1. DOMAĆA ZADAĆA – AK. GOD. 2019/20

Domaća zadaća

Implementirajte razred "Matrica" (ili neku sličnu programsku cjelinu) koji omogućava jednostavnije rukovanje objektima tipa dvodimenzionalne matrice. Za tu namjenu ostvarite osnovne elemente:

- podatke o dimenzijama matrice i dinamički zauzete elemente matrice tipa *double*. Smještaj sadržaja matrice u memoriji organizirati po izboru, uz dinamičko zauzimanje memorije.
- potrebne konstruktore i destruktore, po potrebi metode za mijenjanje dimenzija matrice.
- operator ili metoda pridruživanja (funkcionalnost `A = B`).
- metoda koja čita matricu iz tekstne datoteke u kojoj je matrica smještena tako da jedan redak matrice odgovara jednom retku u datoteci. Na osnovu broja elemenata u prvom retku određuje se broj stupaca matrice, a redaka ima koliko ima i redaka u datoteci. Brojevi u jednom retku mogu biti odvojeni razmakom ili tabulatorom. Vidi primjer ulazne datoteke na kraju!
- metoda koja ispisuje matricu u datoteku u istom formatu, te metodu koja ispisuje matricu na ekran.
- metode za postavljanje i dohvat jednog elementa matrice. Može biti implementirano i istom metodom, uz korištenje funkcije koja vraća referencu i/ili nadgradnjom operatora [] (C++) ili nekim drugim postupkom po želji.
- metode za zbrajanje, oduzimanje, množenje i transponiranje matrica, te metode koje obavljaju C operatore "+=" i "-=". Preporuča se da budu implementirane kao nadgrađeni operatori.
- metoda za množenje matrice sa skalarom. Može biti ostvarena i kao operator.
- operator == za usporedbu matrica.

Primjer uporabe uz nadgrađene operatore:

```
Matrica A("A.txt"), B("B.txt"), C;
C = ~A;
C += A * 0.5 * B * (A – 2 * B);
double x = C[0][0];
C[1][1] = x;
```

Primjer ulazne datoteke matrice (elementi matrice 4x3 po retcima):

```
12.5 3.0 9 2
4 5 6 7
8 9 10 11
```

Osim gore navedenih, potrebno je ostvariti i sljedeće elemente:

- metode koje izvode **supstituciju unaprijed** i **supstituciju unatrag**. Metode neka kao ulazni parametar (slobodni vektor s desne strane sustava) primaju vektor čija je duljina jednaka dimenziji kvadratne matrice, a koji je i sam ostvaren objektom razreda Matrica. Također trebaju vraćati vektor kao objekt tipa Matrica (u kojemu će biti upisano rješenje odgovarajućeg postupka).
- metodu (metode) koje izvode **LU i LUP dekompoziciju** (kvadratne) matrice koristeći isti memorijski prostor za spremanje rezultantnih matrica L i U. Izbor odgovarajuće metode može se riješiti nekim dodatnim kontrolnim parametrom. Obratiti pažnju na mogućnost pojave nule kao stožernog (pivot) elementa (metode moraju imati neki mehanizam otkrivanja i prijave pogreške korisniku!).
- metoda za računanje **inverzije** kvadratne matrice uz pomoć LUP dekompozicije. Inverzna matrica se računa stupac po stupac, s jednom LUP dekompozicijom i *n* supstitucija unaprijed i unatrag, kako je pokazano na predavanjima (*u skripti: str. 3-30*). Metoda se može definirati kao unarni operator nad

matricom. Posebnu pažnju obratiti na slučaj kada je matrica singularna (pojava nule za stožerni element).

- metoda za računanje **determinante** matrice. Ako se matrica A rastavi na L, U i P matrice, onda se determinanta matrice A može izračunati kao umnožak determinanti pojedinih matrica: $\det(A) = \det(P^{-1})\det(L)\det(U)$. Pri tome, determinanta permutacijske matrice jednaka je 1 ako nije napravljena niti jedna permutacija redaka ili parni broj permutacija, odnosno -1 ako je napravljen neparan broj permutacija redaka. S druge strane, determinanta trokutastih matrica jednaka je umnošku elemenata na dijagonali. Dakle, determinantu matrice A možemo onda izračunati kao: $A = (-1)^S \left(\prod_{i=1}^{n} l_{ii} \right) \left(\prod_{i=1}^{n} u_{ii} \right)$, pri čemu S predstavlja broj permutacija koji je napravljen, l element L matrice i u element U matrice.
- nije obvezatno, ali može pomoći: metode koje manipuliraju stupcima matrice (postavljaju i vraćaju određeni stupac matrice pomoću objekta istoga tipa), jer se operacije te vrste često koriste u opisanim postupcima.

Važno: obratite pažnju na situacije u kojima se kao stožerni element može pojaviti nula ili neka jako mala vrijednost (*na bilo kojem mjestu u vašoj implementaciji!*). Program treba 'preživjeti' pojavu takvih okolnosti i prijaviti grešku. U tom slučaju ne smije se ispisati rješenje koje matematički nema smisla.

Laboratorijska vježba

1. Kakva treba biti usporedba *double* varijabli kako bi uspoređivanje dalo očekivane rezultate? Isprobajte operator == s elementima matrice kao necijelim brojevima, pomnožite i podijelite sa realnim brojem i usporedite s originalom.
2. Riješite sustav zadan matricama u nastavku. Odredite može li se riješiti LU odnosno LUP dekompozicijom:

$$\begin{bmatrix} 3 & 9 & 6 \\ 4 & 12 & 12 \\ 1 & -1 & 1 \end{bmatrix} \underline{x} = \begin{bmatrix} 12 \\ 12 \\ 1 \end{bmatrix}$$

3. Zadanu matricu rastavite na LU odnosno LUP. Ako je ovom matricom predstavljen sustav jednadžbi, može li se sustav riješiti? (sami definirajte slobodni vektor)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

4. Zadani sustav riješite LU te LUP dekompozicijom. Objasnite razliku u rješenjima! (očituje se prilikom uporabe *double* varijabli)

$$\begin{bmatrix} 0.000001 & 3000000 & 2000000 \\ 1000000 & 2000000 & 3000000 \\ 2000000 & 1000000 & 2000000 \end{bmatrix} \underline{x} = \begin{bmatrix} 12000000.000001 \\ 14000000 \\ 10000000 \end{bmatrix}$$

5. Zadani sustav riješite odgovarajućom metodom. Objasnite razliku između dobivenog i točnog rješenja.

$$\begin{bmatrix} 0 & 1 & 2 \\ 2 & 0 & 3 \\ 3 & 5 & 1 \end{bmatrix} \underline{x} = \begin{bmatrix} 6 \\ 9 \\ 3 \end{bmatrix}$$

6. Rješavanje sljedećeg sustava *moglo* bi zadati problema vašoj implementaciji. O čemu to ovisi? Kako je moguće izbjeći ovaj problem, transformacijom zadanog sustava tako da rješenje ostane nepromijenjeno? (*Napomena:* postavite vrijednost epsilona za ovaj primjer na 10^{-6})

$$\begin{bmatrix} 4000000000 & 1000000000 & 3000000000 \\ 4 & 2 & 7 \\ 0.0000000003 & 0.0000000005 & 0.0000000002 \end{bmatrix} \underline{x} = \begin{bmatrix} 9000000000 \\ 15 \\ 0.0000000015 \end{bmatrix}$$

7. Korištenjem LUP dekompozicije izračunajte inverz zadane matrice te ispišite dobiveni rezultat:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

8. Korištenjem LUP dekompozicije izračunajte inverz zadane matrice te ispišite dobiveni rezultat:

$$\begin{bmatrix} 4 & -5 & -2 \\ 5 & -6 & -2 \\ -8 & 9 & 3 \end{bmatrix}$$

9. Korištenjem LUP dekompozicije izračunajte determinantu matrice:

$$\begin{bmatrix} 4 & -5 & -2 \\ 5 & -6 & -2 \\ -8 & 9 & 3 \end{bmatrix}$$

10. Korištenjem LUP dekompozicije izračunajte determinantu matrice:

$$\begin{bmatrix} 3 & 9 & 6 \\ 4 & 12 & 12 \\ 1 & -1 & 1 \end{bmatrix}$$

Napomena

Prije predaje laboratorijske vježbe pripremite datoteke s rezultatima svih prethodnih zadataka. Također pripremite i **glavni program** čijim će se pokretanjem odjednom pokrenuti svi zadaci i ispisati rezultati za svaki od zadataka.

Demonstracija funkcionalnosti u MATLAB-u

Ovaj dio vježbe izvodi se na predavanjima.

Potrebno je napisati skriptu za programski paket MATLAB koja će učitati matricu sustava i slobodni vektor (iz istih datoteka kao programska implementacija), uz pomoć LUP dekompozicije pronaći i ispisati matrice L, U i matricu permutacija P, te međurezultat y. Također naći rješenje sustava te invertiranu matricu sustava. U skripti se (između ostalih) preporučuje koristiti sljedeće ugrađene funkcije: *lu*, *inv* te lijevo dijeljenje (\ ). Usporedite dobiveno rješenje sa onim koje ste dobili vašim programom.

Odabrana poglavlja iz MATLAB Helpa:

- *Mathematics: Matrices and Linear Algebra: Solving Linear Systems of Equations: Computational Considerations*
- *Mathematics: Matrices and Linear Algebra: Solving Linear Systems of Equations: Square Systems*
- *Mathematics: Matrices and Linear Algebra: LU Factorization*
- *MATLAB Functions: lu*