Zápočtový program na NMIN112

Vedoucí cvičení: Mgr. Jan Střeleček

Student: Natálie Horenská

Program: operace s maticemi

Uživatelská dokumentace

Vážení uživatelé,

tento program vám umožní provádět různé operace s maticemi, včetně násobení matic, výpočtu determinantu, určení hodnosti matice a mnoho dalšího. Následující seznam popisuje jednotlivé operace, které můžete vykonávat pomocí tohoto programu:

1. Násobení matic

Tuto operaci spustíte zadáním "násobeni" do programu. Program vás požádá, abyste zadali dvě matice, které chcete násobit a následně vám zobrazí výsledek násobení těchto dvou matic.

2. Determinant matice

Tuto operaci spustíte zadáním "determinant" do programu. Program vás požádá, abyste zadali matici, pro kterou chcete vypočítat determinant a následně zobrazí vypočtený determinant matice.

3. Hodnost matice

Tuto operaci spustíte zadáním "hodnost" do programu. Program vás požádá, abyste zadali matici, pro kterou chcete určit hodnost a následně vám zobrazí hodnost matice.

4. Inverzní matice

Tuto operaci spustíte zadáním "inverze" do programu. Program vás požádá, abyste zadali matici, pro kterou chcete vypočítat inverzi a následně vám zobrazí inverzi matice.

5. Jádro matice

Tuto operaci spustíte zadáním "jadro" do programu. Program vás požádá, abyste zadali matici, pro kterou chcete najít jádro a následně vám zobrazí bázi jádra matice.

6. Vlastní čísla matice 2x2

Tuto operaci spustíte zadáním "vlastni cisla" do programu. Program vás požádá, abyste zadali matici, pro kterou chcete vypočítat vlastní čísla a následně vám zobrazí vypočetená vlastní čísla matice.

7. Vlastní vektory matice 2x2

Tuto operaci spustíte zadáním "vlastni vektory" do programu. Program vás požádá, abyste zadali matici, pro kterou chcete určit vlastní vektroy a následně vám zobrazí vypočtené vlastní vektory.

8. Diagonalizace matice 2x2

Tuto operaci spustíte zadáním "diagonalizace" do programu. Program vás požádá, abyste zadali matici, kterou chcete diagonalizovat. Program vám následně zobrazí matici vlastních vektorů, diagonální matici s vlastními čísly a inverzní matici vlastních vektorů.

9. Moore_penroseova pseudoinverze matice

Tuto operaci spustíte zadáním "pseudoinverze" do programu. Program vás požádá, abyste zadali matici, pro kterou chcete určit pseudoinverzi a následně vám zobrazí pseudoinverzi matice.

10. Řešení soustavy lineárních rovnic

Tuto operaci spustíte zadáním "reseni slr" do programu. Program vás požádá, abyste zadali matici soustavy rovnic a pravou stranu této soustavy a následně vám zobrazí řešení soustavy.

11. Řešení soustavy s více proměnnými

Tuto operaci spustíte zadáním "reseni slr 2" do programu. Program vás požádá, abyste zadali matici soustavy rovnic a pravou stranu této soustavy. Program vám následně zobrazí bázi řešení soustavy.

12. Metoda nejmenších čtverců

Tuto operaci spustíte zadáním "metoda ctvercu" do programu. Program vás požádá, abyste zadali matici soustavy rovnic a pravou stranu této soustavy. Následně vám zobrazí řešení soustavy lineárních rovnic pomocí metody nejmenších čtverců.

13. Násobení řídkých matic

Tuto operaci spustíte zadáním "nasobeni rm" do programu. Program vás požádá, abyste zadali dvě řídké matice, které chcete násobit, a následně vám zobrazí výsledek násobení těchto dvou řídkých matic.

14. Determinant řídké matice

Tuto operaci spustíte zadáním "determinant rm" do programu. Program vás požádá, abyste zadali řídkou matici, pro kterou chcete vypočítat determinant. Následně vám zobrazí vypočtený determinant řídké matice.

15. Hodnost řídké matice

Tuto operaci spustíte zadáním "hodnost rm" do programu. Program vás požádá, abyste zadali řídkou matici, pro kterou chcete určit hodnost a následně zobrazí hodnost řídké matice.

Jak používat program

1. Spusťte program v Pythonu.
2. Program vás bude vyzívat k zadání operace, kterou chcete provést. Zadejte název operace podle výše uvedeného seznamu (například "násobeni" nebo "determinant").
3. Program vás požádá o další vstupy, jako jsou matice, vektory nebo čísla, v závislosti na zvolené operaci. Zadejte tyto hodnoty podle pokynů programu:

   Pro zadání matice postupujte takto: Pro každý řádek matice vypište čísla oddělená mezerami. Oddělte jednotlivé řádky stiskem klávesy "Enter".

   Pro zadání pravé strany soustavy lineárních rovnic stačí tyto čísla napsat za sebe na řádek a oddělit mezerami.

4. Program vám zobrazí výsledek operace.

Programátorská dokumentace

Program je navržen jako jednoduchá aplikace napsaná v jazyce Python. Je rozdělen do několika funkcí, které provádějí specifické operace lineární algebry. Uživatel má možnost volit, kterou operaci chce provést, a poté zadává vstupní data prostřednictvím standardního vstupu (klávesnice). V celém programu jsou matice reprezentovány jako seznamy seznamů. V celém programu jsou datové struktury jako seznamy seznamů, slovníky a datové typy jako integer a float požívány pro reprezentaci matic a výsledků výpočtů.

Funkce pro operace s maticemi

I. *Funkce* **`read_matrix()`**

Načte číslo, které udá počet řádků matice, poté čte jednotlivé řádky matice, kde uživatel zadá čísla oddělená mezerami. Výsledkem je matice ve formě sezamu seznamů.

II. *Funkce* **`print_matrix(matrix)`**

Tato funkce vytiskne matici na výstup. Prochází řádky matice a vypisuje jednotlivé prvky oddělené mezerami.

III. *Funkce* **`gauss_elimination(matrix = None)`**

Aplikuje Gaussovskou eliminaci na matici pro operace jako výpočet determinantu, výpočet inverze matice a řešení soustav lineárních rovnic.

Vstup je matice. Algoritmus spočívá v postupném eliminování prvků pod diagonálou, aby se dostala matice do odstupňovaného tvaru. Poté změní hodnoty pivotů, tedy prvních nenulových prvků v každém řádku na 1 a hodnoty nad pivotem vynuluje. Výstupem je matice v odstupňovaném tvaru.

IV. *Funkce* **`matrix_multiplication(A = None, B = None)`**

Tato funkce načte dvě matice ze vstupu a výstupem je jejich součin. První dvě smyčky (i a j) procházejí řádky a sloupce výsledné matice, zatímco vnitřní smyčka (k) provádí skutečné násobení a sčítání.

Ošetření případných chyb:
ValueError: Matice nemaji kompatibilni rozmery pro nasobeni.

V. *Funnkce* **`determinan(matrix = None)`**

Vstupem je čtvercová matice a výstupem je hodnota int nebo float, určující determinant matice.

Funkce determinant používá rekurzivní expanzi podle menších hodnot, známou jako Laplaceův rozvoj (podle sloupce). Hlavní logika spočívá v tom, že se determinant vypočítá rozkladem matice na menší podmatice (menší o jednu řadu a jeden sloupec) a následně se rekurzivně spočítá determinant takto získaných menších matic.

Ošetření případných chyb:
ValueError: Matice neni ctvercova.

VI. *Funkce* **`rank(matrix = None)`**

Vstupem je matice a výstupem je hodnota int udávající hodnost matice.

Funkce rank() vypočítá hodnost matice, tzn. kolik nezávislých řádků nebo sloupců matice obsahuje. Za použití Gaussovy eliminace redukuje matici do odstupňované formy. V této podobě se nulové řádky nacházejí na konci matice. Výsledná hodnost je počet nenulových řádků.

VII. *Funkce* **`*inverse(matrix = None)*`**

Vstupem je matice, kterou chceme invertovat. Výstupem je inverzní matice. Je zde použita Gaussova-Jordanova eliminace. Proces zahrnuje vytvoření rozšířené matice kombinací původní a jednotkové matice a postupné transformování původní části matice do jednotkové, zatímco jednotková část se mění na inverzní matici.

Ošetření případných chyb:
ValueError: Matice není čtvercová.
ValueError: Matice je singulární a nemá inverzi.

VIII. *Fukce* **`*null_space(matrix = None)*`**

Vstupem je matice, ke které hledáme jádro a výstupem báze jádra matice. Nulový prostor je definován jako množina všech vektorů, které po vynásobení maticí dávají nulový vektor.

Hlavní algoritmy a datové struktury:
Použití Gaussovy eliminace k nalezení redukovaného řádkového echelonového tvaru (RREF) matice.
Identifikace pivotních a nepivotních sloupců.

Ošetření případných chyb:
ValueError: Matice nemá žádný řádek.

IX. *Funkce* **`*eigen_values(matrix = None)*`**

Vstupem je 2x2 matice, pro kterou chceme najít vlastní čísla a výstupem je seznam vlastních čísel, tzn. hodnot, které umožnují, aby matice mohla být diagonalizována, a to vytvořením a řešením charakteristického polynomu. Aktuální implementace je omezena na matice 2x2 a vypočítává vlastní čísla přímo z determinantu.

Ošetření případných chyb:
ValueError: Matice neni ctvercova.

X. *Funkce* **`*eigen_vectors(matrix = None)*`**

Vstupem je matice 2x2, pro kterou chceme najít vlastní vektory a výstupem je seznam vlastních vektorů matice, tzn. matice vektorů, které se při násobení maticí "natáhnou" nebo "stlačí" pouze v jednom směru. Výpočet je založen na řešení maticové rovnice kde 'A' je matice, λ je vlastní číslo a 'v' je hledaný vlastní vektor.

Ošetření případných chyb:
ValueError: Matice není čtvercová.

XI. *Funkce* **`*diagonalization(matrix = None)*`**

Vstupem je čtvercová matice, kterou chceme diagonalizovat. Funkce diagonalization() provede diagonalizaci dané čtvercové matice a vrací tři matice: matici vlastních vektoru P, diagonální matici D s vlastními čísly a inverzní matici p_inv.

Hlavní algoritmy a datové struktury:
Výpočet vlastních čísel a vlastních vektorů matice.
Výpočet inverzní matice.

Případné ošetření chyb:
ValueError: Matice není čtvercová.

XII. *Funkce* **`transpose(matrix)`**

Vstupem je matice, kterou chceme transponovat a výstupem transponovaná matice.

XIII. *Funkce* ***`moore_penrose_pseudoinverse(matrix = None)`***

Vstupem je matice, pro kterou chceme zjistit pseudoinverzní matici a výstupem je daná pseudoinverzní matice, která se používá pro řešení problému regrese, kdy tradiční inverzní matice neexistuje.

Hlavní algoritmy a datové struktury:
Výpočet pseudoinverzní matice využívá násobení matic a inverzní matici.

Ošetření případných chyb:
ValueError: Matice nemá žádný řádek nebo sloupec.

Funkce pro řešení soustav lineárních rovnic

XIV. *Funkce* ***`gaussian_elimination_solve(A = None, b = None)`***

Vstupem je matice soustavy lineárních rovnic a vektor pravých stran. Funkce gaussian_elimination_solve() řeší soustavu lineárních rovnic pomocí Gaussovy eliminace. Nejprve provádí přímý průchod, který konvertuje matici na horní trojúhelníkovou formu. Poté se provádí zpětný průchod, který vypočítá výsledné hodnoty proměnných.

Hlavní algoritmy a datové struktury:
Částečné pivotování pro výběr řádku s největším prvkem jako pivot.
Rozšíření matice pro uložení pravých stran rovnic.

Ošetření případných chyb:
ValueError: Matice A a vektor b nemaji kompatibilni rozmery.
ValueError: Matice A neni ctvercova.
ValueError: Matice A je singularni.

XV. *Funkce* ***`multidimensional_solution()`***

Funkce `multidimensional_solution()` hledá bázi pro řešení lineárních soustav, které mají více řešení. Vstupem funkce je matice soustavy a pravá strana soustavy. Identifikuje pivotní sloupce pomocí Gaussovy eliminace na rozšířené matici a identifikuje ne-pivotní sloupce. Vytváří bázi pro více řešení soustavy z ne-pivotních sloupců a pivotních sloupců. Funkce vrací bázi všech řešení soustavy.

XVI. *Funkce* ***`matrix_vector_multiply(A, v)`***

Vstupem je matice A a vektor v. Výstupem je vektor, který je součinem matice A s vektorem v.

XVII. *Funkce* ***`least_squares_solution(A = None, b = None)`***

Vstupem je matice soustavy lineárních rovnic, které mají více rovnic než proměnných a vektor pravé strany rovnice. Výstupem je vektor určující řešení pomoci metody nejmenších čtverců. Funkce využívá transponování matice, násobení matic a řešení soustav pomocí Gaussovy eliminace.

Ošetření případných chyb:
ValueError: Matice A a vektor b nemaji kompatibilni rozmery.

Funkce pro řídké matice (Sparse Matrices)

Třída SparseMatrices

Třída SparseMatrix poskytuje metody pro práci s řídkými maticemi. Výhodou ukládání v řídkém formátu je úspora paměti, protože ukládáme pouze nenulové prvky a jejich indexy. Obsahuje tyto metody:

XVIII. *Funkce* **`__init__(self, matrix = None)`**

Inicializuje řídkou matici z dané plné matice. Řídké matice jsou tedy v celé třídě reprezentovány slovníkem.

XIX. *Funkce* **`add_element(self, i, j, element)`**

Přidá prvek na danou pozici [i, j] v řídké matici. Vstupem jsou indexy prvku matice a daný prvek.

XX. *Funkce* **`access(self, i, j)`**

Vstupem jsou dvě čísla určující po řadě index řádku a sloupce v matici a výstupem je prvek na této pozici.

XXI. *Funkce* **`convert_to_full_matrix(self)`**

Výstupem této funkce je plná matice.

XXII. *Funkce* **`multiply_sparse(self = None, sparse1 = None, sparse2 = None)`**

Tato funkce provádí násobení dvou řídkých matic. Algoritmus využívá slovníky pro rychlý přístup k nenulovým prvkům. Výstupem je součin těchto matic.

XXIII. *Funkce* **`det_sparse(self, matrix = None)`**

Vstupem této funkce je řídká matice. Výstupem hodnota jejího determinantu. Funkce opět využívá Laplaceův rozvoj podle sloupce.

XXIV. *Funkce* **`rank_sparse(self, matrix = None)`**

Vstupem je matice a výstupem její hodnost. Funkce hledá hodnost matice tak, že spočítá počet nenulových řádků a počet nenulových sloupců matice. Hodnost matice je pak určena jako menší z těchto dvou počtů.

XXV. *Funkce* **`main()`**

Tato hlavní funkce programu umožňuje uživateli provádět různé operace s maticemi. Uživatel má možnost vybrat z několika operací, jako je násobení matic, výpočet determinantu, výpočet hodnosti matice, inverze matice a další. Tato funkce zpracovává vstup od uživatele, provádí požadovanou operaci a výsledky vypisuje na standardní výstup.