Matrični račun u Pythonu

Jazvec, Maja

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:199639

Rights / Prava: Attribution-NonCommercial-NoDerivs 3.0 Unported/Imenovanje-Nekomercijalno-Bez prerada 3.0

Download date / Datum preuzimanja: **2024-03-29**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Maja Jazvec

MATRIČNI RAČUN U PYTHONU

ZAVRŠNI RAD

Varaždin, 2020.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Maja Jazvec

Matični broj: 44733/16–R

Studij: Primjena informacijske tehnologije u poslovanju

MATRIČNI RAČUN U PYTHONU

ZAVRŠNI RAD

Mentor :

Izv. prof. dr. sc. Zlatko Erjavec

Varaždin, kolovoz 2020.

Maja Jazvec

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristila drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Predmet ovog rada je matrični račun implementiran u programski jezik Python. U prvom dijelu objašnjen je pojam matrice te je dan prikaz osnovnih matričnih operacija i svojstava matrica. Također, uvedeni su pojmovi determinante i inverzne matrice kao dva bitna koncepta za rješavanje matričnih jednadžbi. U drugom dijelu rada obrađuje se Python modul NumPy namjenjen za efikasno računanje s matricama. U njemu se na konkretnim primjerima provode matrične operacije; zbrajanje i oduzimanje, množenje skalarom i množenje matrica. NumPy sadrži modul *np.linalg* koji nudi niz rutina za računanje iz područja linearne algebre. Pomoću modula izračunata je determinanta, rang, trag i inverzna matrica. Također, prikazana je funkcija za rješavanje matrične jednadžbe. U ovom radu želi se prikazati snaga Python NumPy biblioteke u radu s matricama te upoznati čitatelja sa funkcionalnostima iste.

Ključne riječi: matrica; determinanta; matrične jednadžbe; Python; NumPy

Sadržaj

1. Uvod 1

2. Metode i tehnike rada 2

3. Definicija i primjeri matrica 3

4. Operacije s matricama 7
4.1. Zbrajanje matrica 7
4.2. Oduzimanje matrica 7
4.3. Množenje matrice skalarom 8
4.4. Množenje matrica 8
4.5. Determinante 10
4.6. Inverzna matrica 13

5. Python 15
5.1. NumPy 16
5.1.1. Stvaranje matrica 17
5.1.2. Indeksiranje u matricama 21
5.1.3. Zbrajanje, oduzimanje i množenje matrice skalarom 22
5.1.4. *Broadcasting* 23
5.1.5. Množenje matrica 23
5.1.6. NumPy modul za linearnu algebru 25
5.1.7. Jednadžba $\mathbf{AX} + \mathbf{XB} = \mathbf{C}$ 27

6. Zaključak 29

Popis literature 31

Popis slika 32

1. Uvod

Predmet ovog rada je matrični račun implementiran u programski jezik Python. Matrice su matematički alat s mnogobrojnim primjenama. Mogu se koristiti u bilo kakvom problemu gdje se podaci, na određeni način, mogu prikazati kao višedimenzionalni objekti. Matrice imaju dugu povijest primjene u rješavanju linearnih jednadžbi te provjeru postojanja rješenja jednadžbe računanjem determinante.

Neka od područja primjene matrica su:

- računalna grafika;
- strojno učenje;
- statistika;
- fizika;
- elektrotehnika (rješavanje jednadžbi strujnih krugova);
- kriptografija (Hill šifra).

Za efikasniji rad s matricama koriste se različiti programski alati. U ovom radu korišten je programski jezik Python i biblioteka NumPy. Python modul NumPy jedan je od najpoznatijih *open-source* programskih alata za jednostavan i brz rad s matricama. Cilj ovog rada je objasniti samu logiku iza osnovnih matričnih operacija te prikazati snagu i efikasnost NumPy biblioteke u računanju s matricama.

Struktura rada sastoji se od šest poglavlja, od kojih se prvo odnosi na uvod, a šesto na zaključak. Nakon uvoda, u drugom poglavlju su ukratko opisane metode i tehnike korištene pri razradi teme te su navedeni korišteni programski alati. Treće poglavlje sadrži definiciju i objašnjenje matrice i navodi neke specifične matrice i njihove karakteristike. U četvrtom poglavlju su definirane računske operacije s matricama; zbrajanje i oduzimanje matrica, množenje matrica i množenje matrica brojem. Svaka matrična operacija prikazana je na primjeru. Također, uveden je pojam determinante i inverzne matrice te prikazan njihov izračun.

Peto poglavlje započinje uvodom u Python i biblioteku NumPy. Opisani su temljeni objekti biblioteke te su prikazane različite funkcije za kreiranje matrica. Zatim su, na primjerima, opisane metode i funkcije koje se koriste za matrične operacije. Objašnjen je i *broadcasting* kao bitan koncept za razumijevanje aritmetičkih operacija korištenjem NumPy biblioteke. Na kraju poglavlja, prikazan je NumPy modul za linearnu algebru. Navedene su funkcije za izračun determinante, ranga, traga i inverza matrice te je prikazan postupak rješavanja sustava linearnih i matričnih jednadžbi.

2. Metode i tehnike rada

Razradi prvog, teorijskog, dijela rada prethodilo je proučavanje raspoložive građe iz područja linearne algebre, specifično matrica. Matrični račun, kao cjelina, rasčlanjen je na jednostavnije djelove. Prema tome, kreće se od samog pojma matrice. Na taj način opisane su sve karakteristike matrica bitne za razumijevanje računskih operacija. Matrične operacije su prikazane na primjerima, prilikom čega je analizirano ponašanje matrica u pojedinoj operaciji. Na temelju proučavanja djelova doneseni su zaključci o cjelini.

U drugom dijelu rada, korišteni su relevantni internetski izvori i znanstveni članci na temu Pythona, NumPy biblioteke te općenito znanstvenog računanja u Pythonu. Za praktični dio rada, preuzet je Python, verzije 3.7.2. te Python modul NumPy. Za rad s Pythonom i izradu primjera, korišteno je integrirano razvojno okruženje Visual Studio Code. Sve NumPy funkcije i metode isprobane su na konkretnim primjerima da bi se moglo analizirati ponašanje funkcija, razlike između pojedinih funkcija te koje se greške javljaju i kada. Na temelju toga doneseni su i zaključci.

Rad je pisan pomoću LaTexa, programskoj jezika za stvaranje dokumenata korištenjem uređivača Tex. LaTex omogućuje jednostavniji prikaz matrica, jednadžba te formula. Preuzeta je MikTex distribucija za operacijski sustav Windows, a kao editor korišten je TeXstudio.

3. Definicija i primjeri matrica

Elezović [7] navodi da je matrica pravokutna tablica ispunjena njezinim elementima. Osim realnih brojeva, elementi matrice mogu biti kompleksni brojevi, ali i drugi objekti poput funkcija, vektora, pa čak i samih matrica. U ovom radu promatraju se samo matrice realnih brojeva.

Strože matematički matricu definiramo; Neka je $\mathbb{R}$ skup realnih brojeva te $m, n \in \mathbb{N}$.

Svako preslikavanje

$$A : \{1, 2, \ldots, m\} \times \{1, 2, \ldots, n\} \to \mathbb{R}$$

naziva se **matrica tipa** (m, n) s koeficijentima (elementima) iz skupa $\mathbb{R}$. [1]

Djelovanje svakog takvog preslikavanja se zapisuje kao tablica od m redaka i n stupaca:

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{13} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}$$

gdje je a_{ij} vrijednost funkcije A u paru (i, j). Skup svih matrica s m redaka i n stupaca s elementima iz skupa $\mathbb{R}$ označavamo s $M_{mn}(\mathbb{R})$ [1]

Elementi matrice su brojevi $a_{ij},\ i = 1, \ldots, m,\ j = 1, \ldots, n,$. Prvi (i) označava broj retka, a drugi (j) broj stupca u kojem se element nalazi.

Npr., matrica

$$B = \begin{bmatrix} 6 & 0 & -2 \\ 2 & 9 & \pi \end{bmatrix}$$

je tipa 2×3 ili $(2, 3)$, a b_{13} je element matrice koji se nalazi na mjestu presjeka prvog retka i trećeg stupca. Odnosno, $b_{13} = -2$.

Jednakost matrica

Matrice $A = [a_{ij}]$ i $B = [b_{ij}]$ su jednake ako su identične, odnosno ako

- su istog tipa (imaju jednak broj redaka i stupaca);
- imaju jednake odgovarajuće elemente, tj. vrijedi $a_{ij} = b_{ij}$ za sve i, j. [7]

Npr., matrice $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ i $B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$ nisu jednake jer nisu istog tipa.

U nastavku je navedeno nekoliko karakterističnih matrica te elementarne transformacije nad matricama.

Kvadratna matrica

Ako je broj redaka m jednak broju stupaca n za matricu kažemo da je ***kvadratna matrica*** reda n. Skup svih kvadratnih matrica reda n označavamo s M_n. [7]

Primjer kvadratne matrice reda 3 je:

$$A = \begin{bmatrix} 8 & 2 & -6 \\ 1 & 8 & 0 \\ -6 & 2 & 8 \end{bmatrix}.$$

Dijagonalna matrica

Dijagonalna matrica je kvadratna matrica čiji su elementi izvan glavne dijagonale jednaki 0, odnosno $a_{ij} = 0$ za sve $i \neq j$. [7]

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 7 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 0 \\ 0 & 6 \end{bmatrix}$$

Ove matrice možemo zapisat i kao: $A = diag(-1, 0, 7)$ i $B = diag(2, 0, 6)$.

Jedinična matrica

Jedinična matrica je dijagonalna matrica čiji su elementi na glavnoj dijagonali jednaki 1. Standardna oznaka za jediničnu matricu je I. [7]

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

Nulmatrica

Matrica čiji su svi elementi jednaki 0 naziva se ***nulmatrica*** i označava s O, bez obzira kojeg je tipa ili reda. [7]

$$O = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad O = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Trokutaste matrice

Gornjotrokutasta matrica je kvadratna matrica čiji su elementi **ispod** glavne dijagonale jednaki 0. [7]

$$A = \begin{bmatrix} 0 & 1 & 4 \\ 0 & 7 & 12 \\ 0 & 0 & -3 \end{bmatrix} \quad B = \begin{bmatrix} 8 & 4 & 1 & -2 \\ 0 & -11 & 5 & 6 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 7 \end{bmatrix}$$

Donjotrokutasta matrica je kvadratna matrica čiji su elementi **iznad** glavne dijagonale jednaki 0. [7]

$$A = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix}$$

Matrica kao vektor

Matrice koje imaju samo jedan redak ili samo jedan stupac nazivamo vektorima. Ako je matrica tipa $(1, n)$ ona je ***jednoredna matrica*** ili vektor-redak, a ako je matrica tipa $(m, 1)$ ona je ***jednostupčana matrica*** ili vektor-stupac. [7]

Matrica $A = \begin{bmatrix} 3 & 0 & 0 & 7 \end{bmatrix}$ je jednoredna, a matrica $B = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 7 \end{bmatrix}$ je jednostupčana.

Transponirana matrica

Transponirana matrica matrice A tipa (m, n) je matrica A^T tipa (n, m) za koju vrijedi

$$[A^T]_{ij} = [A]_{ji}, \quad \forall i, j.$$

Dakle, retci matrice A postaju stupci matrice A^T. [3]

Primjer transponiranja:

$$A = \begin{bmatrix} 3 & -1 & 2 \\ 7 & 13 & 6 \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} 3 & 7 \\ -1 & 13 \\ 2 & 6 \end{bmatrix}.$$

Simetrična matrica

Kvadratna matrica $A = [a_{ij}]$ je ***simetrična*** ako vrijedi

$$a_{ij} = a_{ji}, \quad \forall i, j.$$

Drugim riječima, zrcaljenjem s obzirom na glavnu dijagonalu matrica se ne mijenja [3] :

$$A = \begin{bmatrix} 1 & 4 \\ 4 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 8 & 4 & 1 & -2 \\ 4 & 0 & 5 & 6 \\ 1 & 5 & 1 & 3 \\ -2 & 6 & 3 & 7 \end{bmatrix}.$$

Antisimetrična matrica

Kvadratna matrica $A = [a_{ij}]$ je ***antisimetrična*** ako vrijedi

$$a_{ij} = -a_{ji}, \quad \forall i, j.$$

Kako za elemente na glavnoj dijagonali vrjedi $i = j$, dobije se da je $a_{ii} = -a_{ii}$ odnosno $a_{ii} = 0$ za svaki i. Prema tome, antisimetrične matrice na glavnoj dijagonali imaju nule. [3]

$$A = \begin{bmatrix} 0 & 4 \\ -4 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & -4 & 2 \\ 4 & 0 & -6 \\ -2 & 6 & 0 \end{bmatrix}.$$

Elementarne transformacije

Elementarne transformacije nad matricom A:

- zamjena dva retka (stupca) matrice A;
- množenje dva retka (stupca) matrice brojem različitim od nule;
- dodavanje nekog retka (stupca) matrice nekom drugom retku (stupcu) te matrice. [8]

Neka su $A, B \in M_{mn}\mathbb{R}$. Matrica A je ***ekvivalentna*** matrici B, oznaka $A \sim B$, ako se B može dobiti iz A konačnim brojem elemetarnih operacija nad matricom A. [8]

4. Operacije s matricama

4.1. Zbrajanje matrica

Zbrajati i oduzimati mogu se samo matrice istog tipa, a rezultat je opet matrica istog tipa kao početne matrice.

Neka su $A = [a_{ij}]$ i $B = [b_{ij}]$ matrice tipa (m, n). **Zbroj matrica** A i B je matrica $C = [c_{ij}]$ tipa (m, n) za koju vrijedi

$$c_{ij} = a_{ij} + b_{ij}, \quad \forall i, j. \quad \text{[3]}$$

Na primjer:

$$\begin{bmatrix} 0 & 4 & 12 \\ 2 & 4 & -3 \\ -4 & 0 & -9 \end{bmatrix} + \begin{bmatrix} 2 & -4 & 2 \\ 4 & 0 & -6 \\ 2 & 6 & 0 \end{bmatrix} = \begin{bmatrix} 0+2 & 4+(-4) & 12+2 \\ 2+4 & 4+0 & -3+(-6) \\ -4+2 & 0+6 & -9+0 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 14 \\ 6 & 4 & -9 \\ -2 & 6 & -9 \end{bmatrix}$$

Dakle, zbroj dviju matrica A i B je matrica čiji elementi predstavljaju zbroj odgovarajućih elemenata matrica A i B. Budući da je zbrajanje matrica ekvivalentno zbrajanju realnih brojeva, kod zbrajanje matrica vrijede ista svojstva kao kod zbrajanja realnih brojeva. Ako su A, B, C matrice tipa (m, n) za njih vrijedi:

- asocijativnost $(A + B) + C = A + (B + C)$;
- komutativnost $A + B = B + A$;
- neutralni element kod zbrajanja matrica je nulmatrica $A + O = A = O + A$;
- postojanje suprotnog elementa (***suprotna matrica*** - matrica koja se dobije iz početne tako da se svakom elementu promijeni predznak) $A + (-1)A = O$. [3]

4.2. Oduzimanje matrica

Dvije matrice istog tipa oduzimamo tako da im oduzmemo odgovarajuće elemente i kao rezultat dobijemo matricu istog tipa kao i početne. Ako je $A = [a_{ij}]$ i $B = [b_{ij}]$, tada je $A - B = [a_{ij} - b_{ij}]$. [3]

Na primjer:

$$\begin{bmatrix} 0 & 4 & 12 \\ 2 & 4 & -3 \\ -4 & 0 & -9 \end{bmatrix} - \begin{bmatrix} 2 & -4 & 2 \\ 4 & 0 & -6 \\ 2 & 6 & 0 \end{bmatrix} = \begin{bmatrix} 0-2 & 4-(-4) & 12-2 \\ 2-4 & 4-0 & -3-(-6) \\ -4-2 & 0-6 & -9-0 \end{bmatrix} = \begin{bmatrix} -2 & 8 & 10 \\ -2 & 4 & 3 \\ -6 & -6 & -9 \end{bmatrix}$$

Oduzimanje se može definirati i kao pribrajanje suprotne matrice: $A - B = A + (-1)B$.

4.3. Množenje matrice skalarom

Nekaj je $A = [a_{ij}] \in M_{mn}$ i $k \in \mathbb{R}$.

Produkt matrice A i realnog broja k je matrica $C[c_{ij}] \in M_{mn}$ takva da je

$$c_{ij} = ka_{ij}, \quad \forall i, j. \quad \text{[3]}$$

Dakle, matrica se množi skalarom tako da se svaki njezin element množi tim skalarom. Na primjer, ako je

$$A = \begin{bmatrix} 0 & 4 & 12 \\ 2 & 4 & -3 \\ -4 & 0 & -9 \end{bmatrix},$$

onda je

$$2A = \begin{bmatrix} 0 & 8 & 24 \\ 4 & 8 & -6 \\ -8 & 0 & -18 \end{bmatrix}.$$

Brojem možemo množiti bilo koju matricu. Također, množenjem jedinične matrice nekim brojem dobijemo ***skalarnu matricu***. Isto kao i kod zbrajanja, kod množenja matrica skalarom vrijede ista svojstva kao kod množenja realnih brojeva. Ako su $k, l \in \mathbb{R}$, a $A, B \in M_{mn}$ tada vrijedi:

- kvaziasocijativnost $k(lA) = (kl)A$;
- postojanje neutralnog elementa $1 \cdot A = A$;
- distributivnost u odnosu na zbrajanje skalara $(k + l)A = kA + lA$;
- distributivnost u odnosu na zbrajanje matrica $k(A + B) = kA + kB$. [3]

4.4. Množenje matrica

Množenje matrica nije analogno množenju brojeva. Za razliku od zbrajanja, množenje matrica je operacija u kojoj ni faktori, ni rezultat nisu matrice istog tipa. Prije definiranja samog množenja moramo znati da se množiti mogu samo ulančane matrice.

Matrice $\mathbf{A}$ i $\mathbf{B}$ su ulančane ako je broj stupaca matrice A jednak broju redaka matrice B, tj. ako je A tipa (m, n), a B tipa (n, p). Prema tome, može se uočiti:

- da su dvije kvadratne matrice istog reda uvijek ulančane;
- ako je A ulančana s B, ne vrijedi nužno da je B ulančana s A (osim ako je $m = n$). [7]

Primjerice, matrica $A \in M_{32}$ je ulančana s matricom $B \in M_{24}$, ali matrica $B \in M_{24}$ nije ulančana s matricom $A \in M_{32}$.

Produkt matrice $A = a_{ij}$ tipa (m, n) i matrice $B = b_{ij}$ tipa (n, p) je matrica $C = c_{ij}$ tipa (m, p) za koje vrijedi

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \quad \text{[3]}$$

Dakle, element matrice C dobije se tako da se elementi i-tog retka matrice A pomnože s odgovarajućim elementima j-tog stupca matrice B, pa se dobiveni produkti zbroje.

Neka je:

$$A = \begin{bmatrix} 1 & 4 & 3 \\ 2 & 0 & -1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 2 \\ 4 & 3 \\ -2 & 1 \end{bmatrix}$$

tada je

$$AB = \begin{bmatrix} 1 & 4 & 3 \\ 2 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 4 & 3 \\ -2 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} (1 \cdot 1 + 4 \cdot 4 + 3 \cdot (-2)) & (1 \cdot 2 + 4 \cdot 3 + 3 \cdot 1) \\ (2 \cdot 1 + 0 \cdot 4 + (-1) \cdot (-2)) & (2 \cdot 2 + 0 \cdot 3 + (-1) \cdot 1) \end{bmatrix} = \begin{bmatrix} 11 & 17 \\ 4 & 3 \end{bmatrix},$$

a

$$BA = \begin{bmatrix} 1 & 2 \\ 4 & 3 \\ -2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 4 & 3 \\ 2 & 0 & -1 \end{bmatrix} =$$

$$= \begin{bmatrix} (1 \cdot 1 + 2 \cdot 2) & (1 \cdot 4 + 2 \cdot 0) & (1 \cdot 3 + 2 \cdot (-1)) \\ (4 \cdot 1 + 3 \cdot 2) & (4 \cdot 4 + 3 \cdot 0) & (4 \cdot 3 + 3 \cdot (-1)) \\ ((-2) \cdot 1 + 1 \cdot 2) & ((-2) \cdot 4 + 1 \cdot 0) & ((-2) \cdot 3 + 1 \cdot (-1)) \end{bmatrix} = \begin{bmatrix} 5 & 4 & 1 \\ 10 & 16 & 9 \\ 0 & -8 & -7 \end{bmatrix}$$

Iz ovog primjera se vidi da množenje matrica nije komutativno. Odnosno,

$$A \cdot B \neq B \cdot A.$$

Matrice A i B za koje je $A \cdot B = B \cdot A$ nazivamo ***komutativnim matricama***. [8]

Na primjer,

$$\begin{bmatrix} 1 & 1 \\ -2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}, \quad \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ -2 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}.$$

Svojstva množenja matrica

Ako su A, B, C matrice, a $k \in \mathbb{R}$, za množenje matrica vrijedi:

- asocijativnost $A(BC) = A(BC)$;
- kvaziasocijativnost $k(AB) = (kA)B = A(kB)$;
- distributivnost prema zbrajanju $(A + B)C = AC + BC, \quad A(B + C) = AB + AC$;
- $(AB)^T = B^T A^T$;
- neutralni element za množenje je jedinična matrica $AI = IA = A$, gdje je A kvadratna matrica.

kada su god navedeni produkti definirani. [8]

U nastavku su navedena zanimljiva svojstva množenja matica koja se razlikuju u odnosu na množenje u skupu realnih brojeva. U skupu jednakost $ab = 0$ povlači da je $a = 0$ ili $b = 0$, dok u algebri matrica nije tako. Produkt dviju matrica može biti nulmatrica, a da nijedan faktor nije 0. Na primjer,

$$\begin{bmatrix} 1 & -2 & 4 \\ -2 & 3 & 5 \end{bmatrix} \cdot \begin{bmatrix} 2 & 4 \\ 3 & 6 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Ove matrice su takozvani djelitelji nule, odnosno matrice različite od nulmatrice, za koje vrijedi $AB = 0$. [1]

Također, u skupu svaki element $a \neq 0$ ima svoj multiplikativni inverz, odnosno element $a^{-1} \in \mathbb{R}$ sa svojstvom $aa^{-1} = a^{-1}a = 1$. Na primjer, $a = 3$ i $a^{-1} = \frac{1}{3}$. Međutim, kod množenja matrica multiplikativni inverz ne mora uvijek nužno postojati. Matrice koje imaju svoj multiplikativni inverz su ***regularne matrice*** (vidi 4.6). [1]

4.5. Determinante

Svakoj se kvadratnoj matrici pridružuje skalar - njezina determinanta. Osim što se koriste kao sredstvo za rješavanje sustava linearnih jednadžbi, determinante daju važne informacije o rangu i regularnosti dane matrice. [8]

Determinanta prvog reda

Determinanta matrice $A = [a_{11}]$ je $\det A = |a_{11}| = a_{11}$. [3]

Na primjer, ako je $A = [5]$, $\det A = |5| = 5$.

Determinanta drugog reda

Za matrice drugog reda determinanta se računa ovako:

$$\det A = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$ [8]

Na primjer, za matricu $A = \begin{bmatrix} 3 & -3 \\ 4 & 5 \end{bmatrix}$, $\det A = \begin{vmatrix} 3 & -3 \\ 4 & 5 \end{vmatrix} = 3 \cdot 5 - (-3) \cdot 4 = 27.$

Determinanta trećeg reda

Za jednostavnije računanje determinante trećeg reda koristi se **Sarrusovo pravilo** i ono vrijedi samo za determinante trećeg reda. Prva dva stupca determinante se dopišu iza trećeg, zatim se zbroje produkti elemenata u smjeru glavne dijagonale, te oduzmu produkti elemenata u smjeru sporedne dijagonale [3]:

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \begin{matrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{matrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}$$

$$-a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32} - a_{13}a_{22}a_{31}$$

Na primjer, za matricu

$$A = \begin{bmatrix} 2 & -1 & 4 \\ 1 & 0 & 4 \\ 3 & 2 & -2 \end{bmatrix}$$

njezina determinanta iznosi

$$\det A = \begin{vmatrix} 2 & -1 & 4 \\ 1 & 0 & 4 \\ 3 & 2 & -2 \end{vmatrix} \begin{matrix} 2 & -1 \\ 1 & 0 \\ 3 & 2 \end{matrix} = 2 \cdot 0 \cdot (-2) + (-1) \cdot 4 \cdot 3 + 4 \cdot 1 \cdot 2$$

$$-4 \cdot 0 \cdot 3 - 2 \cdot 4 \cdot 2 - (-1) \cdot 1 \cdot (-2) = -22.$$

Svojstva determinanti

- Transponiranjem matrice vrijednost determinante se ne mijenja: $\det A = \det A^T$;
- Ako zamjenimo mjesta bilo koja dva retka ili stupca matrice, determinanta mijenja predznak;
- Ako matrica A ima dva jednaka retka ili stupca, onda je $\det A = 0$;

- Determinanta gornjo ili donjotrokutaste matrice jednaka je produktu elemenata na glavnoj dijagonali;
- Ako su svi elementi jednog retka ili stupca matrice A jednaki nula, onda je $\det A = 0$;
- Determinanta jedinične matrice I jednaka je 1;
- Ako bilo koji redak ili stupac u matrici A pomnožimo realnim brojem k, determinanta rezultirajuće matrice jednaka je $k \det A$;
- Zajednički faktor svih elemenata nekog retka ili stupca može se izlučiti izvan determinante;
- Ako neki redak (stupac) u determinanti dodamo nekom drugom retku (stupcu), vrijednost determinante se neće promijeniti;
- Ako umnožak nekog retka (stupca) s nekim brojem dodamo nekom drugom retku (stupcu), vrijednost determinante se neće promijeniti;
- Binet-Cauchyjev teorem. A ko su A i B kvadratne matrice istog reda, tada je $\det AB = \det A \det B$. [3]

Determinanta n-tog reda

Determinatu matrice bilo kojeg reda može se računati pomoću **Laplaceovog razvoja** u kojem se vrijednost determinante izražava pomoću determinanti nižeg reda.

Determinanta matrice A reda n može se razvijati:

- po i-tom retku:
$$\det A = \sum_{j=1}^{n} a_{ij} A_{ij} = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} M_{ij};$$
- po j-tom stupcu:
$$\det A = \sum_{i=1}^{n} a_{ij} A_{ij} = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} M_{ij}.$$ [7]

Gdje je M_{ij} **minora** elementa a_{ij}, a A_{ij} **algebarski komplement** ili **kofaktor** elementa a_{ij}.

Minora elementa a_{ij} je determinanta matrice koja se dobije brisanjem i-tog retka i j-tog stupca matrice A. Pomoću minora definiramo algebarske komplemente matričnih elemenenata: $A_{ij} = (-1)^{i+j} M_{ij}$. [7]

Prema tome, računanje determinante petog reda svodi se na računanje najviše pet determinanti četvrtog reda, što dovodi do računanja najviše 20 determinanti trećeg reda. Zato, primjena Laplaceovog pravila ima smisla za determinante do reda 4. Determinante višeg reda se računaju tako da se svedu na gornju ili donju trokutastu formu korištenjem elementarnih transformacija i svojstva determinanta. [3]

4.6. Inverzna matrica

Kvadratna matrica $A \in M_n(\mathbb{R})$ reda n je ***invertibilna*** ako postoji matrica $X \in M_n(\mathbb{R})$ takva da vrijedi

$$AX = XA = I$$

gdje I označava jediničnu matricu reda n.

U tom se slučaju matrica X zove multiplikativni inverz ili ***inverzna matrica*** od A i označava s A^{-1}. [8]

Invertibilnu matricu nazivamo još i ***regularna*** matrica. Matrica $A \in M_n(\mathbb{R})$ koja nema multiplikativni inverz je ***singularna***. [8]

Kako znamo da matrica ima svoj inverz, odnosno da je regularna? Kvadratna matrica A je regularna ako je $detA \neq 0$. [8]

Svojstva inverzne matrice

- $(A^{-1})^{-1} = A$;
- $(A^T)^{-1} = (A^{-1})^T$;
- $(AB)^{-1} = B^{-1}A^{-1}$;
- $I^{-1} = I$ gdje je I kvadratna matrica reda n [1]

Izračun inverzne matrice

Svaka regularna matrica ima inverznu matricu za koju vrijedi :

$$A^{-1} = \frac{1}{\det A} A^*$$

gje je A^* adjunkta matrice A. ***Adjunkta matrice*** je transponirana matrica kofaktora matrice A. [8]

Računanje inverzne matrice po ovoj formuli ima smisla za matrice do reda 3, inverz matrice bilo kojeg reda efikasno se računa pomoću **Gaussovog postupka**. [3]

U nastavku je izračunat inverz matrice $A = \begin{bmatrix} 1 & 5 & 1 \\ 0 & 5 & 2 \\ 1 & -1 & 0 \end{bmatrix}$ Gaussovim postupkom.

Prvo se formira par matrica A i I, koje su odjeljene isprekidanom crtom. Zatim se provode elementarne transformacije po recima tako dugo dok se na lijevoj strani ne dobije jedinična matrica. Matrica koju dobijemo na desnoj strani je inverzna matrica matrice A. [3]

$$\left[\begin{array}{ccc|ccc} 1 & 5 & 1 & 1 & 0 & 0 \\ 0 & 5 & 2 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 \end{array}\right] \begin{array}{l} \\ \\ \cdot(-1) \\ \\ + \end{array} \sim \left[\begin{array}{ccc|ccc} 1 & 5 & 1 & 1 & 0 & 0 \\ 0 & 5 & 2 & 0 & 1 & 0 \\ 0 & -6 & -1 & -1 & 0 & 1 \end{array}\right] \begin{array}{l} \\ \\ \cdot 2 \\ + \\ + \end{array} \sim$$

$$\left[\begin{array}{ccc|ccc} 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & -7 & 0 & -2 & 1 & 2 \\ 0 & -6 & -1 & -1 & 0 & 1 \end{array}\right] \begin{array}{l} \\ \cdot -\frac{1}{7} \\ \cdot(-1) \end{array} \sim \left[\begin{array}{ccc|ccc} 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & \frac{2}{7} & -\frac{1}{7} & -\frac{2}{7} \\ 0 & 6 & 1 & 1 & 0 & -1 \end{array}\right] \begin{array}{l} + \\ \\ \cdot(-6) \\ + \end{array} \sim$$

$$\sim \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & \frac{2}{7} & -\frac{1}{7} & \frac{5}{7} \\ 0 & 1 & 0 & \frac{2}{7} & -\frac{1}{7} & -\frac{2}{7} \\ 0 & 0 & 1 & -\frac{5}{7} & \frac{6}{7} & \frac{5}{7} \end{array}\right]$$

Prema tome, inverz matrice A je

$$A^{-1} = \begin{bmatrix} \frac{2}{7} & -\frac{1}{7} & \frac{5}{7} \\ \frac{2}{7} & -\frac{1}{7} & -\frac{2}{7} \\ -\frac{5}{7} & \frac{6}{7} & \frac{5}{7} \end{bmatrix}.$$

5. Python

Python je interpreterski, interaktivni programski jezik visoke razine. Interpreterski programski jezici su jezici kod kojih se izvorni kod izvršava direktno uz pomoć interpretera. Kod ovakvog tipa programskih jezika nema potrebe za kompajliranjem prije izvršavanja, tj. prevođenja u izvršni oblik. Zbog toga je razvojni proces kraći nego u Javi ili C-u, ali su programi pisani u Pythonu sporiji u izvršavanju. [9]

Podržava različite programske paradigme, kao što su objektno orjentirana, funkcionalna itd. Prema tome na različite načine se može pristupiti istom problemu. Nastao je 1990-ih godina u Nizozemskoj, a razvio ga je programer Guido van Rossum. Ime je dobio prema kultnoj britanskoj TV komediji "Monty Python's Flying Circus". Od svojih početaka Python je **Open source** što znači da je instalacija i preuzimanje besplatno i jednostavno, a izvorni kod je lako dostupan. [18]

Slika 1: Python logo (Izvor: Python.org)

Pyton je jezik opće namjene. To znači da je dizajniran za pisanje programa različitih domena upotrebe. Neka od područja korištenja Pythona su:

- kao i drugi objektno orjentirani jezici koristi se za razvoj Weba. Django i Flask dva su najpopularnija Python okvira za razvoj Web aplikacija;
- koristi se za pisanje složenih znanstvenih aplikacija i numeričkih metoda. Objekti iz stvarnog svijeta pretvaraju se u matematičke modele i njihovo djelovanje simulira se izvršavanjem formula;
- vrlo je popularan u području umjetne inteligencije, a posebno kod implementacija algoritma za različite metode strojnog učenja;
- mnoge kompanije ga koriste za razna testiranja, razvoj korisničkog sučelja, animaciju, razvoj igrica, obradu slike, OS programiranje, kreiranje XML fajlova, analizu podataka i drugo. [17] [5]

Mnogi se slažu da je jedna od glavnih značajki Pythona njegova jednostavnost. Pythonova jednostavna i intuitivna sintaksa koja se lako uči naglašava čitljivost i time smanjuje troškove

održavanja programa. Također, neovisan je o platformi što znači da se programi pisani u Pythonu mogu pokrenuti na gotovo svakoj platformi korištenoj danas, bez da se mijenjaju. [10]

Još jedna bitna značajka je da se može povezivati s komponentama pisanim u drugim jezicima. Na primjer skripta pisana u Pythonu može pozivati postojeće C/C++ biblioteke ili komunicirati sa klasama u Javi. [10]

Python podržava module i pakete, što potiče modularnost programa i ponovnu upotrebu koda. Za veću funkcionalnost Pythona i lakše korištenje istoga potrebno je dodatno instalirati pojedine biblioteke. Biblioteke uključuju razne module i funkcije koje podržavaju različite operacije s različitim vrstama podataka. U ovom radu je korištena biblioteka NumPy koja je opisana u poglavlju 5.1. [4]

Zbog svoje snage, jednostavnosti i široke upotrebe, nije čudo da se Python nalazi na drugom mjestu ljestvice popularnosti programskih jezika koju izdaje analitička tvrtka RedMonk. [15]

Python je našao primjenu u mnogim poznatim tvrtkama i aplikacijama kao što su:

- Google (YouTube, Gmail, mnogi djelovi tražilice);
- Yahoo (Yahoo Groups, Yahoo Maps);
- CERN (za složene matematičke izračune);
- Netflix (Python koristi za analizu podataka, za preporuku filmova i serija korisnicima);
- NASA (ubrzanje procesa kodiranja). [6]

5.1. NumPy

Osnovne operacije koje se koriste u znanstvenom programiranju uključuju nizove, matrice, integrale, rješavanje diferencijalnih jednadžbi, statistiku i još mnogo toga. Python, prema zadanim postavkama, nema ugrađenu nijednu od ovih funkcionalnosti, osim nekih osnovnih matematičkih operacija kao što su zbrajanje, oduzimanje, množenje. Međutim, te operacije rade samo sa skalarima, tj. varijablama, a ne s nizovima ili matricama. Zato postoje posebne biblioteke koje omogućuju učinkovitu upotrebu jezika u te svrhe. Jedna od tih biblioteka je NumPy. [2]

NumPy (engl. *Numeric Python*) je jedna od najpopularnijih paketa za znanstveno programiranje u Pythonu. Dolazi u obliku open-source modula koji pruža podršku za upravljanje opširnim, više dimenzionalnim nizovima i matricama, te implementira široki spektar matematičkih funkcija za efikasno upravljanje nad tim poljima. Između ostalog, NumPy sadrži:

- optimiziran objekt za predstavljanje višedimenzionalnih polja - ndarray;
- ugrađene funkcije za automatski *broadcasting* - proširivanje dimenzije polja;
- funkcije za linearnu algebru, statistiku, trigonometriju, računanje matricama;

- alate za povezivanje s C/C++ ili Fortran kodom. [21]

Temeljni objekt cijele biblioteke je ***ndarray***. To je objekt kojim se opisuju višedimenzionalna polja. Objekt *ndarray* je homogena zbirka elemenata indeksiranih s N cijelih brojeva. Dva bitna podatka koja ga definiraju su: oblik polja i tip podatka od kojih se sastoji. [2]

NumPy polja su slična listama u Pythonu. Oba objekta predstavljaju određenu zbirku podataka i indeksirani su brojevima. No postoje dvije bitne razlike. Prva je da se u listu mogu spremiti elementi različitih tipova podataka. Na primjer, prvi element može biti lista, drugi riječnik (engl. *dictionary*) i slično. S druge strane, svako NumPy polje je homogena zbirka iste vrste podatka, svaki element zauzima blok memorije iste veličine, a svaki blok memorije se tumači na potpuno isti način. Druga razlika je da polja nisu dinamička. To znači da se ne može mijenjati njihova veličina. Ako se želi dodati element u listu on se samo nadoda u već postojeću listu. Međutim, ako se želi dodati element u polje, stvara se novo polje koje se sastoji od starih i novih elemenata. [2]

Još jedan bitan objekt koji pruža NumPy su univerzalne fukcije (*ufunc*). Trenutno postoji više od 70 univerzalnih funkcija koje pokrivaju širok izbor operacija, od osnovnih aritmetičkih operacija do trigonometrije. One su instanca *numpy.ufunc* klase i djeluju nad odgovarajućim elementima polja. To znači da polja moraju biti istih dimenzija. Metoda koja je povezana s univerzalnim funkcijama je *Broadcasting*. *Broadcasting* (vidi 5.1.4) omogućuje da univerzalne funkcije rade s poljima različitog oblika. [16] [14]

NumPy se često koristi zajedno s paketima SciPy (engl. *Scientific Python*) i Matplotlib. Ova se kombinacija može koristiti kao zamjena za komerijalni MatLab, popularnu platformu za tehničko računanje. SciPy paket proširuje funkcionalnost NumPy-a s funkcijama za integraciju, interpolaciju, statistiku, linearnu algebru... [2]

Pyton, prema zadanim postavkama, nema ugrađeni objekt za stvaranje matrica, ali ih može prikazati kao ugnježdene liste. Međutim, znanstveno programiranje često zahtjeva visoke brzine izvođenja što Pythonove liste ne pružaju. Ako se radi o velikoj količini podataka NumPy polja će biti puno brža i zauzet će manje memorije. Također, s NumPy poljima se može računati na načine na koje se ne može s običnim listama. [2]

NumPy se uvozi pomoću sljedeće naredbe.

```
>>> import numpy as np
```

Ispis 5.1: Učitavanje NumPy biblioteke

5.1.1. Stvaranje matrica

Matrice su dvodimenzionalna polja. Prema tome, NumPy omogućuje da se matrica prikaže kao objekt *np.array* klase ili kao objekt njezine podklase *np.matrix*. Glavna razlika je da NumPy matrice mogu imati najviše dvije dimenzije, dok su NumPy polja n-dimenzionalna.

S obzirom da *np.matrix* podklasa nasljeđuje sve atribute i metode *np.array* klase, s NumPy poljima se može raditi sve što i s NumPy matricama, uz drugačije notacije u linearnoj algebri. [2]

Matrice se mogu kreirati na nekoliko načina: pretvorbom iz lista, ugrađenim funkcijama (*ones(), zeros()*...), pozivanjem posebnih funkcija biblioteke (*random()*) ili čitanjem s diska. [20] U nastavku je navedeno nekoliko najčešćih primjera kreiranja matrica.

Već je spomenuto da se matrica može prikazati kao ugnježđena lista. Ta lista, pretvara se u polje pozivanjem ***array()*** funkcije. Funkcija ima dva parametra. Prvi je lista koja se želi pretvoriti u polje, a drugi je tip podataka koji se želi spremiti u polje. Objekt ***dtype*** može biti bilo koji od standardnih Python tipova: int, fload, complex,...U primjeru su navedeni i najvažniji atributi objekata *ndarray* klase: ***ndim*** - broj dimenzija; ***shape*** - oblik polja; ***size*** - broj elemenata u polju. [20]

```
>>> A = np.array([[0, 4, 12], [2, 4, -3], [-4, 0, -9]], dtype=int)
>>> print('A =', A)

  A = [[0 4 12]
    [2 4 -3]
    [-4 0 -9]]

>>> print(A.ndim)
  2

>>> print(A.size)
  9

>>> print(A.shape)
  (3,3)
```

Ispis 5.2: *array()* funkcija

Za stvaranje polja koriste se i ugrađene funkcije poput:

- ***empty()*** stvara proizvoljno polje određenog oblika i tipa podataka. [11]

```
>>> B = np.empty([3,2], dtype = int)
>>> print('B =', B)

    B = [[22473951 1047593886]
      [69390443 3928504336]
      [22119585 84756320]]
```

Ispis 5.3: *empty()* funkcija

- ***zeros()*** će kreirati polje čiji su svi elemneti nule. Dok će funkcija ***ones()*** kreirati polje čiji su svi elementi jedinice. Kao parametre imaju željeni oblik polja i tip podataka. [11]

```
>>> C = np.zeros((3, 4), dtype=int)
>>> print('C =', C)

    C = [[0 0 0 0]
      [0 0 0 0]
      [0 0 0 0]]

>>> D = np.ones((3, 3), dtype=float)
>>> print('D =', D)

    D = [[1. 1. 1.]
      [1. 1. 1.]
      [1. 1. 1.]]
```

Ispis 5.4: *zeros()* i *ones()* funkcije

- ***full()*** stvara polje definiranog oblika čiji su svi elementi jednaki definiranoj vrijednosti skalara. [11]

```
>>> E = np.full((3,2), 6)
>>> print('E =', E)

    E = [[6 6]
      [6 6]
      [6 6]]
```

Ispis 5.5: *full()* funkcija

- ***identity()*** stvara jediničnu matricu. Odnosno kvadratno polje s jedinicama na glavoj dijagonali. [11]

```
>>> I = np.identity(4, dtype=float)
>>> print('I =', I)

    I = [[1. 0. 0. 0.]
      [0. 1. 0. 0.]
      [0. 0. 1. 0.]
      [0. 0. 0. 1.]]
```

Ispis 5.6: *identity()* funkcija

- ***arange()*** stvara polje s ravnomjerno raspoređenim vrijednostima u određenom intervalu. Funkcija kao parametar uzima početnu i završnu vrijednost intervala te razmak između

vrijednosti. Ukoliko razmak nije definiran on je, prema postavkama, jedinica. Da bi se promjenio oblik polja koristi se funkcija ***reshape()***. Ona za parametar ima željeni oblik matrice. [11]

```
>>> F = np.arange(1, 7, dtype = int)
>>> print('F =',F)

    F = [[ 1 2 3 4 5 6]]

>>> G = np.arange(0, 27, 3).reshape(3,3)
>>> print('G =',G)

    G = [[0 3 6]
      [9 12 15]
      [18 21 24]]
```

Ispis 5.7: *arange()* funkcija

- ***linspace()***, kao i ***arange()***, stvara polje elemenata s ravnomjerno raspoređenim vrijednostima u određenom intervalu. Međutim, *linspace()* uključuje i završnu vrijednost intervala. Isto tako, bolje je koristiti *linspace()* ako je razmak između vrijednosti decimalni broj.

  Za parametre uzima početnu i završnu vrijednost intervala. Dodatno se naredbom *num* može odrediti od koliko će se elemenata sastojati polje. Ukoliko se ovaj parametar ne definira *linspace()* vraća polje od 50 elemenata u zadanom intervalu. [11]

```
>>> Y = np.linspace(1, 4, num = 8)
>>> print('Y =',Y)

    Y = [[1. 1.42857143 1.85714286 2.8571429 2.71428571 3.14285714
        3.57142857 4.]]
```

Ispis 5.8: *linspace()* funkcija

- ***diag()*** stvara dijagonalnu matricu. Za argument uzima listu elemenata koji se raspoređuju po glavnoj dijagonali. [11]

```
>>> Z = np.diag([3, -4, 13, 9])
>>> print('Z =',Z)
    Z = [[3 0 0 0]
      [0 -4 0 0]
      [0 0 13 0]
      [0 0 0 9]]
```

Ispis 5.9: *diag()* funkcija

5.1.2. Indeksiranje u matricama

Polja mogu sadržavati velike količine podataka. Međutim, ponekad se javlja potreba za pristupom samo određenim elementima ili elementima koji zadovoljavaju neki kriterij. To omogućuje indeksiranje.

Pojedinim elementima polja se pristupa pomoću indeksa. NumPy polja indeksiraju se od nule, prema tome prvi element ima indeks 0, drugi indeks 1, a zadnji index -1. [12]

Ako se želi pristupiti većem broju elemenata koriste se tzv. rezovi (engl. *slice*). Sintaksa reza je polje[početak:kraj:korak], gdje je početak uključen u rez, a kraj nije. [12]

```
>>> B = np.arange(1, 11)
>>> print('B =', B)

B = [[1 2 3 4 5 6 7 8 9 10]]

>>> print(B[4])      #peti element: 5

>>> print(B[-1])     #zadnji  element: 10

>>> print(B[1:5])    #rez od drugog do petog elementa: [2 3 4 5]

>>> print(B[0:-1:2]) #svaki drugi element, počševi od prvog: [1 3 5 7 9]
```

Ispis 5.10: Dohvaćanje elemenata iz jednodimenzionalnog polja

```
>>> A = np.array([[2, -1, 4, 5], [1, 0, 4, 12], [3, 2, -2, -7],[8, 3, -2,
    0]], dtype=int)
>>> print('A =', A)

A = [[2, -1, 4, 5]
[1, 0, 4, 12]
[3, 2, -2, -7]
[8, 3, -2, 0]]

>>> print(A[1,-1]) #zadnji element drugog retka: 12

>>> print(A[:,0]) #prvi stupac matrice: [2 1 3 8]

>>> print(A[-1,1:4]) #zadnji red, elementi s ideksima 1, 2, 3 : [3 -2 0]

>>> print(A[0:2,1]) #drugi element iz prvog i drugog reda : [-1 0]

>>> print(A[0:4:2,::-2]) # Svaki drugi element, u obrnutom poretku, iz
    prvog i trećeg reda matrice A : [[5 -1]
                                     [-7 2]]
```

Ispis 5.11: Dohvaćanje elemenata matrice

5.1.3. Zbrajanje, oduzimanje i množenje matrice skalarom

Aritmetičke operacije nad poljima se izvode nad odgovarajućim elemenatima. To znači da polja moraju biti istih dimenzija ili moraju zadovoljavati pravila za *broadcasting* (5.1.4). [14]

Operacije se izvode pozivanjem univerzalnih funkcija (*ufunc*), npr.:

- ***np.add()*** za zbrajanje,
- ***np.subtract()*** za oduzimanje,
- ***np.multiply()*** za množenje

ili jednostavnije, upotrebom standardnih operatora (+, -, *) koji interno pozivaju pripadajuću funkciju. Također, treba napomenuti da funkcija *np.multiply()* množi odgovarajuće elemente polja, no matrice se tako ne množe. Za matrično množenje se koriste druge metode koje su objašnjene u poglavlju 5.1.5. [14]

```
    A = [[0 4 12]
      [2 4 -3]]

    B = [[2 -4 2]
      [4 0 -6]]

    C = [[1 2 3]]

>>> print('A + B =',np.add(A,B))         # ili (A + B)
>>> print('A - B =',np.subtract(A,B))    # ili (A - B)
>>> print('A * B =',np.multiply(A,B))    # ili (A * B)
>>> print('2A =', 2 * A)
>>> print('A + C =',A + C)

  A + B = [[2 0 14]
      [6 4 -9]]

  A - B = [[-2 8 10]
      [-2 4 3]]

  A * B = [[0 -16 24]
      [8 0 18]]

  2A = [[0 8 24]
     [4 8 -6]]

  A + C = [[1 6 15]
      [3 6 0]]
```

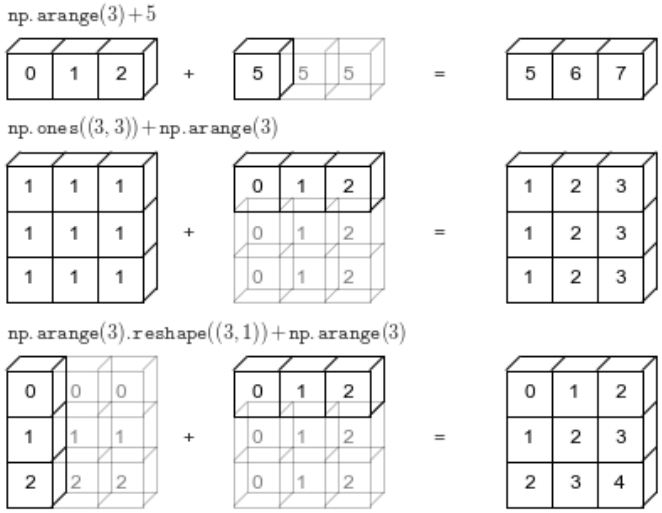
Ispis 5.12: Aritmetičke operacije

5.1.4. *Broadcasting*

Broadcasting omogućuje izvođenje aritmetičkih operacija (zbrajanje, oduzimanje, množenje,...) nad poljima različitih dimenzija.

U ispisu 5.12, *broadcasting* je korišten prilikom zbrajanja matrice A i C. Matrica A je tipa (2,3), matrica C tipa (1,3), ali rezultat njihova zbrajanja je nova matrica tipa (2,3). U osnovi, matrica C se produžila i poprimila oblik matrice A. Novi elementi, u produženoj matrici C, su kopije izvornih skalara.

Kako funkcionira *broadcasting* može se vidjeti na slici 2.



Slika 2: Broadcasting (Izvor: **2016**, VanderPlas)

Svijetle kutije na slici predstavljaju proširene vrijednosti, no Python tim vrijednostima ne dodjeljuje memoriju, već za izračun koristi izvornu skalarnu vrijednost što dovodi do veće efikasnosti programa. [22]

Broadcasting slijedi skup pravila za utvrđivanje interakcije između dva polja. Ukoliko matrice, koje sudjeluju u operaciji, ne zadovoljavaju pravila, ne može se izvršiti proširivanja na zajedničku dimenziju. U tom slučaju, Python javlja grešku kao na slici 3. [22]

```
ValueError: operands could not be broadcast together with shapes (3,2) (3,)
PS C:\Users\Maja\Desktop\tmp>
```

Slika 3: *Broadcasting* greška

5.1.5. Množenje matrica

Jedna od najčešćih operacija u linearnoj algebri je množenje matrica. Množenje matrica nije skalarno množenje. Međutim, svaki element u matrici rezultata je skalarni produkt (engl.

dot product) retka lijeve matrice sa stupcem desne. Za matrično množenje koristi se funkcija ***np.dot()*** ili u Pythonu, verzije 3.5 i više, operator *@*. Ako su matrice objekti *np.matrix* klase za njihovo množenje koristi se i standardni operator za množenje *. Međutim, ako su matrice objekti *np.array* klase, operator * će pomonožiti odgovarajuće elemente matrice što nije matrično množenje. Različite metode množenja matrica prikazani su na primjeru u nastavku. Ako matrice nisu ulančane, Python javlja grešku. [2]

```
>>> A = np.array([[1, 4, 3], [2, 0, -1]])
>>> print('A =', A)

  A = [[1 4 3]
    [2 0 -1]]

>>> B = np.array([[1, 2], [4, 3], [-2, 1]])
>>> print('B =', B)

  B = [[1 2]
    [4 3]
    [-2 1]]

>>> print('AB =', A.dot(B))

  AB = [[11 17]
    [4 3]]

>>> print('A@B =', A@B)

  A@B = [[11 17]
    [4 3]]

>>> print('b^T=', B.transpose())   # ili B.T

  B^T = [[1 4 -2]
      [2 3 1]]

# np.matrix klasa
>>> a = np.matrix([[1, 4, 3], [2, 0, -1]])
>>> b = np.matrix([[1, 2], [4, 3], [-2, 1]])
>>> print('a * b =', a * b)

  a * b = [[11 17]
      [4 3]]
```

Ispis 5.13: Množenje matrica

Još jedna korisna operacija u linearnoj algebri je ***transponiranje*** matrica. Transponiranje se provodi pozivanjem ***transpose()*** funkcije. To je unarna operacija koja matrici pridružuje njoj transponiranu matricu. U gornjem primjeru transponirana je matrica B.

5.1.6. NumPy modul za linearnu algebru

NumPy sadrži modul ***np.linalg*** u kojem se nalaze sve potrebne funkcije za rad s matricama. Neke od osnovnih funkcija su:

- ***np.linalg.det()*** za izračun determinante. Ako matrica nije kvadratna, Phyton javlja grešku;
- ***np.linalg.inv()*** za izračun inverzne matrice. Ukoliko je matrica singularna, Python javlja grešku;
- ***np.linalg.matrix_rank()*** za izračun ranga matrice;
- ***np.trace()*** za izračun traga matrice, odnosno zbroja elemenata na glavnoj dijagonali. [13]

Na primjer, za matricu

$$C = \begin{bmatrix} 2 & -4 & 2 & 8 & 12 \\ 4 & 0 & -6 & 1 & 2 \\ -2 & 6 & 0 & -3 & 2 \\ 4 & -1 & 0 & 1 & 9 \\ -1 & 1 & 0 & -3 & 6 \end{bmatrix}$$

izračunate su njezina determinanta, rang, trag i inverzna matrica:

```
>>> C = np.array([[2, -4, 2, 8, 12], [4, 0, -6, 1, 2], [-2, 6, 0, -3, 2],
    [4, -1, 0, 1, 9], [-1, 1, 0, -3, 6]])
>>> print('C =', C)

    C = [[2 -4 2 8 12]
      [4 0 -6 1 2]
      [-2 6 0 -3 2]
      [4 -1 0 1 9]
      [-1 1 0 -3 6]]

>>> print('det_C =',np.linalg.det(C))

    det_C = -8724

>>> print('rang_C =',np.linalg.matrix_rank(C))

    rang_C = 5

>>> print('trag_C =',np.trace(C))

    trag_C = 9
```

Ispis 5.14: Determinanta, rang i trag matrice C

```
>>> print('inv_C =',np.linalg.inv(C))

    inv_C = [[-0.10797799 -0.03599266 0.0126089 0.29252636 -0.21503897]
             [0.00756534 -0.00252178 0.20930766 0.05593764 -0.16964695]
             [-0.04126547 -0.180421182 0.02498854 0.17973407 -0.13525906]
             [0.11141678 0.03713893 0.08253095 -0.08528198 -0.13480055]
             [0.03645117 0.01215039 0.00848235 -0.00320954 0.09170105]]
```

Ispis 5.15: Inverz matrice C

- ***np.linalg.solve()*** za rješavanje linearnih i matričnih jednadžbi. Računa vrijednost nepoznanice. [13]

  Na primjer, sustav linearnih jednadžbi s tri nepoznanice

$$a - 3b + c = 12$$
$$b + 5c = 4$$
$$9a + 3b - c = -3$$

  može se prikazati u matričnom obliku kao

$$\begin{bmatrix} 1 & -3 & 1 \\ 0 & 1 & 5 \\ 9 & 3 & -1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 12 \\ 4 \\ -3 \end{bmatrix}.$$

Ako se ove matrice, po redu, nazovu D, X i E, jednadžba glasi $DX = E$. Zatim, pomoću funkcije *solve()* dobijemo vrijednosti matrice X, odnosno rješenje sustava.

```
>>> D = np.array([[1, -3, 1], [0, 1, 5], [9, 3, -1]])
>>> E = np.array([[12], [4], [-3]])

>>> X = np.linalg.solve(D,E)
>>> print('X =', X)

  X = [[0.9]
    [-3.21875]
    [1.44375]]
```

Ispis 5.16: *solve()* funkcija

Prema tome, rješenje sustava je: $a = 0.9, \quad b = -3.21875, \quad c = 1.44375$. Također, ova se jednadžba može riješiti pomoću inverzne matrice. Tada ona glasi $X = D^{-1}E$.

5.1.7. Jednadžba $\mathbf{AX + XB = C}$

Jednadžba $AX + XB = C$ naziva se Sylvesterova jednadžba i ona se ne može riješiti pomoću inverzne matrice. Ne može se izlučiti X zato što se uz matricu A nalazi s desne strane, a uz matricu B s lijeve strane. Rješava se tako da se odredi format matrice X i u zadanu jednadžbu se uvrsti matrica tog formata s nepoznatim elementima. Izračuna se matrica na lijevoj strani pa se nakon toga izjednače odgovarajući elementi matrice s lijeve i desne strane. Dobije se sustav linearnih jednadžbi koji se riješi. [3]

Na primjer, ako su matrice

$$A = \begin{bmatrix} 1 & -3 \\ 2 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -4 & 7 & 2 \\ 1 & 3 & 1 \\ -1 & 2 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 6 & -2 & 8 \\ 1 & 3 & 2 \end{bmatrix}$$

matrica X mora biti tipa (2,3), odnosno

$$C = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}.$$

Prema tome, jednadžba glasi

$$\begin{bmatrix} 1 & -3 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} + \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \cdot \begin{bmatrix} -4 & 7 & 2 \\ 1 & 3 & 1 \\ -1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 6 & -2 & 8 \\ 1 & 3 & 2 \end{bmatrix}.$$

Zbrajanjem produkta matrica na lijevoj strani dobije se da je

$$\begin{bmatrix} -3a + b - c - 3d & 7a + 4b + 2c - 3e & 2a + b + 2c - 3f \\ 2a - 3d + e - f & 2b + 7d + 4e + 2f & 2c + 2d + e + 2f \end{bmatrix} = \begin{bmatrix} 6 & -2 & 8 \\ 1 & 3 & 2 \end{bmatrix}.$$

Zatim se matrična jednadžba svede na sustav jednadžbi sa 6 nepoznanica

$$\begin{aligned} -3a + b - c - 3d &= 6 \\ 7a + 4b + 2c - 3e &= -2 \\ 2a + b + 2c - 3f &= 8 \\ 2a - 3d + e - f &= 1 \\ 2b + 7d + 4e + 2f &= 3 \\ 2c + 2d + e + 2f &= 2. \end{aligned}$$

Rješenje sustava je: $a = -1.6, \quad b = 2.1, \quad c = 2.32, \quad d = -0.48, \quad e = 1.28, \quad f = -1.49.$ Primjetimo da se ovaj sustav može prikazati i u obliku matrične jednadžbe

$$\begin{bmatrix} -3 & 1 & -1 & -3 & 0 & 0 \\ 7 & 4 & 2 & 0 & -3 & 0 \\ 2 & 1 & 2 & 0 & 0 & -3 \\ 2 & 0 & 0 & -3 & 1 & -1 \\ 0 & 2 & 0 & 7 & 4 & 2 \\ 0 & 0 & 2 & 2 & 1 & 2 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} 6 \\ -2 \\ 8 \\ 1 \\ 3 \\ 2 \end{bmatrix}.$$

Sada se ova matrična jednadžba može riješiti pomoću inverzne matrice ili funkcije *solve()*.

```
>>> D = np.array([[-3, 1, -1, -3, 0, 0], [7, 4, 2, 0, -3, 0],
              [2, 1, 2, 0, 0, -3], [2, 0, 0, -3, 1, -1],
              [0, 2, 0, 7, 4, 2], [ 0, 0, 2, 2, 1, 2]])
>>> E = np.array([[6], [-2], [8], [1], [3], [2]])

>>> X = np.linalg.solve(D,E)
>>> print('X =', X)

X = [[-1.5983843 ]
    [ 2.09578765]
    [ 2.32256203]
    [-0.47720716]
    [ 1.27986151]
    [-1.48528563]]
```

Ispis 5.17: *linalg.solve()* funkcija

Međutim, Python nudi brži dolazak do rezultata. Već je spomenuto da SciPy proširuje mogućnosti NumPy s funkcijama za linearnu algebru. Prema tome, Sylvesterova jednadžba se efikasnije može riješiti pomoću SciPy modula i funkcije ***linalg.solve_sylvester()***. [19]

```
>>> import numpy as np
>>> from scipy import linalg
>>> A = np.array([[1, -3], [2, 1]])
>>> B = np.array([[-4, 7, 2], [1, 3, 1], [-1, 2, 1]])
>>> C = np.array([[6, -2, 8], [1, 3, 2]])

>>> X =linalg.solve_sylvester(A,B,C)
>>> print('X =', X)

  X = [[-1.5983843   2.09578765  2.32256203]
      [-0.47720716  1.27986151 -1.48528563]]
```

Ispis 5.18: *linalg.solve_sylvester()* funkcija

6. Zaključak

Matrice omogućuju kompaktan zapis velike količine podataka i ubrzavaju operacije nad tim podacima. Iz tog razloga često su korištene u znanosti, ali su i nezamjenjiv koncept za rješavanje praktičnih problema. Prvotna primjena matrica za jednostavnije rješavanje sustava linearnih jednadžbi napredovala je te se danas matrice primjenjuju u računarskoj znanosti, fizici, kemiji, ekonomiji, kriptografiji i mnogim drugim disciplinama.

Matrične su operacije same po sebi jednostavne. Međutim, kada se radi o velikoj količini podataka, što povlači rad s matricama velikih dimenzija, ručno računanje postaje neefikasno, nepraktično i praktički neizvedivo. Zato se kod rješavanja takvih problema koristi snaga računala i različiti programski alati koji automatiziraju matrične operacije. Jedan od tih alata je Python.

Python je jedan od najpopularnijih i najjednostavnijih programskih jezika i omogućuje implementaciju određenih operacija pomoću njegovih ugrađenih objekata: lista. Međutim, Python zbog svoje jednostavnosti i visoke dinamičnosti gubi na efikasnosti i brzini, što je problematično kada je potrebno obaviti puno matričnih operacija nad matricama velikih dimenzija u kratkom vremenu. Na primjer, u računlnoj grafici gdje se matrične operacije mogu izvršavati nad cijelim zaslonom, što koristi matrice veličine rezolucije (npr. 1920x1080), te još dodatne dimenzije za boju. Iz tog razloga napravljen je modul NumPy koji koristi brzinu jezika *C*, ali dinamičnost i jednostavnost Python-a. NumPy-eva glavna komponenta su fiksna, n-dimenzionalna polja ostvarena pomoću *ndarray* objekata, koja omogućuju brzo i efikasno izvršavanje matričnih operacija, čak i ako radimo s velikom količinom podataka, tj. s matricama velikih dimenzija. Također, osim brzine, NumPy modul omogućuje računanje s matricama koje Pythonove liste ne pružaju.

NumPy je dio jezgrenih paketa SciPy-a koji je spoj različitih modula za efikasno znanstveno računanje, te uz SciPy nudi praktički sve operacije linearne algebre, već optimalno implementirane i spremne za uporabu. NumPy i SciPy su dva snažna Python paketa koja omogućuju upotrebu Pythona učinkovito u znanstvene svrhe.

Popis literature

[1] D. Bakić, *Linearna algebra*. Školska knjiga, 2008.

[2] E. Bressert, „ScyPy and NumPy", O'Reilly Media, 2013. adresa: https://books.google.hr/books?hl=hr&lr=&id=c-xzkDMDev0C&oi=fnd&pg=PR2&dq=NumPy&ots=Z6WTBUtbz9&sig=ywp0gnPs6jFESozwoKg1KQ1-u70&redir_esc=y#v=onepage&q=NumPy&f=false.

[3] B. Divjak i T. Hunjak, *Matematika za informatičare*. Tiva, Varaždin, 2004.

[4] EDUCBA, „Python Features". adresa: https://www.educba.com/python-features/.

[5] ——, „Uses of Python". adresa: https://www.educba.com/uses-of-python/.

[6] ——, „What is Python?". adresa: https://www.educba.com/what-is-python/.

[7] N. Elezović, *Linearna algebra*. Element, 1995.

[8] K. Horvatić, *Linearna algebra*. Golden marketing, 2004.

[9] M. Hruška, „Osnove programiranja, Python", Sveučilište u Zagrebu, 2018. adresa: https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d450_polaznik.pdf.

[10] M. Lutz, *Programming Python*. O'Reilly Media, 2001. adresa: https://books.google.hr/books?hl=hr&lr=&id=c8pV-TzyfBUC&oi=fnd&pg=PR11&dq=Python&ots=n4bI5OTUWW&sig=QY7CChWQGvAngFqqxkwlAns57Rg&redir_esc=y#v=onepage&q=Python&f=false.

[11] NumPy.org, „Array creation routine". adresa: https://numpy.org/doc/stable/reference/routines.array-creation.html#routines-array-creation.

[12] ——, „Indexing". adresa: https://numpy.org/doc/stable/reference/arrays.indexing.html.

[13] ——, „Linear algebra". adresa: https://numpy.org/doc/stable/reference/routines.linalg.html.

[14] ——, „Universal functions". adresa: https://numpy.org/doc/stable/reference/ufuncs.html#math-operations.

[15] S. O'Grady, „The RedMonk Programming Language Rankings", 2020. adresa: https://redmonk.com/sogrady/2020/02/28/language-rankings-1-20/.

[16] T. Oliphant, *A guid to NumPy*. Trelgol Publishing, 2006. adresa: https://ecs.wgtn.ac.nz/foswiki/pub/Support/ManualPagesAndDocumentation/numpybook.pdf.

[17] Python.org, „Python Success Stories". adresa: https://www.python.org/about/success/.

[18] ——, „What is Python? Executive Summary". adresa: https://www.python.org/doc/essays/blurb/.

[19] SciPy.org, „Linear algebra". adresa: https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.solve_sylvester.html.

[20] S. Shell, „An introduction to NumPy and SciPy". adresa: https://sites.engineering.ucsb.edu/~shell/che210d/numpy.pdf.

[21] Tutorialspoint, „NumPy-introduction". adresa: https://www.tutorialspoint.com/numpy/numpy_introduction.htm.

[22] J. VanderPlas, *Python Data Science Handbook*. O'Reilly Media, 2016. adresa: https://jakevdp.github.io/PythonDataScienceHandbook/.

Popis slika

1. Python logo (Izvor: Python.org) 15
2. Broadcasting (Izvor: **2016**, VanderPlas) 23
3. *Broadcasting* greška 23