4. feladat

Kincskeresés vektorosan

Ismerősünk ismét segítségünket kérte. Nagyapja naplójában talált egy feljegyzést arról, hogy hol van az ő nagyapjától kapott kincs elrejtve: "Lépj előre x_0 lépést, jobbra fordulva x_1 lépést, majd balra fordulva x_2 lépést, és ezt ismételd!" Azt sejti, hogy honnan kell indulni, csakhogy nagyapja a hely eléréséhez szükséges lépések számából álló x vektort beszorozta egy A mátrixxal és az eredményül kapott b vektort írta le. Szerencsére eszébe jutott, hogy nagyapjával matekot is játszottak régebben, és elővette "Játékos matek" feliratú dossziéját. Ebben két megfelelő méretű mátrixot is talált, ezek egyike lehet, amit nagyapja használt.

Készítsünk C programot, mely megvalósítja a Gauss-Jordan-elimináció algoritmusát, vagyis megoldja az A x = b lineáris egyenletrendszert! Itt A egy $n \cdot n$ méretű adott mátrix, x és b pedig n elemű oszlopvektorok tetszőleges n értékkel (A és b adott, x az ismeretlenekből álló vektor).

Lépésenként bővítsük a programot az alábbi pontok szerint, és a beadott fájl nevében utaljunk rá, hogy melyik szinten működik, mégpedig main-a.c, main-b.c ... main-z.c. A főbb lépésekre, amelyekre az adott feladatrészhez szükség van, írjunk külön függvényeket: beolvasás, egy sor osztása egy számmal, egy sor számszorosának hozzáadása másik sorhoz, egy sor normálása, absz. értékbeli maximumkeresés egy sorban, ill. egy oszloprészben, ill. egy almátrixban, két sor felcserélése, stb. (ahogy azt az előadásanyag is sugallja).

A program a következő parancsargumentumokat használja ebben a sorrendben:

$$\text{A\_mátrix\_fájlja} \;\; \text{b\_vektor\_fájlja} \;\; n$$

A kincs és a pontszám eléréséhez a programnak az egyenletet teljesítő megoldást kell adnia az alábbi mátrixokkal és vektorokkal. Az első hármat csak tesztelésre szánjuk, azok egy egyenletrendszert teljesítő A,x,b tömbök. A másik három a feljegyzések közt talált mátrix és vektor, amikhez keressük a megoldásvektort. Természetesen lehet saját kis mátrixokkal is ellenőrizni a program helyes működését..

A3x3.dat, x3x1.dat, b3x1.dat, Beauty6x6.dat, Beast6x6.dat, b6x1.dat

a) Az A mátrixot és b vektort olvassuk be a parancsargumentumban megadott fájlokból! Ehhez lehetőleg egy függvényt kell írni, hiszen lehet a vektort beolvasni úgy, mint egy mátrixot, aminek a szélessége 1, vagy a mátrixot úgy, mint egy $n \cdot n$ hosszúságú vektort! E fájlok fejléce, ami '#' karakterrel kezdődő sorokból áll, a mátrix létrehozásával kapcsolatos adatokat tartalmaz. Ezekkel nincs más teendőnk, mint egymás után kiírni az eredményfájlba, aminek output.dat legyen a neve. Az adatokat a fájlok többi sorában találjuk, soronként legfeljebb egyet. De lehetnek a fájlban üres sorok is. Pl. gyakran használt formátum, hogy a mátrix egy sorához tartozó elemek a fájlban egymás utáni sorokban vannak, de egy üres sor választja el őket a következő mátrixsor elemeitől.

Látszik ebből, hogy a beolvasás során minden sornál el kell dönteni, hogy '#'-tel kezdődik-e, ill. üres sor-e, és annak megfelelően dolgozni fel a sor tartalmát.

Segítség a beolvasáshoz: Ha a szokott fscanf("%s",s) utasítással olvasnánk be a fájlból, akkor a fejléc részben található első spaceig olvasna csak, következő utasításra olvasna tovább. Így nehéz lenne eldönteni, hogy a folytatás még a fejléc része, vagy pedig adatokat tartalmazó sor. Ezért javasolt az fgets(s,length,f) utasítás használatát. (Persze más megoldások is lehetségesek.) Ez beolvas az f filepointerrel megnyitott fájlból egy sort az s stringbe, legfeljebb length hosszúságban (ezt most vehetjük 256-nak, mert nem lesznek ennél hosszabbak a sorok). Az s stringet ezután tudjuk kezelni attól függően, hogy milyen karakterrel kezdődik. Jó tudni még, hogy az fgets beolvassa a sort lezáró karaktereket is a sztringbe, így üres sor esetén a sztringet tároló tömb első karaktere nem a null karakter lesz, hanem Dos/Windows formátumú fájl esetén \r lesz, linuxos esetén pedig \n. (A)

b) Írjuk meg a Gauss-Jordan-elimináció algoritmusát pivotálás nélkül! A Beauty6x6.dat használatánál, ill kisebb mátrixok esetén általában megbízható eredményt kapunk, azt lehet az A x szorzás elvégzésével ellenőrizni (pl. pythonban, de ezt nem kell beadni).

Tanulságos viszont, hogy vannak furcsán viselkedő, u.n. rosszul kondícionált mátrixok, ilyen a Beast6x6.dat-ban tárolt is. Ezek esetében az eredmény az egymástól kicsit eltérő (de helyes) programok esetében egymástól és a valódi kiindulási x-től lényegesen eltérő eredményt ad. Ugyanakkor mindegyik eredményre az A x szorzat nagy pontossággal visszaadja a b vektort. Próbáljuk ki és írjuk be a readme.txt fájlba az x-re kapott eredmény első komponensét! (De nem szükséges emiatt a programon változtatni.) Összehasonlításul megadjuk, hogy egy extra pontos eljárás az első három elemre azt adta ki, hogy 60, 50, 40. (A)

c) Végezze el a program a számolást a LAPACK csomag használatával is! A kétféle számolás egymás után lefuthat, hogy ne kelljen külön programváltozatot írni. Az egyenletrendszer megoldásához javasolt a dgesv függvény használata. Sorfolytonosan tárolt mátrix esetében a szokásos egyszerű tárolás esetén így tudjuk meghívni:

```
info = LAPACKE_dgesv(LAPACK_ROW_MAJOR, n, 1, mx, n, ipiv, v, 1);
```

ahol mx a mátrix pointere, v pedig a vektoré, ami kezdetben a b vektort kell tartalmazza, és a függvény lefutása után a megoldás vektort kapjuk meg benne. n a mátrix és vektor közös mérete. ipiv egy n elemű int tömb pointere kell legyen (a lefutás után kiegészítű információkat tartalmaz)

A lapack csomagot mindenki telepítheti saját gépére, ill. lehet használni a kooplex rendszerben. A kooplexen való használat esetén ahhoz, hogy elérjük a programból a kívánt függvényt a programban a

```
#include <lapacke.h>
```

sorra van szükség, a fordításkor pedig

```
gcc -Wall main-c.c -llapacke
```

módon kell a csomagot megadni. (A)

d) Bővítsük a programunkban a saját megoldófüggvényünket részleges pivotálással (azaz a Gauss-eliminációs lépések előtt végezzük el a sorok normálását a maximális abszolút értékű elemükkel, másrészt a lineárkombinációkhoz válasszunk az oszlopokból legmegfelelőbb pivot elemet). Az elimináció során a program detektálja, ha a mátrix numerikusan szinguláris, és írja ki, hogy melyik oszlopban ütközött problémába! Ez esetben is írjuk be a readme.txt fájlba a Beast mátrix esetében az x-re kapott eredmény első komponensét! (T)

e) Szorgalmi feladat: Gyártsunk nagyméretű, véletlen elemű mátrixokat és vektort, és hasonlítsuk össze a saját és a LAPACK változat sebességét, illetve skálázását! Nem külön programot kell írni, hanem hogyha az első parancsargumentumban megadott fájlnév "-" (vagy '-' karakterrel kezdődik), akkor beolvasás helyett a program a harmadik argumentumban megadott méretben hozzon létre mátrixot és vektort, és mérje meg a futásidőt. A skálázás vizsgálata esetén legyen a beolvasott n a legnagyobb vizsgált méret.