Krzysztof Wesołowski
Automatyka i Robotyka
Grupa: środa, 14.00 - LD_3
Data: 15 XII 2008

Rozwiązywanie układów równań liniowych

Metody dokładne:

Dane do obliczeń

$$mWsp := \begin{pmatrix} 1 & 8 & 2 \\ -20 & 12 & 8 \\ -3 & 5 & 17 \end{pmatrix} \qquad WW := \begin{pmatrix} 46 \\ 8 \\ 93 \end{pmatrix}$$

Sprawdzam właściwy wynik w celu późniejszej konfrontacji wyników moich obliczeń z rzeczywistością:

$$\text{lsolve}(mWsp, WW) = \begin{pmatrix} 4 \\ 4 \\ 5 \end{pmatrix}$$

Metoda Gaussa

Pierwszym krokiem który ulatwi dalsza implementację jest utworzenie macierzy uzupełnionej, dzieki czemu operacje na wierszach będą przeksztalcać zarowno współczynniki jak i wyrazy wolne.

$$mUz := \text{augment}(mWsp, WW)$$

$$mUz = \begin{pmatrix} 1 & 8 & 2 & 46 \\ -20 & 12 & 8 & 8 \\ -3 & 5 & 17 & 93 \end{pmatrix}$$

Teraz mozemy utworzyc funkcje ktora przeksztalci wyzej wymieniona macierz tak aby macierz wspolczynnikow byla macierza trojkatna. zakladam tutaj ze nie wystapia przy tym problemy z dzieleniem przez zero etc. najpierw więc należy sprawdzić czy na przekatnej nie ma wyrazow rownych 0, oraz czy wyznacznik jest rozny od zera.

$$|mWsp| = 2.564 \times 10^3$$

Mozemy wiec droga przeksztalcen uzyskac trojkatna macierz wspolczynnikow, oraz odpowiadajacy jej wektor wyrazow wolnych, co rezlizuje poniższa funkcja:

$$\text{tGauss}(M) := \left| \begin{array}{l} \text{wynik} \leftarrow M \\ \text{for } \text{kol} \in 0 .. \text{rows}(M) - 2 \\ \quad \text{for } \text{wiersz} \in \text{kol} + 1 .. \text{rows}(M) - 1 \\ \quad\quad \left| \begin{array}{l} \text{wsp} \leftarrow \dfrac{\text{wynik}_{\text{wiersz},\text{kol}}}{\text{wynik}_{\text{kol},\text{kol}}} \\ \text{for } \text{tk} \in 0 .. \text{cols}(M) - 1 \\ \quad \text{wynik}_{\text{wiersz},\text{tk}} \leftarrow \text{wynik}_{\text{wiersz},\text{tk}} - \text{wynik}_{\text{kol},\text{tk}} \cdot \text{wsp} \end{array} \right. \\ \text{wynik} \end{array} \right.$$

Poniżej wyświetlam wynik jej dzialania: macierz trojkatna wraz z dolaczonym wektorem wyrazow wolnych. Taka macierz latwo przeksztalcic do macierzy diagonalnej z dolaczonym wektorem wyrazow wolnych, co w praktyce da nam juz rozwiazanie.

$$\text{tGauss}(\text{mUz}) = \begin{pmatrix} 1 & 8 & 2 & 46 \\ 0 & 172 & 48 & 928 \\ 0 & 0 & 14.907 & 74.535 \end{pmatrix}$$

Na potrzeby metody Gaussa jak i metody rozkladu LU zaimplementuje funkcje sprowadzajaca macierz trojkatna uzupelniona o wektor wyrazów wolnych do odpowiednio uzypelnionej macierzy diagonalnej (Triangle to Diagonal), sama funkcja rozpoznaja z jaka macierza trojkatna ma doczynienia.

$$\text{TtoD}(M) := \left| \begin{array}{l} \text{for } \text{kol} \in \text{rows}(M) - 1 .. 1 \qquad \text{if } M_{\text{rows}(M)-1,0} = 0 \\ \quad \text{for } \text{wie} \in 0 .. \text{kol} - 1 \\ \quad\quad \left| \begin{array}{l} \text{wsp} \leftarrow \dfrac{M_{\text{wie},\text{kol}}}{M_{\text{kol},\text{kol}}} \\ \text{for } i \in 0 .. \text{cols}(M) - 1 \\ \quad M_{\text{wie},i} \leftarrow M_{\text{wie},i} - M_{\text{kol},i} \cdot \text{wsp} \end{array} \right. \\ M \leftarrow \text{tGauss}(M) \quad \text{otherwise} \\ M \end{array} \right.$$

A ponizej prezentuje wynik jej działania:

$$\text{dGauss} := \text{TtoD}(\text{tGauss}(\text{mUz}))$$

$$\text{dGauss} = \begin{pmatrix} 1 & 0 & 0 & 4 \\ 0 & 172 & 0 & 688 \\ 0 & 0 & 14.907 & 74.535 \end{pmatrix}$$

Pozostała jeszcze funkcja podajaca szukany wektor X korzystajaca z macierzy diagonalnej z dolaczonym wektorem wyrazow wolnych (Diagonal to X)

$$\text{DtoX}(M) := \left| \begin{array}{l} \text{for } i \in 0..\text{cols}(M) - 2 \\ \quad \text{wynik}_i \leftarrow \dfrac{M_{i,\,\text{cols}(M)-1}}{M_{i,\,i}} \\ \text{wynik} \end{array} \right.$$

$$\text{DtoX}(\text{dGauss}) = \begin{pmatrix} 4 \\ 4 \\ 5 \end{pmatrix}$$

Oraz przydatna na przyszlosc funkcja podajaca wynik prosto z macierzy trojkątnej uzupelnionej,(Triangle to X) bedaca zlozeniem powyzszczych funkcji:

$$\text{TtoX}(M) := \text{DtoX}(\text{TtoD}(M))$$

Oraz jej działanie:

$$\text{TtoX}(\text{tGauss}(\text{mUz})) = \begin{pmatrix} 4 \\ 4 \\ 5 \end{pmatrix}$$

Ewentualnie mozna jeszcze zlozyc ww. funkcje w jedna rozwiazaujaca uklad o zadanej macierzy wsp i wektorze wyrazow wolnych, podobnej do lsolve:

$$\text{gsolve}(M, w) := \text{TtoX}(\text{tGauss}(\text{augment}(M, w)))$$

$$\text{gsolve}(\text{mWsp}, \text{WW}) = \begin{pmatrix} 4 \\ 4 \\ 5 \end{pmatrix}$$

Metoda rozkładu LU

Macierz U, czyli macierz górna w rozkładzie LU jest macierza ptrzymywana przez zwykly algortytm Gaussa, i jest ona maceirza trojkatna gorna.

$$\text{mUpper}(M) := \left| \begin{array}{l} \text{wynik} \leftarrow M \\ \text{for } \text{kol} \in 0..\text{rows}(M) - 2 \\ \quad \text{for } \text{wiersz} \in \text{kol} + 1..\text{rows}(M) - 1 \\ \quad\quad \left| \begin{array}{l} \text{wsp} \leftarrow \dfrac{\text{wynik}_{\text{wiersz},\,\text{kol}}}{\text{wynik}_{\text{kol},\,\text{kol}}} \\ \text{for } \text{tk} \in 0..\text{cols}(M) - 1 \\ \quad \text{wynik}_{\text{wiersz},\,\text{tk}} \leftarrow \text{wynik}_{\text{wiersz},\,\text{tk}} - \text{wynik}_{\text{kol},\,\text{tk}} \cdot \text{wsp} \end{array} \right. \\ \text{wynik} \end{array} \right.$$

Poniżej wynik czyli macierz gorna z naszej macierzy wspolczynnikow:

$$\text{mUpper}(\text{mWsp}) = \begin{pmatrix} 1 & 8 & 2 \\ 0 & 172 & 48 \\ 0 & 0 & 14.907 \end{pmatrix}$$

Z kolei macierz L, zwana macierza dolna jest macierza o 1 na przekatnej, oraz elementach pod przekatna równym współczynnikom przez ktore nalezalo pomnozyc wiersz o numrze rownym biezacej kolumnie w celu elimanacji danego wsplczynnika.

$$
\begin{aligned}
&\text{mLower}(M) := \\
&\quad \text{temp} \leftarrow M \\
&\quad \text{for } i \in 0..\text{rows}(M) - 1 \\
&\qquad \text{wynik}_{i,i} \leftarrow 1 \\
&\quad \text{for } kol \in 0..\text{rows}(M) - 2 \\
&\qquad \text{for } wiersz \in kol + 1..\text{rows}(M) - 1 \\
&\qquad\quad wsp \leftarrow \frac{\text{temp}_{wiersz,kol}}{\text{temp}_{kol,kol}} \\
&\qquad\quad \text{wynik}_{wiersz,kol} \leftarrow wsp \\
&\qquad\quad \text{for } tk \in 0..\text{cols}(M) - 1 \\
&\qquad\qquad \text{temp}_{wiersz,tk} \leftarrow \text{temp}_{wiersz,tk} - \text{temp}_{kol,tk} \cdot wsp \\
&\quad \text{wynik}
\end{aligned}
$$

Poniżej wynik czyli macierz dolna z naszej macierzy wspolczynnikow:

$$\text{mLower}(\text{mWsp}) = \begin{pmatrix} 1 & 0 & 0 \\ -20 & 1 & 0 \\ -3 & 0.169 & 1 \end{pmatrix}$$

Oraz sprawdzenie czy dokonany rozkład LU jest poprawny:

$$\text{mWsp} - \text{mLower}(\text{mWsp}) \cdot \text{mUpper}(\text{mWsp}) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Pozostaje wiec wyznacznie z rownan
mWsp x = WW*
*L * U * x = WW*
*L * y = WW*
*U * x = y*
Poszukiwanego wektora x

Najpierw obliczamy tymczasowy wektor y, rozwiazujac prosty uklad rownan z macierza trojkatna dolna, korzystamy z wczesniej przygotowanych funkcji:

$$\text{tmp} := \text{TtoX}(\text{augment}(\text{mLower}(\text{mWsp}), \text{WW}))$$

$$\text{tmp} = \begin{pmatrix} 46 \\ 928 \\ 74.535 \end{pmatrix}$$

Nastepnie korzystajac z wyliczonego wektora y i macierzy trojkatnej gornej obliczamy koncowe rozwiazanie:

$$\mathrm{TtoX}(\mathrm{augment}(\mathrm{mUpper}(\mathrm{mWsp}),\mathrm{tmp})) = \begin{pmatrix} 4 \\ 4 \\ 5 \end{pmatrix}$$

Podsumowujac stworzmy jeszcze funkcje bedaca nasza implementacja metody lsolve, o nazwie lusolve.

$$\mathrm{lusolve}(M,w) := \mathrm{TtoX}(\mathrm{augment}(\mathrm{mUpper}(M),\mathrm{TtoX}(\mathrm{augment}(\mathrm{mLower}(M),w))))$$

I przetestujmy jej działanie:

$$\mathrm{lusolve}(\mathrm{mWsp},\mathrm{WW}) = \begin{pmatrix} 4 \\ 4 \\ 5 \end{pmatrix}$$

Metody przybliżone - iteracyjne

Dane do obliczeń

$$\mathrm{mWsp} := \begin{pmatrix} 2 & 3 & 0 \\ 0 & 2 & 2 \\ 1 & 0 & 9 \end{pmatrix} \qquad \mathrm{WW} := \begin{pmatrix} 33 \\ 36 \\ 84 \end{pmatrix}$$

Sprawdzam ich rozwiazanie uzywajac wbudowanych metod rozwiazania, w celu pozniejszego porownywania wynikow::

$$\mathrm{lsolve}(\mathrm{mWsp},\mathrm{WW}) = \begin{pmatrix} 3 \\ 9 \\ 9 \end{pmatrix}$$

Metoda Jacobiego

Funcje rozbijajace macierz współczynników na macierze L,D,U

$$\begin{aligned} &\mathrm{iL}(M) := \\ &\quad \text{for } w \in 0..\mathrm{rows}(M)-1 \\ &\quad\quad \text{for } k \in 0..\mathrm{cols}(M)-1 \\ &\quad\quad\quad \mathrm{wynik}_{w,k} \leftarrow M_{w,k} \text{ if } w > k \\ &\quad\quad\quad \mathrm{wynik}_{w,k} \leftarrow 0 \text{ otherwise} \\ &\quad \mathrm{wynik} \end{aligned}$$

$$\begin{aligned} &\mathrm{iD}(M) := \\ &\quad \text{for } w \in 0..\mathrm{rows}(M)-1 \\ &\quad\quad \text{for } k \in 0..\mathrm{cols}(M)-1 \\ &\quad\quad\quad \mathrm{wynik}_{w,k} \leftarrow M_{w,k} \text{ if } w = k \\ &\quad\quad\quad \mathrm{wynik}_{w,k} \leftarrow 0 \text{ otherwise} \\ &\quad \mathrm{wynik} \end{aligned}$$

```
iU(M) := | for w ∈ 0..rows(M) − 1
         |   for k ∈ 0..cols(M) − 1
         |     wynik_{w,k} ← M_{w,k}  if w < k
         |     wynik_{w,k} ← 0  otherwise
         | wynik
```

I rozbite macierze

$$iL(mWsp) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \qquad iD(mWsp) = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 9 \end{pmatrix} \qquad iU(mWsp) = \begin{pmatrix} 0 & 3 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix}$$

Przy stosowaniu metod iteracyjnych otrzymujemy wzór iteracyjny postaci $x^{i+1} = M\,x^{i} + w$, pozostaje wiec obliczenie macierzy M i wektora w, dla metody Jacobiego:

$$JacM(M) := -iD(M)^{-1} \cdot (iL(M) + iU(M))$$

$$JacW(M, w) := iD(M)^{-1} \cdot w$$

Potrzebny bedzie jeszcze promien spektralny macierzy

```
pspe(M) := | wektor ← eigenvals(M)
           | wektor ← |wektor|  (vectorize)
           | max(wektor)
```

I odpowiednia norma maksimum wektora

```
wmax(w) := | for i ∈ 0..last(w)
           |   w_i ← |w_i|
           | max(w)
```

Na potrzeby tego sprawozdania sprawdzam promień spektralny macierzy. Nie właczam tego do implementacji gdyż jest to skrajnie nieefektywne(wiecej w wstepie teoretycznym)

$$pspe(JacM(mWsp)) = 0.55$$

$TOL := 10^{-4}$ *Ustawiam tolerancje wg zaleceń*

Oraz sama metoda obliczajaca kolejne przyblizenia w oparciu o macierze M i wektor W, zwane mnoz i dodaj wewnatrz funkcji, w celu prezentacji wynikow dodatkowo mozna ograniczyc ilosc iteracji, tak aby zobaczyc wynik po ich zadanej liczbie.

```
jacsolve(M, w, limit) := | mnoz ← JacM(M)
                         | dodaj ← JacW(M, w)
                         | for i ∈ 0.. last(w)
                         |     current_i ← 0
                         |     previous_i ← 0
                         | for i ∈ 0.. limit
                         |     current ← mnoz·previous + dodaj
                         |     (return stack(current, i)) if wmax(current − previous) < TOL
                         |     previous ← current
                         | stack(current, i)
```

I rozwiązanie, z dolaczona iloscia iteracji

$$\text{jacsolve}(\text{mWsp}, \text{WW}, 50) = \begin{pmatrix} 3 \\ 9 \\ 9 \\ 21 \end{pmatrix}$$

Metoda Gaussa-Seidla

funkcje obliczajace macierz M i wektor b, uzywane w algorytmie iteracyjnym

$$\text{gsM}(M) := -(\text{iD}(M) + \text{iL}(M))^{-1} \cdot \text{iU}(M)$$

$$\text{gsW}(M, w) := (\text{iD}(M) + \text{iL}(M))^{-1} \cdot w$$

Na potrzeby tego sprawozdania sprawdzam promień spektralny macierzy. Nie właczam tego do implementacji gdyż jest to skrajnie nieefektywne(wiecej w wstepie teoretycznym)

$$\text{pspe}(\text{gsM}(\text{mWsp})) = 0.408$$

$$
\begin{aligned}
&\text{gssolve}(M, w, \text{limit}) := \\
&\quad \text{mnoz} \leftarrow \text{gsM}(M) \\
&\quad \text{dodaj} \leftarrow \text{gsW}(M, w) \\
&\quad \text{for } i \in 0..\text{last}(w) \\
&\quad\quad \text{current}_i \leftarrow 0 \\
&\quad\quad \text{previous}_i \leftarrow 0 \\
&\quad \text{for } i \in 0..\text{limit} \\
&\quad\quad \text{for } k \in 0..\text{last}(w) \\
&\quad\quad\quad \text{current}_k \leftarrow \left\lceil \sum_{j=0}^{\text{last}(w)} \left(\text{mnoz}_{k,j} \cdot \text{current}_j\right) + \text{dodaj}_k \right\rceil \\
&\quad\quad (\text{return } \text{stack}(\text{current}, i)) \text{ if } \text{wmax}(\text{current} - \text{previous}) < \text{TOL} \\
&\quad\quad \text{previous} \leftarrow \text{current} \\
&\quad \text{stack}(\text{current}, i)
\end{aligned}
$$

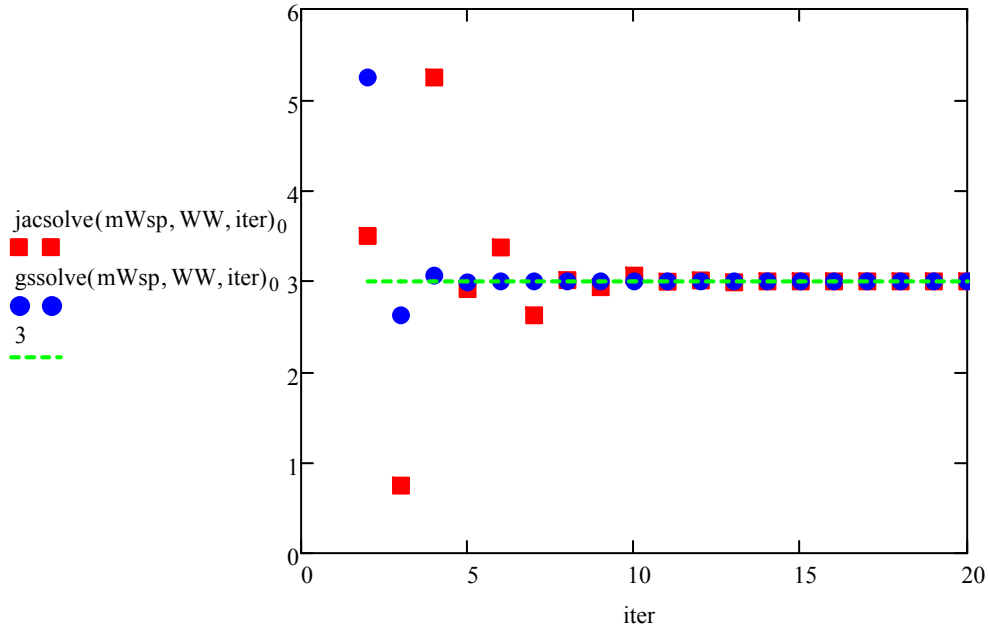
Oraz przyklad działania funkcji:

$$
\text{gssolve}(\text{mWsp}, \text{WW}, 500) = \begin{pmatrix} 3 \\ 9 \\ 9 \\ 9 \end{pmatrix}
$$

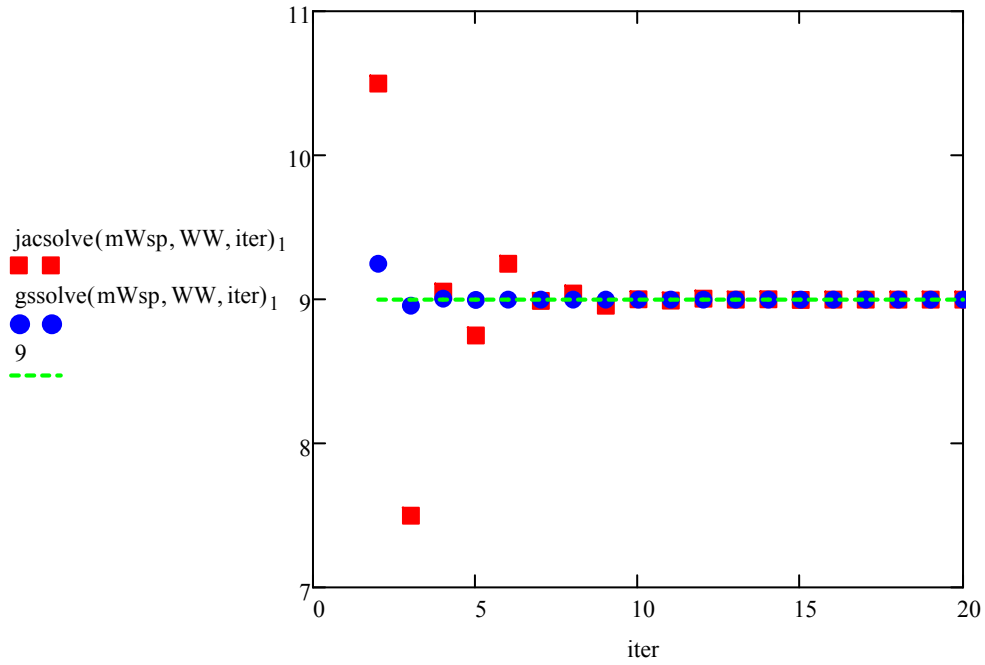
Ostatecznie mozna porownac powyzsze metody, porownojac wyniki w kolejnych iteracjach:

$$
\text{iter} := 2..20
$$

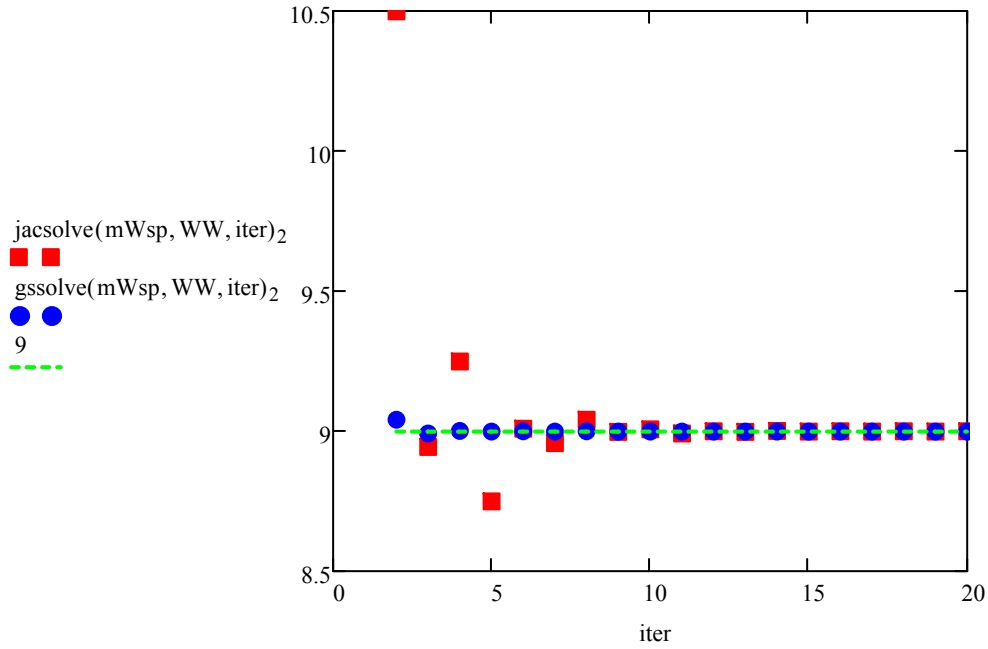
Przebieg przyblizania pierwszej skladowej wyniku:



Przebieg przyblizania drugiej skladowej wyniku:



Przebieg przyblizania trzeciej skladowej wyniku:



Wnioski zalaczam w pliku PDF