# I. Bevezetés

A matematika, tárgyát és módszereit tekintve, sajátos tudomány, mely részben a többi tudomány által vizsgált, részben pedig a matematika „belső” fejlődéséből adódóan létrejött (felfedezett ill. feltalált) rendszereket, struktúrákat, azok absztrakt, közösen meglévő tulajdonságait vizsgálja. A numerikus analízis a matematikai - elsősorban, analitikus - problémák közelítő megoldásával foglalkozik. Az egyik legrégebbi matematikai írás az YBC 7289-es számú Babilóniai agyagtábla, amely 60-as számrendszerben jegyezte le a \sqrt{2}numerikus közelítését, ami egy egység négyzet átlójának hossza. A numerikus analízis folytatja ezt a hosszú tradíciót, de nem keres pontos megoldásokat, mert a gyakorlatban lehetetlen ilyeneket adni. A numerikus analízis közelítő megoldásokra törekszik, de úgy, hogy bizonyos elfogadható hibahatáron belül maradjanak.

A lineáris egyenletrendszer olyan többváltozós egyenletrendszer, ahol minden ismeretlen változó elsőfokon (azaz első hatványon) szerepel. Egy *m* egyenletből álló és *n* ismeretlent tartalmazó lineáris egyenletrendszer:

a_{11}x_{1}+a_{12}x_{2}+\dots+a_{1n}x_{n}=b_{1}

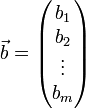
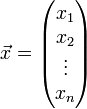
a_{21}x_{1}+a_{22}x_{2}+\dots+a_{2n}x_{n}=b_{2}

\vdots

a_{m1}x_{1}+a_{m2}x_{2}+\dots+a_{mn}x_{n}=b_{m}

Itt az *x*-ek az ismeretlenek, az *a*-k az ismeretlenek együtthatói, és a *b*-k az egyenletek konstansai. A lineáris egyenletrendszert felírhatjuk mátrix és vektorok kombinációjaként, melynek mátrixa egy olyan *m*×*n*-es mátrix, amely a lineáris egyenletrendszer együtthatóit tartalmazza. Az előbbi egyenletrendszer mátrixa:

A=
\begin{bmatrix}
a_{11}&a_{12}&\dots&a_{1n}\\
a_{21}&a_{22}&\dots&a_{2n}\\
\vdots&\vdots&\ddots&\vdots\\
a_{m1}&a_{m2}&\dots&a_{mn}
\end{bmatrix}


Ha bevezetjük a és az jelöléseket, akkor a lineáris egyenletrendszer a következő rövid alakban írható fel:

A\vec{x}=\vec{b}.

Az *A* mátrix és az \vec{x}vektor szorzata formálisan éppen a kívánt egyenleteket adja.

A lineáris és nem lineáris egyenletrendszerek numerikus megoldására vagy azok közelítésére sok algoritmust és módszert dolgoztak ki, és ezeket sokrétűen felhasználják a tudomány szinte minden területén.

A szakdolgozatom témáját keresve találkoztam Dr. Furtenbacher Tiborral és Prof. Dr. Császár Attilával az Eötvös Loránd Tudományegyetem Kémia Intézetének munkatársaival, akik a kvantumkémia egy érdekes ágával, az aktív kísérleti rezgési-forgási energiaszintek vizsgálatával foglalkoznak. Az általuk kitalált MARVEL eljárás nagyméretű, ritka mátrixokkal leírható egyenletrendszerek megoldásának közelítését is magában foglalja.

Ennek az eljárásnak a felületes megismerése adta az irányt a szakdolgozatom témájához. A kiválasztás során viszont motivált az is, hogy egy olyan problémát válasszak, mely átlátható, jól megfogható, és később akár a tanári pályán is tovább fejleszthető.

A probléma két részből állt.

Az első, hogy hogyan ábrázoljuk ezeket a ritka mátrixokat, vektorokat, az alapvető műveleteket hogyan optimalizáljuk ezekre a struktúrákra. Sok megoldást találtam erre, végül egy, a témában íródott szakdolgozat alapján elindulva egy sajátos adatstruktúrát választottam.

A másik a módszerek implementálása ezekre az adattípusokra. Itt a jól megfogalmazott struktúrák miatt már könnyebb volt a dolgom, a nehézséget az adatok összehasonlíthatóságának, az eredmények megjeleníthetőségének megtervezése volt.

# II. Felhasználói dokumentáció

A program Windows operációs rendszerre készült JAVA nyelven, így futásához fel kell telepíteni a Java futtató környezetet (JRE). A használatához elengedhetetlen a számítógéphez rendelt billentyűzet és egér. MEMÓRIA!!!!!!!!

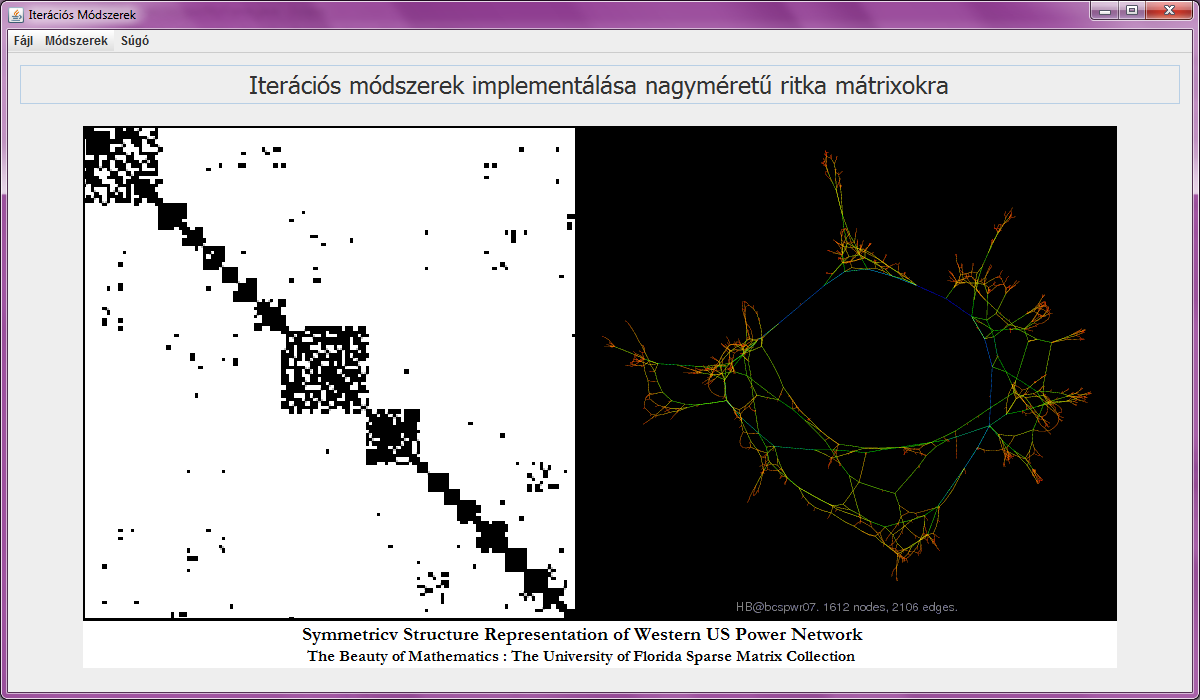
Ha a számítógépünk eleget tesz a követelményeknek, a program telepítés nélkül futtatható az Iteracios\_Modszerek.exe fájlt elinditva (duplán kattintva rá, esetleg parancssorból futtatva).

A program a felhasználó által magadott mátrix és vektor alapján reprezentálható lineáris egyenletrendszer megoldását keresi iterációs módszerekkel, melyekhez több x0 vektorból is kiindulhatunk. Az eredményeket összehasonlíthatjuk a különböző módszereknél, ezekből különböző grafikonokat kérhetünk le.

A program a futása során egy mátrixot, egy b és több x0 vektort képes tárolni, ezekre tudja alkalmazni az iterációs módszereket. Az adatokat megadhatjuk a felületen keresztül, fájlból beolvasva, illetve kérhetünk egy előre definiált tesztadatot, viszont fontos, hogy méretük megegyezzen, azaz ha a mátrix R(n x n)-es, akkor a vektorok is R(n) legyenek. Ezeket az adatokat akár kiírhatjuk fájlba is.

Az alkalmazást leginkább olyan felhasználókra terveztem, akik az iterációs módszereket tanulják, és szeretnének egyszerűen, gyorsan eredményeket elérni, esetleg összehasonlítani őket a nagyméretű környezetben, vagy akár egyéb tudományág problémáihoz felhasználni az eredményeket. Érdekes lehet esetleg a tanároknak is, akik ezzel a programmal reprezentálhatják a módszereket.

## A program általános leírása

Az alkalmazás egy főablakból áll, az ablak felső sorában egy menü helyezkedik el. Az elemeire kattintva legördülő menüben jelennek meg a menüpontok. Ez alatt jelenik meg a program megjelenítő felülete, melyen a megnyitáskor egy üdvözlő képernyő köszönt bennünket. Itt jelennek meg később a különböző panelek, melyeket a beolvasáshoz és kiíráshoz használ az alkalmazás.

1. ábra: Üdvözlő képernyő

### Fájl menü

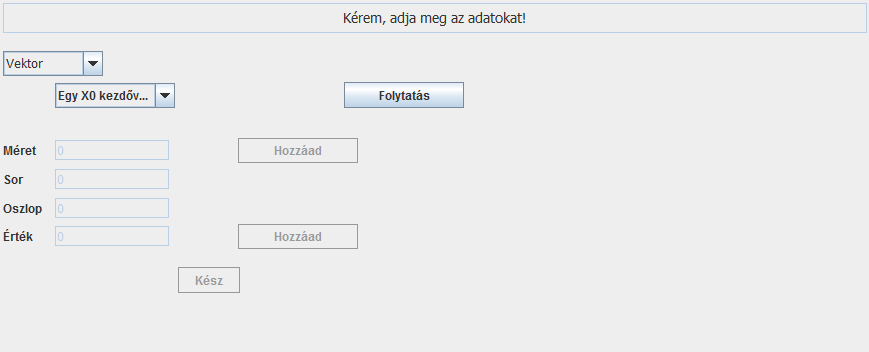
A fájl menüben találhatóak az alapvető funkciók, melyek nem kötődnek közvetlenül a módszerekhez. Az első menüpont az “Új folyamat”, ami törli az alkalmazásból az addig betöltött adatokat, és visszatér az üdvözlő képernyőre.

#### Beolvasás

A fájl menü “Beolvasás” pontja alatt három alpontot is találunk, melyek megnyitják a különböző adatbekérő paneleket.

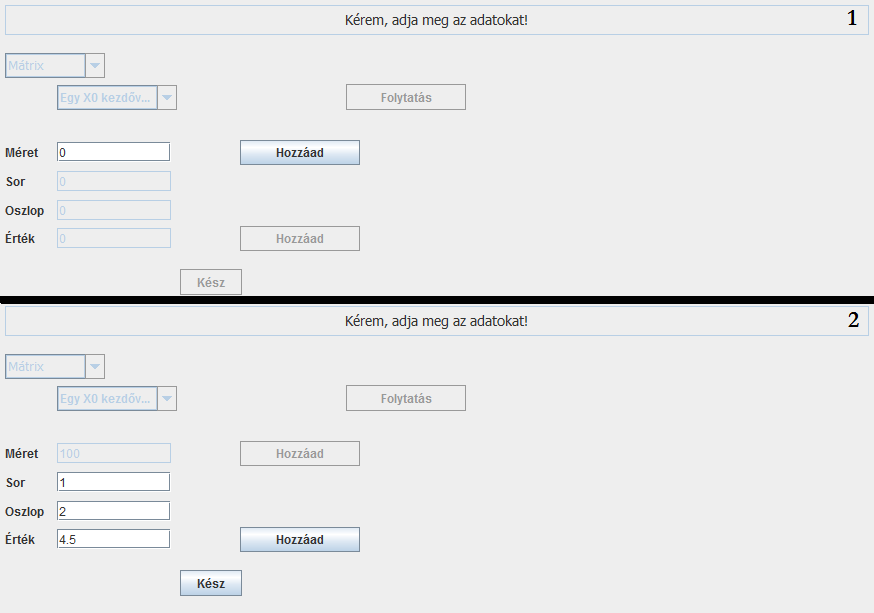
Értékek megadásával

A mátrixot és a vektorokat megadhatjuk a felületen keresztül. Először a legördülő menüből kiválasztjuk, hogy milyen adatot kívánunk megadni. Megadhatjuk a mátrixot vagy a vektorok közül a b jobboldali vektort, vagy az x0 kezdővektorok egyikét. A „Folytatás” gombra kattintva elérhetővé válik a méret megadására szolgáló mező.



2. ábra: Adat megadása a felületen - kezdőállapot

Mátrix megadása

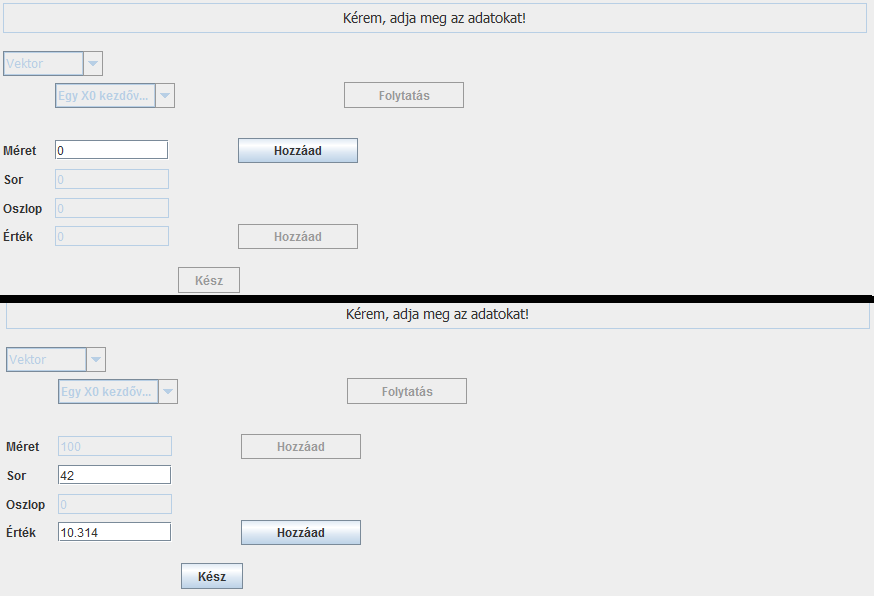
Egy négyzetes mátrixot szeretnénk megadni, a méret mezőbe az értéket írjuk. Ez után a „Hozzáad” gombra kattintva folytathatjuk tovább az értékek megadásával. (3. ábra felső része)

3. ábra: Mátrix megadása felületen keresztül

A sor és az oszlop mezőbe szintén természetes számok adhatóak csak meg, melyek 1 és n között vannak. Az érték mezőbe bármilyen valós számot megadhatunk (csak a 0-tól különböző értékűeket menti a program), az egész és a tizedes számokat ponttal elválasztva. A „Hozzáad” gombra nyomva menthetjük az adott sorhoz és oszlophoz tartozó értéket. (3. ábra alsó része)

Ha már megadtuk az összes nem 0 elemet, a „Kész” gombra nyomva tárolhatjuk el a mátrixot. Ha nem adunk meg egy értéket sem, akkor olyan mátrix kerül tárolásra, melynek minden értéke 0.

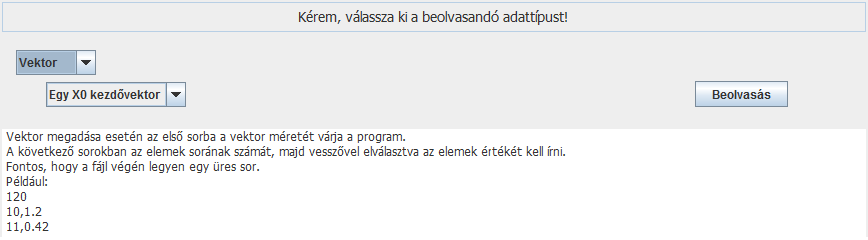
Vektor megadása

Egy vektort szeretnénk megadni, a méret mezőbe az értéket írjuk. Ez után a „Hozzáad” gombra kattintva folytathatjuk tovább az értékek megadásával. (4. ábra felső része)

4. ábra: Vektor megadása felületen keresztül

A sor mezőbe szintén természetes számok adhatóak csak meg, melyek 1 és n között vannak. Az érték mezőbe bármilyen valós számot megadhatunk (csak a 0-tól különböző értékűeket menti a program), az egész és a tizedes számokat ponttal elválasztva. A „Hozzáad” gombra nyomva menthetjük az adott sorhoz tartozó értéket. (4. ábra alsó része)

Ha már megadtuk az összes nem 0 elemet, a „Kész” gombra nyomva tárolhatjuk el a vektort. Ha nem adunk meg egy értéket sem, akkor olyan vektor kerül tárolásra, melynek minden értéke 0.

**Fájlból beolvasással

5. ábra: Adatbekérés fájlból 1

A mátrixot és a vektorokat megadhatjuk úgy, hogy az értékeit fájlból olvassuk be. Először a legördülő menüből kiválasztjuk, hogy milyen adatot kívánunk megadni. Megadhatjuk a mátrixot vagy a vektorok közül a b jobboldali vektort vagy az x0 kezdővektorok egyikét. A „Beolvasás” gombra kattintva megjelenik a fájlkereső ablak, mellyel kiválaszthatjuk az állományaink között a beolvasandó fájlt.

A program által értelmezhető fájl szöveges formátumú (.txt kiterjesztésű), és a felépítése a következő:

* Vektor megadása esetén az első sorba a vektor méretét várja a program. A következő sorokban az elemek sorának számát, majd vesszővel elválasztva az elemek értékét kell írni. Fontos, hogy a fájl végén legyen egy üres sor. Például:

120

10,1.2

11,0.42

…stb

* Mátrix megadása esetén az első sorba a mátrix méretét várja a program. A következő sorokban az elemek sorának, oszlopának számát, majd az elemek értékét kell írni vesszővel elválasztva. Fontos, hogy a fájl végén legyen egy üres sor. Például:

120

10,10,1.2

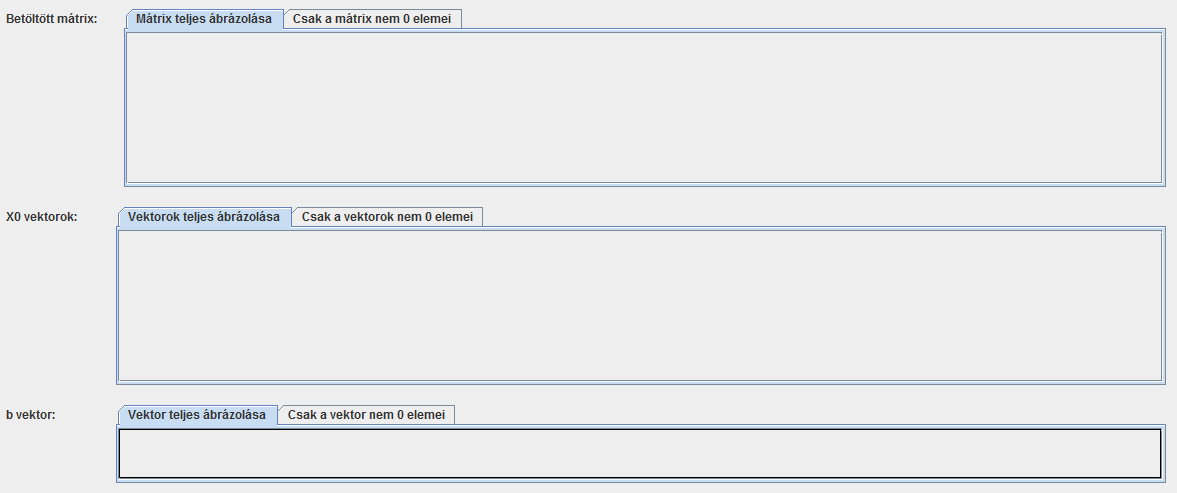
11,11,0.42

…stb

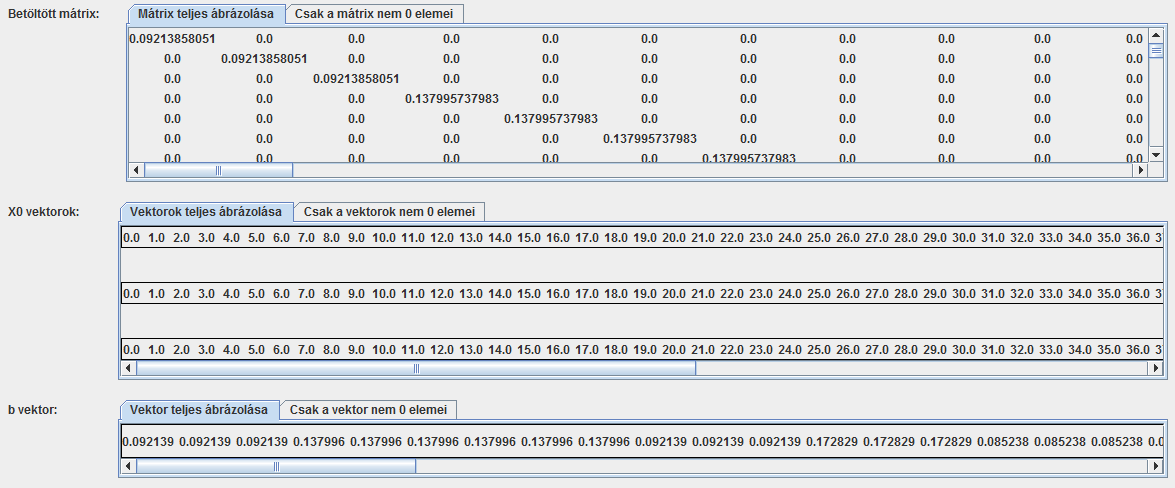
Tesztadat betöltése

Ha nem konkrét problémát szeretnénk megoldani, csak esetleg a program működését megfigyelni, akkor az alkalmazásba betölthető egy előkészített adat együttes. Ekkor nem jelenik meg új képernyő, az program csak egy felugró ablakkal jelzi, hogy a folyamat lezajlott.

#### Aktuális adatok megjelenítése

A betöltött adatokat megtekinthetjük egy kezdetleges megjelenítő panelen. Ha nincs betöltve adat, a panel így néz ki:

6. ábra: adatok nélküli megjelenítőpanel

Ha viszont elérhetőek a vektorok és a mátrix, akkor hasonlóan:

7. ábra: megjelenítőpanel adatokkal

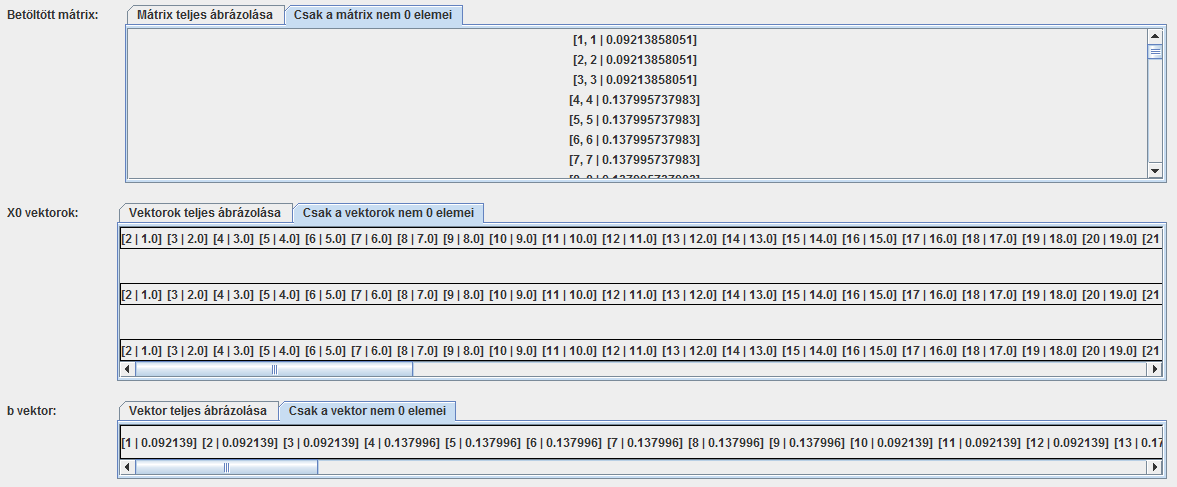
Az felső részen a mátrix, a középsőn a kezdővektorok (soronként, sötét vonallal elválasztva), az alsón a jobboldali vektor értékei jelennek meg egy-egy skrollozható panelen. Minden panel felső részén választhatunk, hogy az összes értéket szeretnénk látni, vagy csak a nem nulla elemeket. A mátrix nem nulla elemei [sorindex, oszlopindex| érték] alakban jelennek meg egymás alatt, pl:

„[1,1|1.2]

[1,5|0.5]

[5,1|0.5] …”

Ezek az értékek sorfolytonosan íródnak ki.

A vektoroknál viszont [index|érték] alakban jelennek meg a nem nulla elemek egymás mellett.

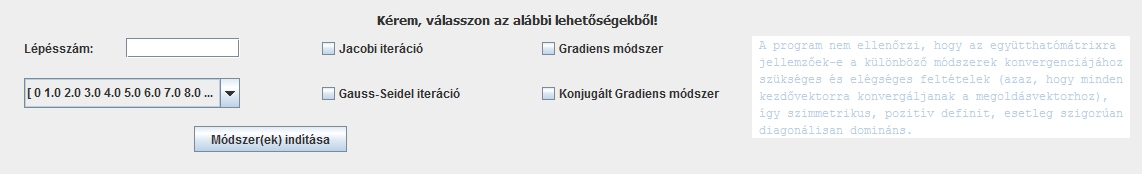
8. ábra: megjelenítőpanel csak a nem nulla elemekkel

A kirajzolás memóriaigénye miatt a program nem engedi a kirajzolást 100x100-as mátrixméret felett.

#### Mentés fájlba

A program képes a memóriában tárolt adatok mentésére szöveges fájlba. A fájl felépítése ugyanolyan lesz, mint amilyet a program bemenetként vár. A megjelenő képernyőn kiválaszthatjuk, hogy mely adatot szeretnénk menteni. Ha az egyik kezdővektort, akkor elérhetővé a kezdővektorokat tartalmazó menü. A „Mentés” gombra kattintva megjelenik a fájlmentő képernyő, amelyben kiválaszthatjuk, hova, milyen néven szeretnénk menteni a fájlt.

### Módszerek

A “**Módszerek kipróbálása**” menüpont rejti a program lényegi részét.

9. ábra: Szűrőpanel

Ha betöltöttünk mátrixot, a kezdő- és jobboldali vektort, és ezek nagysága megegyezik, akkor lehet elkezdeni a folyamatot. A megjelenő képernyőn (szűrőpanel) láthatóak a módszerekre vonatkozó információkat bekérő mezők.

Az első kérdés, hogy hány lépést futtasson a program az adott módszer segítségével. Az alatta lévő lenyíló menüben lehet kiválasztani a betöltött kezdővektorok közül a nekünk kellőt. A képernyő jobb oldalán a módszereket úgy választhatjuk ki, ha nevükre, vagy az előttük lévő jelölőnégyzetre kattintunk. Az összes a megjelölt módszert fogja futtatni a program, és az azokra vonatkozó eredményeket jeleníti meg.

A “Módszer(ek) indítása” gombra kattintva lép működésbe a program, rövid időn belül megjelennek az előző szűrőpanel alatt a módszerek eredményei (eredménypanelek) egy olyan felületen, ami több oldalból áll. Minden megjelölt metódus egy külön oldalon jelenik meg. Az oldalt azonosító „fülön” a módszer neve szerepel, arra kattintva megjelenik az iteráció eredménye.

Az eredménypaneleken megjelenő adatok

Az első grafikonon látható a tapasztalati kontrakciós együttható, míg a másodikon a rezidumvektor normájának változása. Az ezeken ábrázolt eredmények nem láthatóak mindig pontosan. Ha egy oszlop fölé visszük az egeret, a program megjeleníti a konkrét értéket.

A diagramon ábrázolt eredményeket kétféle módon is exportálhatjuk. A kirajzolt képet lementhetjük JPEG fájlba (“Kép mentése” gomb), mely 1200X1200 pixel méretű, de az értékek így sem látszanak pontosan. A másik lehetőség, hogy a megjelenített grafikon pontos értékeit egy txt fájlba írjuk (“Adatok mentése” gomb). Az exportált adat formátuma:

“DÁTUM”-n exportált “MÓDSZER NEVE” számolt “EGYÜTTHATÓ/REZIDUMVEKTOR” változása:

Iterációs lépés száma: “MELYBEN AZ ÉRTÉKET SZÁMOLTUK”; “EGYÜTTHATÓ/REZIDUMVEKTOR” értéke : “ÉRTÉK”

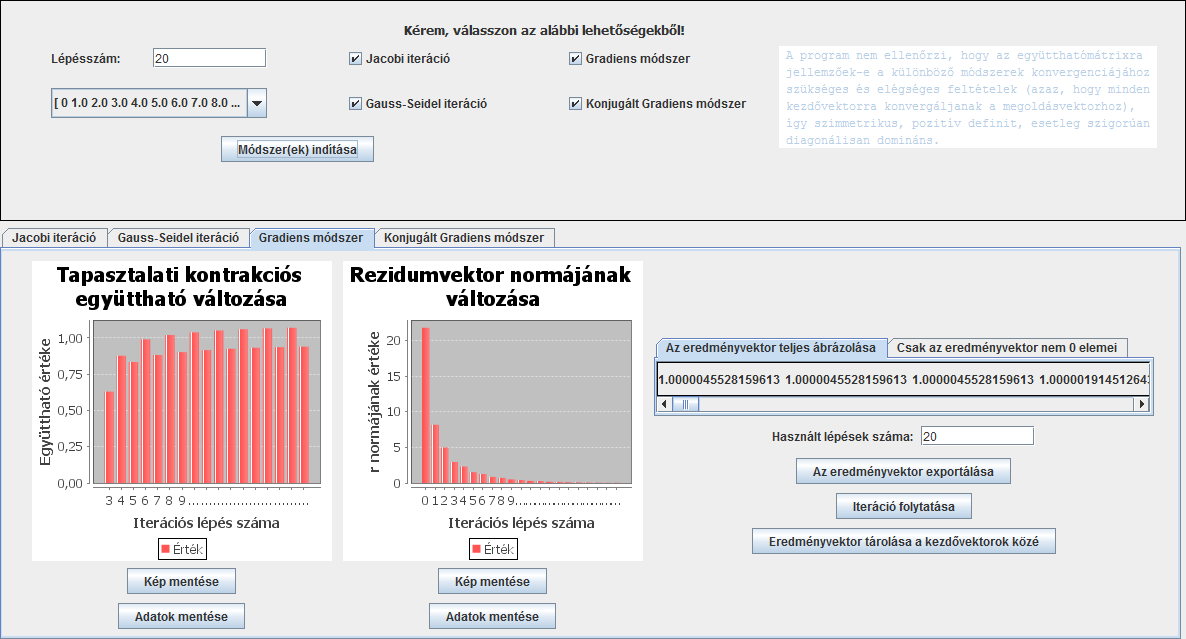
Iterációs lépés száma: “MELYBEN AZ ÉRTÉKET SZÁMOLTUK”; “EGYÜTTHATÓ/REZIDUMVEKTOR” értéke : “ÉRTÉK”

Például:

“2014-05-15-n exportált Jacobi iterációval számolt Tapasztalati kontrakciós együttható változása:

Iterációs lépés száma: 3; Együttható értéke: 0.2581495930773764

Iterációs lépés száma: 4; Együttható értéke: 0.26954823316059784 … “

A diagramok mellet található a már ismert vektorpanel, melyen az iteráció eredményét látjuk. Ezt exportálhatjuk egy fájlba az “Az eredményvektor exportálása” gambbal, mely a “Fájl”-> “Mentés fájlba” menüpontban megismert formátumhoz hasonlóan fog kiíródni.

10. ábra: szűrő- és eredménypanel

Az ereményvektor alatt látható, hogy a módszer hány lépést használt az előre megadott maximumból. Ez akkor lehet kevesebb, ha a program előbb leállítja az iterációt, mert vagy megtalálta, vagy biztosan konvergál a megoldáshoz a módszer (q>5).

A panelen lehetőség van még az iteráció folytatására, azaz a szűrőpanelen megadott lépésszámban újra futtatni az adott iterációt az  eredményvektorral, mint kezdővektorral. Ekkor a grafikonokon az új adatok láthatóak, az új xn is látszik, a használt lépések pedig hozzáadódnak az eddigiekhez.

Lehetőség van még az eredményvektor eltárolására az aktuális kezdővektorok között. Ez akkor lehet hasznos, ha egy módszer eredményvektorából szeretnénk indítani egy másik iterációt. Ahogy a program is figyelmeztet rá, ha használni szeretnénk ezt a vektort, újra kell töltenünk a panelt (“Módszerek” menü -> “Módszerek kipróbálása” gombbal).

### Súgó

A „Súgó” menüben egy általános leírást találunk a programról és a benne lévő funkciókról.

Fejlesztői dokumentáció

A probléma részletes leírása

**A megvalósított adattípus leírása**

**A használt módszerek leírása**

Általános leírás a klasszikus iterációkról

A program a használt módszerekből adódóan felteszi, hogy az együtthatómátrix négyzetes, a determinánsa nullától különböző. Ekkor az egyértelmű megoldást jelölje **x**\*. Az iterációs módszerek általában olyan konvergens sorozatokat konstruálnak, melyek határértéke az egyenlet megoldása. Az

iterációt az A \* **x** = **b** egyenletrendszerrel konzisztensnek hívjuk, ha

(Az **x**\* vektor az egyenletrendszer megoldása.)

Tegyük fel, hogy az A együtthatómátrixot előállítottuk A = S – T alakban, ahol S reguláris mátrix. Ekkor a

egyenlőséget S inverzével balról szorozva, majd **x**-et kifejezve

Ezen egyenlőség miatt az

iteráció konzisztens az egyenletrendszerrel.

Az S mátrixot prekondicionálási mátrixnak hívjuk. Ez a mátrix határozza meg, hogy mennyire nehéz vagy könnyű az iteráció végrehajtása. Megválasztását két, egymással ellentétes követelmény határozza meg. Egyrészt könnyen invertálhatónak kell lennie, hiszen az iterációhoz szükség van a mátrix inverzére, másrészt a

egyenlőség miatt jó lenne, ha S "közel lenne A-hoz", hiszen akkor a B mátrix spektrálsugara jóval kisebb lehetne, mint 1, ami gyors konvergenciát eredményezne.

Az implementált módszerek közül a két klasszikus iteráció (Jacobi és Gauss-Seidel) alapjai az előzőekben taglaltak. Mindkét esetben a mátrixot A = D - L - U is írjuk fel, ahol D a diagonális elemek, L a diagonális alatti elemek -1-szereseinek, míg U a diagonális feletti elemek -1-szereseinek mátrixa. Feltesszük, hogy D főátlójának (azaz A főátlójának) egyik eleme sem nulla. Ez mindig elérhető az egyenletek megfelelő átrendezésével.

*Jacobi-iteráció*

Az S = D és T = U + L választással konstruált

iterációt ( tetszőleges kezdővektor) Jacobi-iterációnak nevezzük. Az iterációt vektorkomponensenként kiírva az

iterációt nyerjük.

*Gauss-Seidel-iteráció*

Az S = D + L és T = U választású

iterációt (**x**0 tetszőleges kezd®vektor) Gauss-Seidel-iterációnak nevezzük. Hogy lássuk az iteráció Jacobi-iterációval való kapcsolatát, alakítsuk át az iterációs képletet. Szorozzunk először balról a (D-L) mátrixszal, majd adjunk hozzá mindkét oldalhoz -et, és végül szorozzunk D inverzével. A fenti ekvivalens átalakítások után az

iterációt nyerjük. Látható, hogy a Jacobi-iterációhoz kép est csak annyi a különbség, hogy a tag helyett a Gauss-Seidel-iterációban szerepel. Látszólag ez az iteráció nem explicit, hiszen a jobb oldalon is szerep el az vektor, de mivel szigorú (a főátlóban nullák szerepelnek) alsó háromszögmátrix, így mégis egyszerűen meghatározható. Az vektor első elemének meghatározásához nincs szükség az vektorra. A második elem meghatározásához pedig csak a korábban meghatározott első elemre van szükség, stb. Még jobban látszik a kapcsolat a két iteráció között, ha a Gauss-Seidel-iteráció utóbbi alakját komponensenként is kiírjuk:

(i = 1,..., n). Ha összevetjük ezt a Jacobi-iteráció képletével, akkor látható, hogy a Jacobi-iteráció az vektor komponenseinek meghatározásához csak az vektort használja, míg a Gauss-Seidel-módszer az vektor i-edik elemének meghatározásához felhasználja a vektor korábban kiszámolt j = 1,…,i-1 indexű elemeit ( megfelelő elemei helyett)

Általános leírás a variációs módszerekről

A lineáris egyenletrendszerek megoldásának egy másik iterációs megoldási lehetősége szimmetrikus, pozitív definit mátrixokra. Az alap ötlet az, hogy megadunk egy többváltozós függvényt, melynek abszolút minimumhelye az egyenletrendszer megoldása. Ezt a minimumhelyet keressük meg egy megfelelő iterációs eljárással.

Legyen tehát szimmetrikus, pozitív definit mátrix, és tekintsük a

n-változós függvényt. Az összeszorzott mátrixok mérete szerint a jobb oldali kifejezés valóban egy valós számot (egy (1x1) -es mátrixot) rendel minden vektorhoz. Ennek a függvénynek egyetlen abszolút minimumhelye van az halmazon és ez pontosan az A \* **x** = **b** egyenletrendszer megoldása.

*Gradient-módszer*

A Gradiens-módszernél egy x pontból, ahol a maradékvektorm az er vektor irányába az pontba lépünk tovább. Innét pedig az itteni

maradékvektor irányában keressük a következő iránymenti minimumot. Az

egyenlőség miatt látható, hogy az iteráció során az egymás utáni keresési irányok merőlegesek egymásra.

A módszer algoritmusa:

*Konjugált Gradiens-módszer*

Természetesen vetődik fel az a kérdés, hogy a keresési irányok másfajta megválasztásával nem lehetne-e gyorsítani valahogy a konvergenciát. Induljunk ki abból az állításból, hogy az pontbeli keresési irány (a Gradiensvektor (-1)-szerese) merőleges az pontbeli maradékvektorra.

Ebből a képletből látható, hogy az pontból az megoldásba vezető vektor teljesíti a feltételt. Az egyszerűbb megfogalmazás kedvéért vezessük be az alábbi fogalmat.

Definíció:

Legyen adva egy szimmetrikus pozitív definit mátrix. Azt mondjuk, hogy az **x** és **y** vektorok *A-konjugáltak (vagy A ortogonálisak),* ha *.*

Vegyük észre, hogy ha A az egységmátrix, akkor az A-ortogonalitás pontosan a hagyományos skaláris szorzatbeli ortogonalitást jelenti.

A defníció segítségével tehát mondhatjuk, hogy a keresési irányt úgy kell egválasztanunk, hogy az legyen A-ortogonális a keresési irányra. Rögtön látjuk, hogy ez lehetséges az megoldás ismerete nélkül is. Keressük

alakban a második keresési irányt. Az A-ortogonalitási feltételt felhasználva ekkor

Továbbá,

Így

Tehát a módszer algoritmusa a következő: