3. feladat

Kincskeresés vektorosan

Ismerősünk ismét segítségünket kérte. Nagyapja naplójában talált egy feljegyzést arról, hogy hol van egy 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 a nagyapja jegyzeteiből az is kiderült, hogy a hely eléréséhez szükséges lépések számából álló x vektort beszorozta egy $\mathbf{A}$ mátrixxal és az eredményül kapott b vektort írta le. Szerencsére eszébe jutott, hogy nagyapjával matekot is játszottak régebben, és talált is számítógépén egy "Játékos matek" feliratú mappát. Ebben volt két fájl, amelyekben mátrixelemeknek tűnő számok voltak, legalább is erre lehetett következtetni a fájlok elején lévő rövid szövegből. Ezen fájlok egyike lehet az, amit nagyapja használt.

Készítsünk C++ programot, mely megvalósítja a Gauss-Jordan-elimináció algoritmusát, vagyis megoldja az $\mathbf{A} \cdot x = b$ lineáris egyenletrendszert! Itt $\mathbf{A}$ egy $N \times N$ méretű adott mátrix, x és b pedig N elemű oszlopvektorok.

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 stb szerint. 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, abszolút értékbeli maximumkeresés egy sorban, ill. egy oszloprészben, ill. egy almátrixban, két sor felcserélése, stb. (ahogy azt az előadás anyaga is sugallja).

A program a következő parancsargumentumokat használja ebben a sorrendben:

$$\mathrm{A\_mátrix\_fájlja}\ \ \mathrm{b\_vektor\_fájlja}\ \ N$$

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 $\mathbf{A}$, 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) Gauss-Jordan-elimináció pivotálás nélkül

Írjuk meg a Gauss-Jordan-elimináció algoritmusát pivotálás nélkül és próbáljuk ki az algoritmust! A Beauty6x6.dat használatánál, ill kisebb mátrixok esetén általában megbízható eredményt kapunk. Ezt lehet ellenőrizni úgy, hogy miután kiszámoltuk az x megoldásvektort, az $\mathbf{A} \cdot x$ szorzás elvégzésével vissza kell kapnunk a b vektort (ezt az egyszerű számolást pl pythonban is meg lehet írni, de ezt nem kell beadni).

Tanulságos viszont, hogy vannak furcsán viselkedő, ú.n. rosszul kondicionált mátrixok, ilyen a Beast6x6.dat-ban tárolt is. Ezek esetében az egymástól kicsit eltérő (de helyes) programok

egymástól lényegesen eltérő x eredményvektort adhatnak. Ugyanakkor mindegyik eredményre az $\mathbf{A} \cdot x$ szorzat nagy pontossággal visszaadja a b vektort, tehát ilymódon nehéz ellenőrizni, hogy melyik program adja vissza pontosabban a valódi x vektort. Összehasonlításul megadjuk, hogy egy extra pontos eljárás az első három elemre azt adta ki, hogy $(60, 50, 40)$. Ha a Beauty mátrixxal a program jól működik, csak a Beasttel pontatlan, akkor nem szükséges megpróbálni ezen javítani. (A)

b) A LAPACK csomag használata

Ismerkedjünk meg a LAPACK numerikus lineáris algebra könyvtár használatával! Végezze el a program az előbbi számolást a LAPACK csomag használatával is, Ehhez a LAPACKE nevű csomagot használjuk, amely egy interfészt biztosít az eredetileg Fortran-ban írt rutinokhoz. Az egyenletrendszer megoldásához a *dgesv* függvény használatát javasoljuk.

A saját rutin és a LAPACK csomagot meghívó számolás egymás után lefuthat, hogy ne kelljen külön programváltozatot írni. $Array$-ban vagy $vector$-ban (javasolt: $vector$) sorfolytonosan tárolt mátrix esetében így tudjuk meghívni:

```
int info = LAPACKE_dgesv(LAPACK_ROW_MAJOR, N, nrhs, M.data(), ldM,
ipiv.data(), v.data(), ldv);
```

Itt N az egyenletrendszer sorainak száma, $nrhs = 1$ a jobb oldali oszlopvektorok száma, az M.data() arra utal, hogy a Lapacke függvénynek az együtthatókat tartalmazó mátrix pointerét (memóriacímét) kell átadni (a pointerekkel bővebben nem foglalkozunk). Hasonlóan, át kell adni a v vektor pointerét, amely vektor kezdetben a b vektort kell tartalmazza, és a függvény lefutása után a megoldást kapjuk meg benne. Továbbá, (esetünkben) $ldM = N$ és $ldv = 1$ az M mátrix és a v vektor *leading dimension*-je (bővebb magyarázat a dgesv függvény leírásában található). Végezetül, az $ipiv$ egy N elemű, int típusú tömb (a lefutás után kiegészítő információkat tartalmaz).

A LAPACKE csomagot mindenki telepítheti saját gépére, ill. lehet használni a k8plex rendszerben is. A k8plexen való használat esetén ahhoz, hogy elérjük a programból a kívánt függvényt a programban a

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

sorra van szükség. A program fordítását és a szükséges könyvtár linkelését pedig

```
g++ -Wall main.cpp -llapacke -o main
```

paranccsal lehet elvégezni. (A)

c) Fejléc kezelés

Bővítsük a programot olymódon, hogy kezelni tudjon egy egyszerű fejlécet az adatfájlban. Ez azt jelenti, hogy a beolvasás során figyelje, a beolvasott sor nem kezdődik-e '#' karakterrel. A '#' karakterrel kezdődő sorokat egyszerűen írja ki. Továbbá a program tudja kezelni, ha üres sorokat talál (lépjen tovább a beolvasás). Egy mátrix adatait ugyanis sokszor kényelmes úgy tárolni a fájlban, hogy a mátrix egyes soraiba tartozó adatok között van egy üres sor.

Kihasználhatjuk azt a tényt, hogy a kapott fájlokban soronként legfeljebb egy szám szerepel. A beolvasát tesztelhetjük a fentebb említett fájlok "-h.dat" végződésú változataival. (T)

d) Részleges pivotálás

Bővítsük a programunkat részleges pivotálással. A Gauss-eliminációs lépések előtt végezzük el a sorok normálását a maximális abszolút értékű elemükkel. Másrészt, az eliminációs lépések során a lineárkombinációkhoz válasszuk ki a legmegfelelőbb pivot elemet. Az elimináció során a programnak detektálnia kell, ha a mátrix numerikusan szinguláris, azaz a pivotelemnek választható elemek abszolút értékben kisebbek mint 10^{-8}! Írja ki, hogy melyik oszlopban ütközött problémába és a választható pivot elemek közül a legnagyobb abszolút értékűt! (T)

e) Szorgalmi

Gyártsunk nagyméretű, véletlen elemű mátrixokat és vektort, és hasonlítsuk össze a saját és a LAPACK változat sebességét, illetve skálázását! Skálázás alatt azt értjük, hogy hogyan függ a futási idő az lineáris egyenletek N számától. Ábrázoljuk a futási időt az N függvényében egy jupyter notebook-ban és próbáljunk hatványfüggvényt illeszteni az adatokra!

A programot úgy írjuk meg, hogy a bemeneti paramétereket kétféleképpen is tudja kezelni: a fentiekben már ismertetett módon, vagyis amikor két tömböt és ezek méretét adjuk neki át, valamint úgy is, hogy ha az első parancsargumentumban "-r" karaktersort találja, akkor csak két argumentumot várjon és a második argumentumban megadott méretben hozzon létre egy véletlen mátrixot és vektort.