i S. Chandrasekaran, *System Theory, the Schur Algorithm and Multidimensional Analysis*, sv. 176, pogl. Algorithms to Solve Hierarchically Semi-separable Systems, str. 255–294, Birkhäuser Basel, Basel, 2007.

[13] Horst Simon, *Low Rank Matrix Approximation Using The Lanczos Bidiagonalization Process With Applications*, SIAM Journal on Scientific Computing **21** (1997), 2257–2274.

[14] S.Singer, *Numerička matematika, predavanja*, Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu.

[15] E. Thomé, *A modified block Lanczos algorithm with fewer vectors*, 2016, `https://arxiv.org/abs/1604.02277`, posjećeno u rujnu 2022.

[16] N. Truhar, *Numerička linearna algebra*, Odjel za matematiku, Sveučilište J. J. Strossmayera u Osijeku, 2010.

Sažetak

U ovom radu bavili smo se aproksimacijama niskog ranga i najpoznatijim algoritmima za iste. Temelj za ove algoritme je dekompozicija na singularne vrijednosti (SVD). No, glavna tema ovog rada je pregled najpoznatijih hijerarhijskih formata matrice (HODLR, $\mathcal{H}$, HSS, $\mathcal{H}^2$) i operacija nad njima. Pokazalo se da uz određene uvjete, poput malog (numeričkog) ranga vandijagonalnih blokova matrice, matrice možemo pohraniti u mnogo manje od $O(n^2)$ prostornih jedinica, a matrične operacije se izvršavaju u skoro linearnom vremenu, što je posebno korisno za matrice velikih dimenzija kakve se najčešće javljaju u praksi. Najveći naglasak stavljen je na HODLR strukturu matrice jer smo istu implementirali u programskom jeziku Python. Na kraju rada dani su neki primjeri matrica koje se mogu pohraniti korištenjem HODLR strukture i dan je pregled najvažnijih primjena hijerarhijskih formata matrica.

Summary

In this thesis, we dealt with low-rank approximation of matrices and well known algorithms for such purpose. The basis for these algorithms is singular value decomposition (SVD). The main topic of this thesis is an overview of hierarchical matrices (HODLR, $\mathcal{H}$, HSS, $\mathcal{H}^2$) and operations on them. It has been shown that under certain conditions, such as a small (numerical) rank of off-diagonal matrix blocks, matrices can be stored in less than $O(n^2)$ storage complexity and matrix operations are executed in almost linear time. This is especially useful for matrices of large dimension, such as those that are most often encountered in practice. The greatest emphasis was put on HODLR matrices because we implemented them in the Python programming language. At the end of the thesis, a number of examples of matrices that can be well stored in HODLR structure and an overview of the most important applications of hierarchical matrix formats is given.

Životopis

Rođena sam 21.3.1997. u Dubrovniku. Nakon završene Osnovne škole Župa dubrovačka, 2011. upisujem Biskupijsku klasičnu gimnaziju Ruđera Boškovića s pravom javnosti u Dubrovniku. Godine 2015. upisujem Preddiplomski sveučilišni studij Matematika na Prirodoslovno-matematičkom fakultetu u Zagrebu, a 2018. Diplomski sveučilišni studij Računarstvo i matematika na istom fakultetu. Paralelno pohađam Glazbenu školu Karlovac, smjer solo pjevanje.