Raport - temat 1

Układy równań liniowych i nieliniowych. Zera wielomianów.

Zadanie 5

Przemysław Wiczołek
gr. F3

17.10.2018

1. Wstęp teoretyczny

Treść zadania: *Rozwiązanie układu równań w dziedzinie zespolonej. Wyjściowy układ sprowadzamy do układu o elementach rzeczywistych i rozwiązujemy go metodą eliminacji Gaussa z częściowym wyborem elementu głównego.*

Układ równań $Ax = b$ można sprowadzić do układu rzeczywistego w podany sposób:

$$\begin{cases} A = B + iC \\ x = y + iz \\ b = c + id \end{cases}$$
$$\implies \quad (B + iC)(y + iz) = c + id$$
$$\implies By - Cz + iBz + iCy = c + id$$

Co daje:

$$\begin{cases} By - Cz = c \\ Bz + Cy = d \end{cases}$$

A więc zwykły układ równań liniowych w dziedzinie rzeczywistej.

Następnie można posłużyć się metodą eliminacji Gaussa z częściowym wyborem, to znaczy przy każdej iteracji metody wybieramy wiersz k o maksymalnej wartości w kolumnie j, a potem zamieniamy j-ty wiersz z k-tym:

$$\textbf{max} => \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{in} & b_i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right] \implies \left[\begin{array}{cccc|c} a_{i1} & a_{i2} & \cdots & a_{in} & b_i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right]$$

Potem wykonujemy eliminację:

$$\left[\begin{array}{cccc|c} a_{i1} & a_{i2} & \cdots & a_{in} & b_i \\ 0 & a_{22} - a_{21}/a_{i1} * a_{i2} & \cdots & a_{2n} - a_{21}/a_{i1} * a_{in} & b_2 - a_{21}/a_{i1} * bi \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & a_{n2} - a_{n1}/a_{i1} * a_{i2} & \cdots & a_{nn} - a_{n1}/a_{i1} * a_{in} & b_n - a_{n1}/a_{i1} * bi \end{array}\right]$$

$$\implies \cdots \implies \left[\begin{array}{cccc|c} a_{i1} & a_{i2} & \cdots & a_{in} & b_i \\ 0 & * & \cdots & * & * \\ \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & \cdots & * & * \end{array}\right]$$

2. Opis metody

Sposób implementacji

Implementacja została przeprowadzona w języku Matlab. Kod został rozdzielony na dwie[1] funkcje: *GEPP_cmplx(A,b)* oraz *GEPP(A,b)*. Testowanie odbywało się w skrypcie *rand_examples.m* z wykorzystaniem losowej macierzy 1200×1200 z wartościami zespolonymi $z = a + bi$, gdzie $a, b \in [10^{-6}, 10^{6}]$.

Funkcja *GEPP(A,b)*, czyli funkcja obliczająca wynik równania metodą eliminacji Gaussa z częściowym wyborem została zaimplementowana zarówno w języku Matlab jak i w C w celu porównania szybkości obliczania i dokładności wyników.

Funkcja *GEPP_cmplx(A,b)* została napisana w języku Matlab, ale była oddzielna dla obu implementacji, ze względu na niemożnośc przekazywania argumentów pionowych w do funkcji w języku C. Sprowadzała ona podany układ do dziedziny rzeczywistej i wywoływała funkcję GEPP do dalszych obliczeń, a później zwracała wynik. Była to główna funkcja programu.

W skrypcie *time_graph.m* został zamieszczony wykres rozmiaru macierzy do czasu.

Warunki stopu

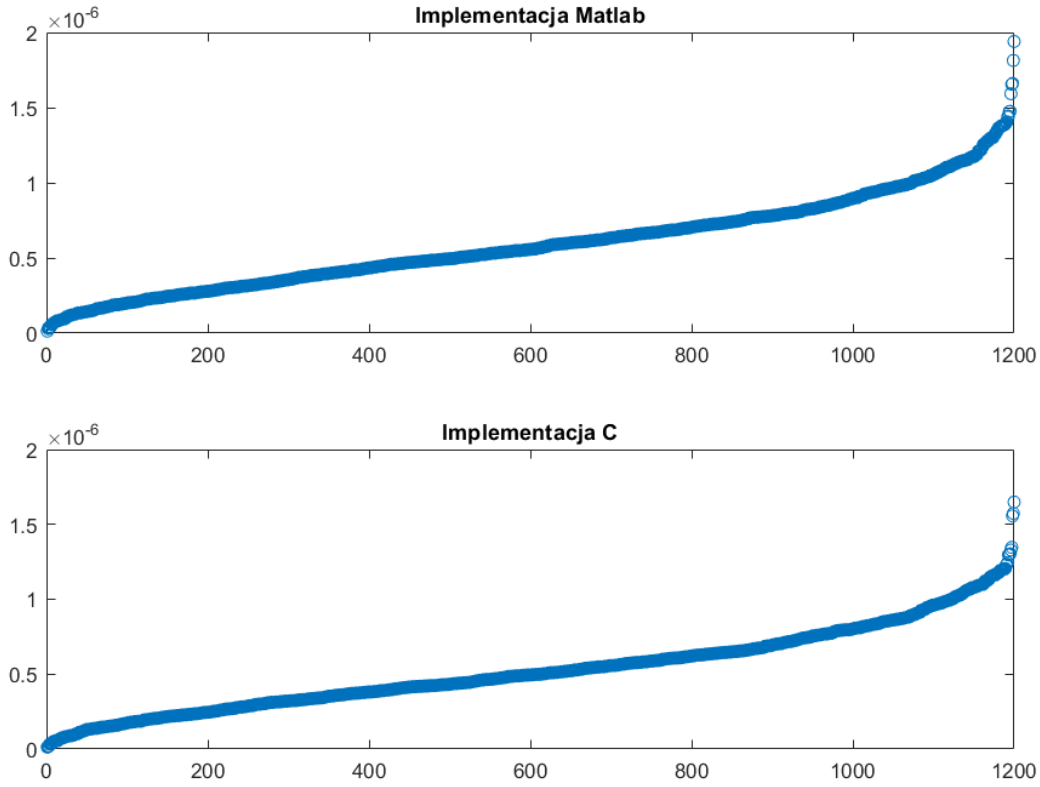
Program nie zostanie wykonany, jeżeli:

- Ilość argumentów wejściowych lub wyjściowych jest niewłaściwa
- Macierz A nie jest kwadratowa
- Rozmiar wektora b jest inny niz rozmiar macierzy A
- Wyznacznik macierzy A jest bliski zeru

[1] Właściwie cztery - dwie dla GEPP w języku C i analogicznie dla Matlaba

3. Wyniki

Dokładność



Posortowane wektory błędu bezwzględnego

Jak można zobaczyć na wykresie błąd w wyniku jest rzędu 10^{-6} dla obu implementacji. Wynika z tego, że w domyślnej implementacji Matlab korzysta z podobnego kodowania liczb zmiennoprzecinkowych jak język C.

Czas

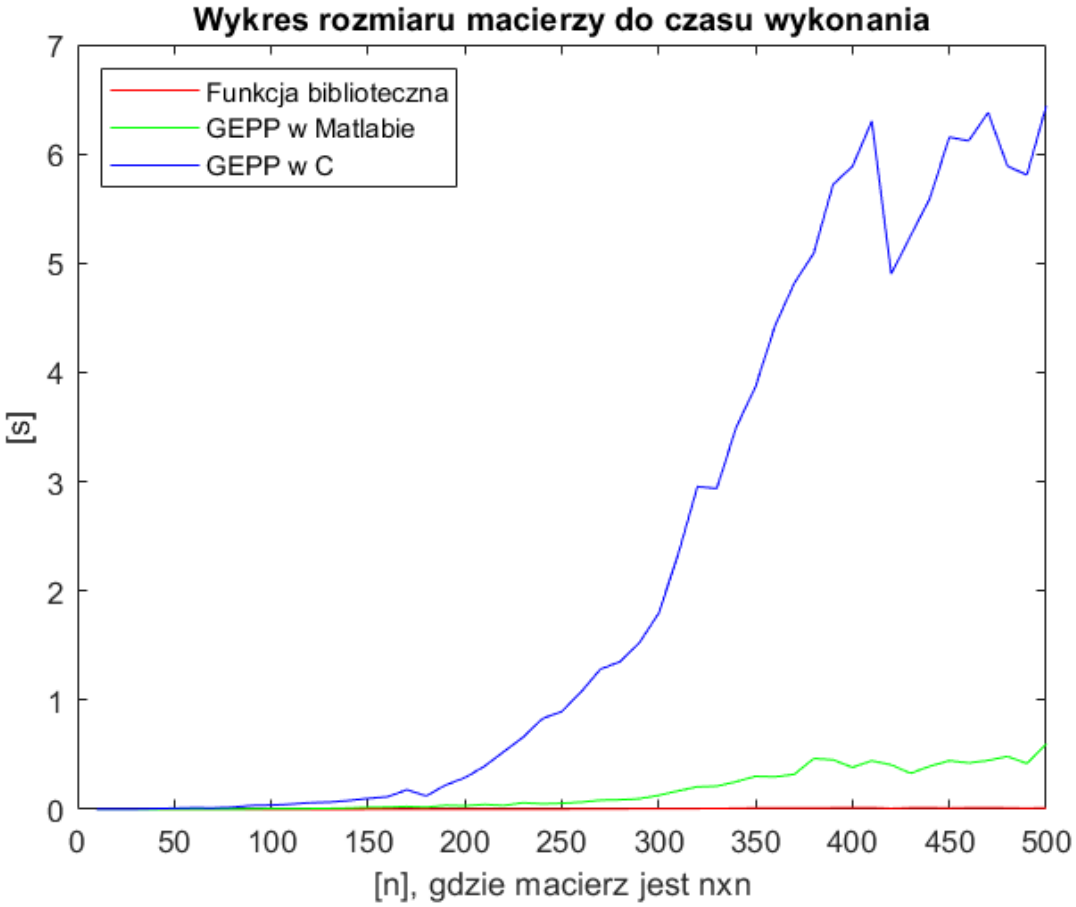
```
% interval -> [c, d]
% matrix size -> n
c = -1e6;
d = -c;
n = 1200;

A = (c-d) .* (rand(n, n)+c + 1i*(rand(n, n)+c));
b = (c-d) .* (rand(n, 1)+c + 1i*(rand(n, 1)+c));


tic
precise_result = A\b;
toc;

tic
c_result = GEPP_cmplx_c(A, b);
toc

tic
my_result = GEPP_cmplx(A, b);
toc
```

Kod skryptu do testów

```
>> examples
Elapsed time is 0.130219 seconds.
Elapsed time is 6.677157 seconds.
Elapsed time is 100.206531 seconds.
```

Czas wykonania poszczególnych funkcji

Jak widać moja implementacja w języku C jest około **15** razy szybsza od tej samej utworzonej w Matlabie. Oczywiście funkcja biblioteczna działa nieporównywalnie lepiej od obu moich realizacji.



Przykłady obliczeniowe (również w pliku *examples.m*)

- Przykład 1

$$A = \begin{bmatrix} 1.0000 - 0.7071i & 1.0000 - 1.4142i & 1.0000 + 0.7071i & 1.0000 - 0.2929i \\ 1.0000 + 1.4142i & 2.0000 - 0.0000i & 1.0000 - 1.4142i & 2.0000 + 0.5000i \\ 1.0000 - 0.7071i & 1.0000 + 1.4142i & 3.0000 + 0.7071i & 1.0000 - 1.7071i \\ 1.0000 + 1.1716i & 2.0000 - 2.0000i & 1.0000 + 6.8284i & 4.0000 - 5.5000i \end{bmatrix}$$

$$b = \begin{bmatrix} 16.0000 + 1.0000i \\ 5.0000 + 8.0000i \\ 9.0000 + 27.0000i \\ 4.0000 + 64.0000i \end{bmatrix}$$

$$x = \begin{bmatrix} 2.9100 + 1.7734i \\ 6.1863 + 5.3519i \\ 4.0495 + 0.0504i \\ -5.9409 + 0.0272i \end{bmatrix}$$

	Błąd	Czas
C	2.6922e-15	0.001537 s
Matlab	3.9721e-15	0.005112 s

- Przykład 2

$$A = \begin{bmatrix} 1.0000 + 1.0000i & 1.0000 + 0.5000i & 1.0000 + 0.3333i & 1.0000 + 0.2500i \\ 1.0000 + 0.5000i & 2.0000 + 1.0000i & 2.0000 + 0.6667i & 2.0000 + 0.5000i \\ 0.0000 + 0.3333i & 2.0000 + 0.6667i & 3.0000 + 1.0000i & 3.0000 + 0.7500i \\ 0.0000 + 0.2500i & 0.0000 + 0.5000i & 3.0000 + 0.7500i & 4.0000 + 1.0000i \end{bmatrix}$$

$$b = \begin{bmatrix} 1.0000 - 2.0000i \\ 0.5000 + 1.0000i \\ 0.3333 + 0.0000i \\ 0.2500 + 0.0000i \end{bmatrix}$$

$$x = \begin{bmatrix} -1.8462 - 2.2308i \\ 4.0933 + 2.2043i \\ -10.7781 - 0.8237i \\ 8.1773 + 0.1981i \end{bmatrix}$$

	Błąd	Czas
C	1.7154e-14	0.000981 s
Matlab	1.0842e-14	0.003475 s

- Przykład 3

$$A = \begin{bmatrix} 16i & 120i & 240i & 140i \\ 120i & 1200i & 2700i & 1680i \\ 240i & 2700i & 6480i & 4200i \\ 140i & 1680i & 4200i & 2800i \end{bmatrix}$$

$$b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

$$x = \begin{bmatrix} 4i \\ 2.7167i \\ 2.1i \\ 1.7214i \end{bmatrix}$$

	Błąd	Czas
C	2.3981e-14	0.000214 s
Matlab	2.0872e-14	0.000320 s

4. Podsumowanie

Po przeprowadzeniu analizy wnioskuję, że obliczenia na liczbach zespolonych w programie Matlab nie są dużo bardziej skomplikowane w porównaniu do liczb rzeczywistych.

Oczywiście ilość danych jest zwiększona czterokrotnie w porównaniu do zwykłego układu równań, aczkolwiek nie wpływa to w żaden sposób na realizację samej funkcji *GEPP*. Korzystając z funkcji bibliotecznych prawie wcale nie odczujemy różnicy w szybkości.

Ciekawą sprawą jest, że nawet zwektoryzowany(na miarę moich możliwości) kod w języku Matlab jest znacznie wolniejszy od tej samej, najprostszej implementacji w języku C. Prawdopodobnie wynika, to z mojej małej znajomości narzędzia MathWorks, chociaż nie szczycę się również dużą wiedzą o C.

Mam przeświadczenie, że język C jest dużo lepszym narzędziem do pisania prostych wydajnych implementacji, szczególnie, że MathWorks oferuje taką możliwość udostępniając swoje API. Nie sprawiło mi wiele trudności, żeby skorzystać z dostępnej dokumentacji, a program w C właściwie nie różnił się od zwykłego.