# Metody numeryczne w elektrotechnice

## Ćwiczenie 3 — Układy równań liniowych i dekompozycje macierzy

Dariusz Borkowski

25 marca 2014

## Wstęp

Na zajęciach rozwiążecie zadania, które pokażą wam różnice, wady i zalety wybranych metod rozwiązywania układów równań liniowych. Dowiesz się na jakie problemy można natrafić podczas stosowania tych metod i jak sobie z nimi poradzić. Cztery zadania są do rozwiązania w Matlabie, jedno w C.

## 1 Porównanie operatorów $\backslash$ i `inv` Matlaba — 1 pkt

Układ równań liniowych

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

w Matlabie można rozwiązywać poprzez odwracanie macierzy funkcją `inv` lub stosując operator dzielenia lewostronnego[1] $\backslash$ w sposób jak poniżej:

```
x1 = inv(A) * b       % odwracanie macierzy A , to samo co:   x = A ^ -1 * b
x2 = A \ b            % dzielenie lewostronne
```

Czas i wynik obu realizacji jest może być różny. Zwykłe odwracanie działa wolniej i w szczególnych przypadkach może dać mniej dokładny wynik, co sami zaraz sprawdzicie. Operator $\backslash$ znajduje rozwiązanie $\boldsymbol{x}$ układu (1) metodą bezpośrednią optymalną dla podanej macierzy $\boldsymbol{A}$. Matlab sprawdza postać macierzy $\boldsymbol{A}$ (m.in. czy macierz jest trójkątna, czy jest dodatnio określona symetryczna, itd.) i w zależności od jej postaci stosuje różnego rodzaju dekompozycje, będące w większości wariantami eliminacji Gaussa. W przypadku najogólniejszym stosowana jest dekompozycja LU.

Porównaj w Matlabie wyniki rozwiązania źle uwarunkowanego układu równań (2) dwoma powyższymi metodami.

$$\begin{bmatrix} eps & eps & -1 \\ eps & -eps & 0 \\ -1 & eps & eps \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{2}$$

Sprawdź czy wyniki `x1` i `x2` obu metod są jednakowe czy różne. W zależności od formatu wyświetlania liczb w Matlabie wyniki mogą wydawać się być jednakowe. Wyświetlenie różnicy `x2-x1` daje rzeczywisty obraz. Znajdź taki format wyświetlania liczb, który pozwala stwierdzić, że `x1`, `x2` nie są równe się bez obliczania ich różnicy. W tym celu wykorzystaj polecenie `format` zapoznając się wcześniej z jego dokumentacją i przykładami użycia.

Porównaj czasy rozwiązywania układu równań postaci (1), gdzie $\boldsymbol{A}$ jest macierzą o wymiarach $100 \times 100$ wygenerowaną funkcją `randn`. W tym celu w pętli rozwiązuj układ równań kolejno obydwoma metodami (obie metody jedna po drugiej w każdym obiegu pętli). Po wykonaniu 100 (na wolniejszych komputerach odpowiednio mniej) przebiegów pętli uśrednij czas rozwiązywania układu równań obydwoma metodami i wypisz na ekran stosunek średnich czasów obliczeń. Do wyznaczania czasu potrzebnego na wykonanie fragmentu programu służą funkcje `tic` i `toc`.

## 2 Eliminacja Gaussa i pivoting — 1 pkt

Wyznacz na papierze rozwiązanie (zostanie sprawdzone!) $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ układu równań:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \qquad \text{gdzie} \qquad \boldsymbol{A} = \begin{bmatrix} 2 & 2 & 4 \\ 1 & 1 & 4 \\ 1 & 4 & 1 \end{bmatrix} \quad \boldsymbol{b} = \begin{bmatrix} 4 \\ 4 \\ 1 \end{bmatrix} \tag{3}$$

[1] Istnieje też operator dzielenia prawostronnego

metodą eliminacji Gaussa. Dzieli się ona na dwa etapy. W pierwszym od wszystkich wierszy oprócz pierwszego odejmujemy wiersz pierwszy pomnożony przez taki współczynnik, który spowoduje, że wszystkie (oprócz pierwszego) elementy pierwszej kolumny się wyzerują. Współczynnik ten ma wartość $\frac{a_{i,1}}{a_{1,1}}$ dla pierwszej kolumny, gdzie $i = 2 \dots N$ jest numerem wiersza, a $N$ rozmiarem macierzy $\boldsymbol{A}$. Analogiczną operację przeprowadzamy na kolejnych wierszach, na których jeszcze nie została przeprowadzona przy czym współczynnik dla $k$-tego wiersza ma wartość $\frac{a_{i,k}}{a_{k,k}}$ gdzie $i = k-1 \dots N$. Te same operacje co na wierszach macierzy przeprowadzamy również na elementach wektora $\boldsymbol{b}$, dlatego dla wygody najczęściej dopisujemy wektor $\boldsymbol{b}$ jako dodatkową kolumnę macierzy $\boldsymbol{A}$.

Po przeprowadzeniu eliminacji otrzymujemy układ równań z macierzą trójkątną, który mozna łatwo rozwiazać stosując podstawienie wstecz. Polega ono na wyznaczaniu wartości niewiadomych począwszy od ostatniego wiersza przekształconego układu równań, w którym występuje tylko jedna niewiadoma (u nas $x_3$). Po wyznaczeniu wartości tej niewiadomej podstawiamy ją do równania, w którym pozostawały tylko dwie niewiadome i wyznaczamy drugą z nich i tak dalej.

Podczas eleminacji natrafisz na problem dzielenia przez zero. Omiń problem zamieniając miejscami dwa ostatnie równania czyli wiersze macierzy rozszerzonej $\left[\begin{array}{c|c} \boldsymbol{A} & \boldsymbol{b} \end{array}\right]$. Ta zamiana wierszy nazywana wyborem elementu głównego lub *pivotingiem* jest stosowana w Matlabie podczas rozwiązywania układów równań liniowych i dekompozycji macierzy. Zamiana następuje nie tylko w przypadku natrafienia na zero na przekątnej macierzy. Matlab zamienia kolejność wierszy macierzy za każdym razem tak[2], aby w mianowniku elementu głównego (czyli opisanego wcześniej współczynnika) występowała największa liczba w danej kolumnie. Ma to na celu zmniejszenie błędów numerycznych związanych z dzieleniem przez liczby bliskie zeru.

Po wykonaniu eliminacji wykonaj podstawienie wstecz, a następnie sprawdź wynik z wynikiem uzyskanym w Matlabie.

## 3 Dekompozycja LU — 2 pkt

Schemat działań dekompozycji LU jest taki sam jak w eliminacji Gaussa, z tym, że oprócz tworzenia górnej macierzy trójkątnej $\boldsymbol{U}$ współczynniki użyte podczas eliminacji służą do zbudowania dolnej macierzy trójkątnej $\boldsymbol{L}$. Otrzymane w wyniku dekompozycji macierze łączy związek

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U} \tag{4}$$

wobec czego układ równań postaci (1) można zapisać jako

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{L}\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b} \tag{5}$$

i po podstawieniu $\boldsymbol{y} = \boldsymbol{U}\boldsymbol{x}$ rozwiązać go poprzez rozwiązanie kolejno dwóch układów równań z macierzami trójkątnymi:

$$\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b} \tag{6}$$

$$\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y} \tag{7}$$

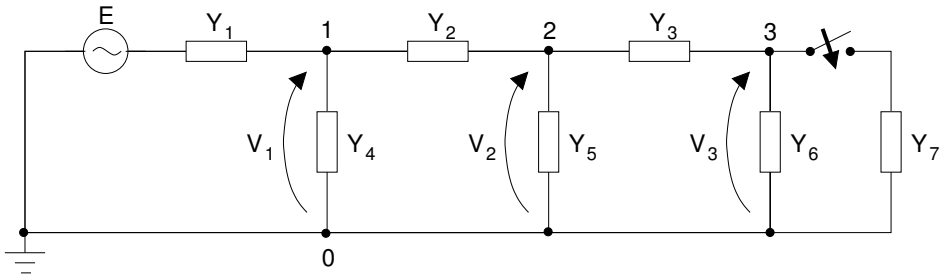
Można to wykonać niewielkim kosztem obliczeniowym przez podstawienie w przód, a następnie wstecz.

Zysk z dekompozycji LU nad rozwiązywaniem układu równań metodą eliminacji Gaussa można zauważyć dopiero gdy potrzbujemy rozwiązać ten układ równań o niezmienionej macierzy $\boldsymbol{A}$ dla wielu różnych wektorów $\boldsymbol{b}$. Zysk wynika z tego, że eliminacja Gaussa operuje również na wektorze $\boldsymbol{b}$ tak więc, przy każdej zmianie wektora $\boldsymbol{b}$ na nowy wykonywana jest cała eliminacja (równoznaczna z dekompozycją LU) i podstawienie wstecz. Jeśli zaś dokonamy dekompozycji LU samej macierzy $\boldsymbol{A}$ to po każdej zmianie wktora $\boldsymbol{b}$ wykonywane są podstawienie w przód i wstecz (nie trzeba powtarzać dekompozycji, która zajmuje więcej czasu niż podstawienia).

W tym zadaniu sprawdzisz, że tak jest w istocie. Jako przykład posłuży nam uproszczony system energetyczny pokazany na rysunku 1. Uproszczenie polega, na wykorzystaniu jedynie modułów admitancji oraz modułu wartości skutecznej napięcia zamiast wartości zespolonych, choć nic nie stoi na przeszkodzie,

[2] Można to sprawdzić wykonując ręcznie dekompozycję macierzy $\boldsymbol{A} = \begin{bmatrix} 2 & 2 & 4 \\ 1 & 2 & 2 \\ 1 & 4 & 1 \end{bmatrix}$ i porównując wynik z wynik górną macierzą trójkątną $\boldsymbol{U}$ uzyskaną w Matlabie z rozkładu LU przez `[L,U]=lu(A)`.

żeby ich użyć. W systemie zasilanym napięciem $235$ V pojawiają się spadki napięcia spowodowane prądami pobieranymi przez odbiorniki $Y_4$, $Y_5$, $Y_6$ płynącymi przez admitancje linii $Y_1$, $Y_2$, $Y_3$, jednak napięcia u wszystkich odbiorców zawierają się w granicach od $225$ do $233$ V. Przyjmijmy, że maksymalne dopuszczalne odchylenie napięcia od wartości znamionowej $230$ V wynosi $\pm 10$ V. Zakład energetyczny pozwolił na zainstalowanie na końcu linii dodatkowego odbiornika $Y_7$ dużej mocy, który spowodował, że w wyniku spadków napięcia wzdłuż linii zasilającej wartości napięć u odbiorców znalazły się poza dopuszczalnym zakresem. Typową praktyką w takiej sytuacji jest regulacja ( zwiększenie) napięcia zasilającego przy pomocy odczepów transformatora rozdzielczego. Równania opisujące obwód można przedstawić w postaci macierzowej, a następnie rozwiązywać go dla coraz większych wartości napięć[3] (które występuje w wektorze po prawej stronie równania), aż wartości napięć w węzłach sieci osiągną wymagane wartości.




Rysunek 1: Schemat systemu energetycznego do zadania 3.

Wyprowadź na papierze 3 równania węzłowe opisujące obwód z rysunku 1 i zapisz je w postaci macierzowej. Rozwiąż ten układ w Matlabie, korzystając z wartości elementów podanych w pliku `lab03z03.m`, aby poznać wartości napięć węzłowych po włączeniu dodatkowego odbiornika i dla napięcia na transformatorze $E = 235\mathrm{V}$. Przeczytaj opis funkcji `lu` i rozłóż za jej pomocą macierz admitancyjną $\boldsymbol{Y}$ na macierze $\boldsymbol{L}$ i $\boldsymbol{U}$. Następnie w pętli zwiększaj napięcie $E$ źródła i rozwiązuj kolejno układy równań o postaci takiej jak (6) i (7), do momentu gdy napięcie w węźle $3$ przekroczy dolną wartość dopuszczoną przez normę. Układy te rozwiązuj z wykorzystaniem operatora $\backslash$. Dzięki niemu Matlab automatycznie dobierze najszybszą metodę rozwiązania równania odpowiednią do postaci macierzy (tutaj trójkątnych). Sprawdź czas wykonywania całej pętli i dobierz tak skok, o jaki będzie zwiększane napięcie w każdym przebiegu pętli, żeby czas wykonywania wynosił około 10 sekund. Następnie zmień w pętli sposób rozwiązywania równania tak, żeby nie korzystać z rozkładu macierzy admitancji $\boldsymbol{Y}$ na $\boldsymbol{L}$ i $\boldsymbol{U}$, lecz aby zmusić Matlaba do każdorazowego przeprowadzenia pełnej dekompozycji LU macierzy admitancji czyli bezpośredniego rozwiązywania równania $\boldsymbol{YV} = \boldsymbol{J}$ (nadal z użyciem operatora $\backslash$). Oblicz stosunek czasu wykonywania obliczeń obydwoma metodami.

## 4 Dekompozycja Cholesky'ego w języku C — 2 pkt

Rozwiąż układ równań z poprzedniego zadania (dla $E = 235\mathrm{V}$ bez dodatkowej admitancji $Y_7$) wykorzystując dekompozycję Cholesky'ego. Dekompozycja ta jest polega na rozkładzie macierzy $\boldsymbol{A}$ na iloczyn $\boldsymbol{LL}^T$ i można ją stosować tylko gdy macierz $\boldsymbol{A}$ jest dodatnio określona (tak jak macierz $Y$ z poprzedniego zadania).

W środowisku CodeBlocks utwórz nowy projekt konsolowy. Następnie do pliku głównego `main.c` wklej szkielet programu z pliku `lab03z04.c` i uzupełnij brakujące części zgodnie z poleceniami podanymi dalej. Program jest prawie kompletny, a jako przykład pozostawiłem w nim dwie linie rozwiązujące układ równań przez dekompozycję LU. Uzupełnij brakujące fragmenty, a następnie zakomentuj dwie w.w. linie i uruchom program. Porównaj otrzymany wynik z wynikiem uzyskanym w poprzednim zadaniu w Matlabie. Następnie zmień znak dowolnej liczby na przekątnej macierzy admitancyjnej $\boldsymbol{Y}$ i uruchom program. Na podstawie wyświetlonego komunikatu wyjaśnij dlaczego program nie zadziałał. Spróbuj w Matlabie wykonać dekompozycję Cholesky'ego (funkcja `chol`) dla tak samo zmodyfikowanej macierzy i sprawdź rezultat.

Poniżej opisałem niezbędne funkcje potrzebne do zmodyfikowania programu w C.

`gsl_matrix_view gsl_matrix_view_array (double * base, size_t n1, size_t n2)` zwraca macierz (a dokładniej widok macierzy) o rozmiarach `n1`×`n2` zawierające liczby z tablicy danych

[3] Tak naprawdę zmiana odczepów transformatora pozwala zazwyczaj na na mało dokładną (zgrubną) regulację napięcia.

base. Widoki macierzy czyli dane typu `gsl_matrix_view` są obiektami tymczasowymi, które służą do operowania na fragmentach macierzy (podmacierzach) lub do przedstawienia jednowymiarowych tablic danych w postaci macierzy (jak w tym zadaniu).

`gsl_vector_view gsl_vector_view_array (double *base, size_t n)` zwraca wektor (a dokładniej widok wektora) o długości `n` zawierający liczby z tablicy danych `base`. Widoki wektorów czyli dane typu `gsl_vector_view` są bardzo podobne do opisanych wyżej widoków macierzy.

UWAGA!!! Widoki macierzy i wektorów nie tworzą nowego obiektu przez jego skopiowanie, a jedynie wskazują na istniejące w pamięci dane, dlatego modyfikacja elementu widoku spowoduje modyfikację elementu oryginalnej macierzy bądź tablicy danych. Z tego samego powodu dopóki widok jest w użyciu nie możemy zniszczyć macierzy bądź tablicy, na bazie których zbudowano widok macierzy. Widoki mogą być podawane jako argument każdej funkcji operującej na macierzach lub wektorach w postaci referencji, np. `&A.matrix` dla widoku macierzy i `&b.vector` dla widoku wektora. Sposób bezpośredni czyli był stosowany na poprzednich zajęciach.

`int gsl_linalg_cholesky_decomp (gsl_matrix * A)` wykonuje dekompozycję Cholesky'ego dodatnio określonej macierzy `A` i w niej umieszcza wynik. Zwraca kod błędu lub zero w przypadku prawidłowego działania. W przypadku użycia jako argumentu widoków macierzy (`gsl_matrix_view`) zamiast macierzy (`gsl_matrix`) należy odwołać się do widoku przez referencję np. `gsl_linalg_cholesky_decomp(&A.matrix)`.

`int gsl_linalg_cholesky_solve (const gsl_matrix * cholesky, const gsl_vector * b, gsl_vector * x)` rozwiązuje, w oparciu o dekompozycję Cholesky'ego, układ równań $\boldsymbol{Ax} = \boldsymbol{b}$ używając podstawienia wstecz. `cholesky` zawiera zdekomponowaną (opisaną wcześniej funkcją) macierz $\boldsymbol{A}$, `b` zawiera wektor $\boldsymbol{b}$, a rozwiązanie umieszczane jest wektorze `x`.

W dokumentacji do GSL-a jest znacznie więcej funkcji operujących na macierzach. Są to np. funkcje do kopiowania, zamieniania wektorów lub macierzy, do wyszukiwania największych i najmniejszych elementów macierzy, dodawanie, odejmowanie, mnozenie przez skalar macierzy lub wektorów i wiele innych. GSL zawier również szereg metod bezpośrednich rozwiązywania układów równań. Metody te są sugerowane do rozwiązywania małych układów. Do wielkich układów sugerowane jest użycie biblioteki LAPACK (http://www.netlib.org/lapack/)/ Biblioteka ta została napisana dla Fortrana77, ale znacznie później została przeportowana do języka C, gdzie występuje pod nazwą CLAPACK.

## 5 Badanie zbieżności iteracyjnej metody rozwiązywania układów równań — 2 pkt

Metody iteracyjne są przeznaczone do rozwiązywania wielkich układów równań. Przybliżają one z zadaną dokładnością prawdziwe rozwiązanie układu. Użytkownik ma zatem wpływ na czas obliczeń, gdzyż może przerwać obliczenia w dowolnej iteracji. Należy mieć świadomość, że im mniej iteracji wykonano, tym wynik uzyskany gorzej przybliża wynik dokładny (prawdziwy). W celu zbadania zachowania metod iteracyjnych wynik dokładny może być wyznaczony numerycznie jedną z metod bezpośrednich poznanych wcześniej[4].

Najłatwiejszą do zaimplementowania i zarazem najwolniejszą z iteracyjnych metod rozwiązywania układów równań liniowych jest metoda Jacobiego bedąca wariantem metody iteracji prostej.

Metoda Jacobiego wykorzystuje rozkład macierzy

$$\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U} \tag{8}$$

gdzie $\boldsymbol{D}$ to macierz zawierająca przekątną macierzy $\boldsymbol{A}$ (poza tym same zera), $\boldsymbol{L}$ to dolna połowa macierzy $\boldsymbol{A}$ (z zerami na i ponad przekątną), $\boldsymbol{U}$ to górna połowa macierzy $\boldsymbol{A}$ (z zerami na i pod przekątną). Macierze $\boldsymbol{L}$ i $\boldsymbol{U}$ nie są wynikiem rozkładu LU. Po dokonaniu rozkładu możemy układ równań zapisać

$$\boldsymbol{Ax} = (\boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U})\,\boldsymbol{x} = \boldsymbol{b} \tag{9}$$

wobec tego

$$\boldsymbol{Dx} = \boldsymbol{b} - (\boldsymbol{L} + \boldsymbol{U})\,\boldsymbol{x} \tag{10}$$

[4] przy założeniu braku błędów wynikających z reprezentacji liczb

Zakładając, że w każdej iteracji obliczamy nowe przybliżenie rozwiązania $\boldsymbol{x}$ można napisać

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{D}^{-1}\left(\boldsymbol{b} - (\boldsymbol{L}+\boldsymbol{U})\,\boldsymbol{x}^{(k)}\right) \tag{11}$$

Wzór jest łatwy w stosowaniu, jednak trzeba pamiętać, że metody iteracyjne nie zawsze są zbieżne. Warunkiem *wystarczającym* zbieżności metody iteracji prostej jest aby

$$\|\boldsymbol{B}\| \leqslant 1 \tag{12}$$

gdzie

$$\boldsymbol{B} = \boldsymbol{D}^{-1}\,(\boldsymbol{L}+\boldsymbol{U}) \tag{13}$$

a $\|\boldsymbol{B}\|$ jest dowolną normą macierzy $\boldsymbol{B}$.

Niestety, zdarzają się przypadki, gdy metoda jest zbieżna, a warunek wystarczający (12) nie jest spełniony, bo nie jest to warunek konieczny. Trudniejszy do sprawdzenia, warunek konieczny, jest dany wzorem

$$\rho(\boldsymbol{B}) < 1 \tag{14}$$

gdzie $\rho(\boldsymbol{B}) = max|\lambda_i|$ jest promieniem spektralnym macierzy $\boldsymbol{B}$, a $\lambda_i$ jest $i$–tą wartością własna macierzy.

W tym zadaniu wykorzystasz metodę Jacobiego do rozwiązania poniższego układu równań,

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 1 & 0 & -2 \\ 0 & 1 & 0 & 0 & -1 & 0 & 1 & 1 \\ 0 & 0 & 4 & -1 & 0 & 2 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & 0 & -1 & -1 & 0 & 0 \\ 1 & 0 & 2 & 0 & -1 & -2 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 2 & 0 \\ -2 & 1 & 0 & 1 & 0 & 0 & 0 & 14 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{15}$$

Otwórz plik `lab03z05.m`, w którym znajdują się dane czyli macierz $\boldsymbol{A}$ i wektor $\boldsymbol{b}$. Rozkład macierzy $\boldsymbol{A}$ na $\boldsymbol{L}$, $\boldsymbol{D}$, $\boldsymbol{U}$ uzyskasz przy pomocy funkcji `tril`, `diag`, `triu`. Maksymalna ilość iteracji `maxiter` wynosi 100. Iteracje wykonuj aż błąd będzie mniejszy od `epsilon`. Za błąd rozwiązania w danej iteracji przyjmij normę różnicy $||\boldsymbol{x}_d - \boldsymbol{x}^{(k)}||$, gdzie $\boldsymbol{x}_d$ to obliczone wcześniej metodą bezpośrednią rozwiązanie dokładne równe $\begin{bmatrix} 1,5 & -7 & 3,5 & 18 & 4,5 & 1,5 & 13 & -0,5 \end{bmatrix}^T$, a $\boldsymbol{x}^{(k)}$ to przybliżenie rozwiązania w $k$-tej iteracji. Normę obliczysz funkcją `norm`. Wyznaczone w kolejnych iteracjach wektory $\boldsymbol{x}^{(k)}$ przechowuj jako kolumny macierzy, będą one potrzebne do wyrysowania przebiegu procesu iteracyjnego. Za początkową wartość wektora $\boldsymbol{x}$ przyjmij $\boldsymbol{x}_d + 5$. Po zakończeniu obliczeń wypisz liczbę iteracji oraz błąd popełniony podczas ostatniej iteracji. Wyrysuj na jednym wykresie kolejne przybliżenia wszystkich niewiadomych układu na tle wartości rozwiązań dokładnych, tak jak pokazano na rysunku 2.

Rozwiąż układ dla początkowych wartości wektora $\boldsymbol{x}$ równych wektorowi $\boldsymbol{b}$ oraz wektorowi zerowemu. W którym przypadku szybciej została osiągnięta zadana dokładność?

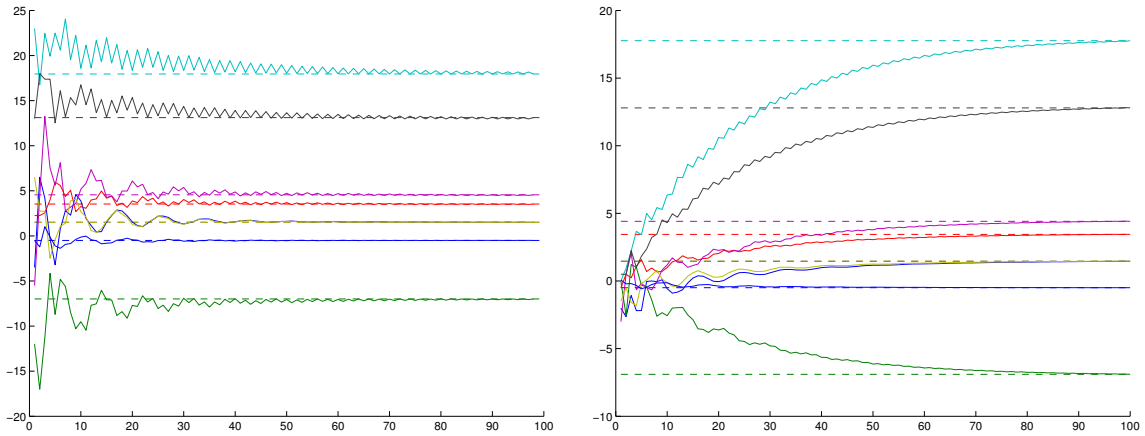
Zmień lewy dolny element ($a_{8,1}$) macierzy $\boldsymbol{A}$ na wartość $-6$. Zobacz jak teraz przebiega proces iteracyjny dla początkowego wektora $\boldsymbol{x} = \boldsymbol{0}$ i sprawdź błąd w ostatniej iteracji, dla poprzednio podanych startowych wartości wektora.

Powróć do oryginalnej macierzy $\boldsymbol{A}$ ($a_{8,1} = -2$), a następnie sprawdź zachowanie algorytmu zmieniając jej prawy dolny element ($a_{8,8}$) kolejno na wartości na wartość $-6; -8; -10$. Obserwując przebiegi rozwiązań na wykresie zwróć uwagę na rozbieżność oraz sposób dochodzenia do wartości prawdziwych z przeregulowaniem (oscylacyjny) lub bez przeregulowania (jak odpowiedź obiektu inercyjnego).

Dla trzech w.w. przypadków sprawdź czy spełnione są warunki wystarczający (13) i konieczny (14) zbieżności metody. Do obliczenia (13) użyj funkcji `norm`, a do obliczenia (14) użyj funkcji `max`, `abs` i `eig`. Sprawdź dla trzech przypadków uwarunkowanie macierzy $\boldsymbol{A}$.

## Na następne zajęcia

Zapoznac się z funkcjami biblioteki GSL oraz Matlaba przeznaczonymi do interpolacji funkcji. Opis tych funkcji GSL znajdziesz pod adresem http://www.gnu.org/software/gsl/manual/html_node/Interpolation.html. Przeczytaj pomoc do matlabowskich funkcji `polyfit`, `polyval`, `interp1` oraz dokumentację do Spline Toolbox i funkcji `csapi`, `csape`, `fnval`. Zobacz również programy demonstaracyjne do Spline Toolbox. Demo uruchamia się w Matlabie poleceniem `demo`.




Rysunek 2: Przebieg rozwiązywania układu ośmiu równań z zadania 5 metodą iteracyjną dla dwóch różnych punktów startowych. Na osi poziomej są numery iteracji, a na pionowej wartości niewiadomych.