MOwNiT – Układy równań liniowych - metody bezpośrednie

Przygotował:
Szymon Budziak

Problem 1:

Elementy macierzy A o wymiarze $n\,x\,n$ są określone wzorem:

$$\begin{cases} a_{1j} = 1 \\ a_{ij} = \dfrac{1}{i+j-1} \quad dla \;\; i \neq 1 \end{cases} \qquad i, j = 1, \ldots, n$$

Przyjmij wektor x jako dowolną n−elementową permutację ze zbioru $\{1, \; -1\}$ i oblicz wektor **b**. Następnie metodą eliminacji Gaussa rozwiąż układ równań liniowych **Ax=b** (przyjmując jako niewiadomą wektor **x**). Przyjmij różną precyzję dla znanych wartości macierzy **A** i wektora **b**. Sprawdź, jak błędy zaokrągleń zaburzają rozwiązanie dla różnych rozmiarów układu (porównaj – zgodnie z wybraną normą – wektory **x** obliczony z **x** zadany). Przeprowadź eksperymenty dla różnych rozmiarów układu.

Rozmiary układu, które zostały przetestowane w tym zadaniu to:
3, 4, …, 80. Przyjęta prezycja to float32, float64, float 128 z biblioteki numpy.

Wyniki do zadania pierwszego

n	float type	norm	A	B	X
3	<class 'numpy.float32'>	$0e^{+0}$	[[1. 1. 1.] [0.5 0.33333334 0.25] [0.33333334 0.25 0.2]]	[1. 0.41666666 0.28333335]	[1. -1. 1.]
3	<class 'numpy.float64'>	$0e^{+00}$	[[1. 1. 1.] [0.5 0.33333333 0.25] [0.33333333 0.25 0.2]]	[1. 0.41666667 0.28333333]	[1. -1. 1.]
3	<class 'numpy.float128'>	$0e^{+0}$	[[1. 1. 1.] [0.5 0.33333333 0.25] [0.33333333 0.25 0.2]]	[1. 0.41666667 0.28333333]	[1. -1. 1.]
4	<class 'numpy.float32'>	$6.646e^{-15}$	[[1. 1. 1. 1.] [0.5 0.33333334 0.25 0.2] [0.33333334 0.25 0.2 0.16666667] [0.25 0.2 0.16666667 0.14285715]]	[0. 0.21666665 0.11666667 0.07380952]	[1. -1. 1. -1.]
4	<class 'numpy.float64'>	$3.018e^{-13}$	[[1. 1. 1. 1.] [0.5 0.33333333 0.25 0.2] [0.33333333 0.25 0.2 0.16666667] [0.25 0.2 0.16666667 0.14285714]]	[0. 0.21666667 0.11666667 0.07380952]	[1. -1. 1. -1.]
4	<class 'numpy.float128'>	$4.187e^{-17}$	[[1. 1. 1. 1.] [0.5 0.33333333 0.25 0.2] [0.33333333 0.25 0.2 0.16666667] [0.25 0.2 0.16666667 0.14285714]]	[0. 0.21666667 0.11666667 0.07380952]	[1. -1. 1. -1.]
5	<class 'numpy.float32'>	$7.282e^{-13}$	[[1. 1. 1. 1. 1.] [0.5 0.33333334 0.25 0.2 0.16666667] [0.33333334 0.25 0.2 0.16666667 0.14285715] [0.25 0.2 0.16666667 0.14285715 0.125] [0.2 0.16666667 0.14285715 0.125 0.11111111]]	[1. 0.38333333 0.25952382 0.19880952 0.16230159]	[1. -1. 1. -1. 1.]
5	<class 'numpy.float64'>	$9.229e^{-12}$	[[1. 1. 1. 1. 1.] [0.5 0.33333333 0.25 0.2 0.16666667] [0.33333333 0.25 0.2 0.16666667 0.14285714] [0.25 0.2 0.16666667 0.14285714 0.125] [0.2 0.16666667 0.14285714 0.125 0.11111111]]	[1. 0.38333333 0.25952381 0.19880952 0.16230159]	[1. -1. 1. -1. 1.]
5	<class 'numpy.float128'>	$8.671e^{-16}$	[[1. 1. 1. 1. 1.] [0.5 0.33333333 0.25 0.2 0.16666667] [0.33333333 0.25 0.2 0.16666667 0.14285714] [0.25 0.2 0.16666667 0.14285714 0.125] [0.2 0.16666667 0.14285714 0.125 0.11111111]]	[1. 0.38333333 0.25952381 0.19880952 0.16230159]	[1. -1. 1. -1. 1.]

6	<class 'numpy.float32'>	$5.440e^{-11}$	[[1. 1. 1. 1. 1. 1.] [0.5 0.33333334 0.25 0.2 0.16666667 0.14285715] [0.33333334 0.25 0.2 0.16666667 0.14285715 0.125] [0.25 0.2 0.16666667 0.14285715 0.125 0.11111111] [0.2 0.16666667 0.14285715 0.125 0.11111111 0.1] [0.16666667 0.14285715 0.125 0.11111111 0.1 0.09090909]]	[0. 0.24047618 0.13452382 0.08769841 0.06230159 0.04678932]	[1. -1. 1. -1. 1. -1.]
6	<class 'numpy.float64'>	$3.637e^{-10}$	[[1. 1. 1. 1. 1. 1.] [0.5 0.33333333 0.25 0.2 0.16666667 0.14285714] [0.33333333 0.25 0.2 0.16666667 0.14285714 0.125] [0.25 0.2 0.16666667 0.14285714 0.125 0.11111111] [0.2 0.16666667 0.14285714 0.125 0.11111111 0.1] [0.16666667 0.14285714 0.125 0.11111111 0.1 0.09090909]]	[0. 0.24047619 0.13452381 0.08769841 0.06230159 0.04678932]	[1. -1. 1. -1. 1. -1.]
6	<class 'numpy.float128'>	$3.012e^{-16}$	[[1. 1. 1. 1. 1. 1.] [0.5 0.33333333 0.25 0.2 0.16666667 0.14285714] [0.33333333 0.25 0.2 0.16666667 0.14285714 0.125] [0.25 0.2 0.16666667 0.14285714 0.125 0.11111111] [0.2 0.16666667 0.14285714 0.125 0.11111111 0.1] [0.16666667 0.14285714 0.125 0.11111111 0.1 0.09090909]]	[0. 0.24047619 0.13452381 0.08769841 0.06230159 0.04678932]	[1. -1. 1. -1. 1. -1.]

Tabela 1: 12 pierwszych wyników z zadania pierwszego

Wnioski

Możemy zauważyć, że błędy zwiększają się wraz z rosnącym rozmiarem układu. Również można zaobserwować, że błąd jest różny dla różnych precyzji. Dla typów float32 oraz float64 błędy te są znacznie większe niż błędy dla typu float128.

Problem 2:

Powtórz eksperyment dla macierzy zadanej wzorem:

$$\begin{cases} a_{ij} = \dfrac{2i}{j} \quad dla \quad j \geq i \\ a_{ij} = a_{ji} \quad dla \quad j < i \end{cases} \qquad i, j = 1, \ldots, n$$

Porównaj wyniki z tym, co otrzymano w przypadku układu z punktu 1). Spróbuj uzasadnić, skąd biorą się różnice w wynikach. Sprawdź uwarunkowanie obu układów.

Rozmiary układu, które zostały przetestowane w tym zadaniu to: 3, 4, …, 80. Przyjęta prezycja to float32, float64, float 128 z biblioteki numpy.

Wyniki do zadania drugiego

n	float type	norm	A	B	X
3	<class 'numpy.float32'>	$0e^{+0}$	[[2. 1. 0.6666667] [1. 2. 1.3333334] [0.6666667 1.3333334 2.]]	[1.66666669 0.33333337 1.33333331]	[1. -1. 1.]
3	<class 'numpy.float64'>	$3.140e^{-16}$	[[2. 1. 0.66666667] [1. 2. 1.33333333] [0.66666667 1.33333333 2.]]	[1.66666667 0.33333333 1.33333333]	[1. -1. 1.]
3	<class 'numpy.float128'>	$1.212e^{-19}$	[[2. 1. 0.66666667] [1. 2. 1.33333333] [0.66666667 1.33333333 2.]]	[1.66666667 0.33333333 1.33333333]	[1. -1. 1.]
4	<class 'numpy.float32'>	$0e^{+0}$	[[2. 1. 0.6666667 0.5] [1. 2. 1.3333334 1.] [0.6666667 1.3333334 2. 1.5] [0.5 1. 1.5 2.]]	[1.16666669 -0.66666663 -0.16666669 -1.]	[1. -1. 1. -1.]
4	<class 'numpy.float64'>	$2.482e^{-16}$	[[2. 1. 0.66666667 0.5] [1. 2. 1.33333333 1.] [0.66666667 1.33333333 2. 1.5] [0.5 1. 1.5 2.]]	[1.16666667 -0.66666667 -0.16666667 -1.]	[1. -1. 1. -1.]

4	<class 'numpy.float128'>	$1.212e^{-19}$	[[2. 1. 0.66666667 0.5] [1. 2. 1.33333333 1.] [0.66666667 1.33333333 2. 1.5] [0.5 1. 1.5 2.]]	[1.16666667 -0.66666667 -0.16666667 -1.]	[1. -1. 1. -1.]
5	<class 'numpy.float32'>	$2.482e^{-16}$	[[2. 1. 0.6666667 0.5 0.4] [1. 2. 1.3333334 1. 0.8] [0.6666667 1.3333334 2. 1.5 1.2] [0.5 1. 1.5 2. 1.6] [0.4 0.8 1.2 1.6 2.]]	[1.56666669 0.13333338 1.03333336 0.60000002 1.20000002]	[1. -1. 1. -1. 1.]
5	<class 'numpy.float64'>	$4.154e^{-16}$	[[2. 1. 0.66666667 0.5 0.4] [1. 2. 1.33333333 1. 0.8] [0.66666667 1.33333333 2. 1.5 1.2] [0.5 1. 1.5 2. 1.6] [0.4 0.8 1.2 1.6 2.]]	[1.56666667 0.13333333 1.03333333 0.6 1.2]	[1. -1. 1. -1. 1.]
5	<class 'numpy.float128'>	$2.484e^{-19}$	[[2. 1. 0.66666667 0.5 0.4] [1. 2. 1.33333333 1. 0.8] [0.66666667 1.33333333 2. 1.5 1.2] [0.5 1. 1.5 2. 1.6] [0.4 0.8 1.2 1.6 2.]]	[1.56666667 0.13333333 1.03333333 0.6 1.2]	[1. -1. 1. -1. 1.]
6	<class 'numpy.float32'>	$3.140e^{-16}$	[[2. 1. 0.6666667 0.5 0.4 0.33333334] [1. 2. 1.3333334 1. 0.8 0.6666667] [0.6666667 1.3333334 2. 1.5 1.2 1.] [0.5 1. 1.5 2. 1.6 1.3333334] [0.4 0.8 1.2 1.6 2. 1.6666666] [0.33333334 0.6666667 1. 1.3333334 1.6666666 2.]]	[1.23333335 -0.5333333 0.03333336 -0.73333335 -0.46666661 -1.00000009]	[1. -1. 1. -1. 1. -1.]
6	<class 'numpy.float64'>	$9.742e^{-16}$	[[2. 1. 0.66666667 0.5 0.4 0.33333333] [1. 2. 1.33333333 1. 0.8 0.66666667] [0.66666667 1.33333333 2. 1.5 1.2 1.] [0.5 1. 1.5 2. 1.6 1.33333333] [0.4 0.8 1.2 1.6 2. 1.66666667] [0.33333333 0.66666667 1. 1.33333333 1.66666667 2.]]	[1.23333333 -0.53333333 0.03333333 -0.73333333 -0.46666667 -1.]	[1. -1. 1. -1. 1. -1.]
6	<class 'numpy.float128'>	$1.626e^{-19}$	[[2. 1. 0.66666667 0.5 0.4 0.33333333] [1. 2. 1.33333333 1. 0.8 0.66666667] [0.66666667 1.33333333 2. 1.5 1.2 1.] [0.5 1. 1.5 2. 1.6 1.33333333] [0.4 0.8 1.2 1.6 2. 1.66666667] [0.33333333 0.66666667 1. 1.33333333 1.66666667 2.]]	[1.23333333 -0.53333333 0.03333333 -0.73333333 -0.46666667 -1.]	[1. -1. 1. -1. 1. -1.]

Tabela 2: 12 pierwszych wyników z zadania drugiego

Porównanie wyników z zadania pierwszego z wynikami z zadania drugiego

n	float type	ex 1 norm	ex 2 norm	ex 1 X given	ex 2 X given	ex 1 X	ex 2 X
3	<class 'numpy.float32'>	$0e^{+0}$	$0e^{+0}$	[1 -1 1]	[1 -1 1]	[1. -1. 1.]	[1. -1. 1.]
3	<class 'numpy.float64'>	$0e^{+00}$	$3.140e^{-16}$	[1 -1 1]	[1 -1 1]	[1. -1. 1.]	[1. -1. 1.]
3	<class 'numpy.float128'>	$0e^{+0}$	$1.212e^{-19}$	[1 -1 1]	[1 -1 1]	[1. -1. 1.]	[1. -1. 1.]
4	<class 'numpy.float32'>	$6.646e^{-15}$	$0e^{+0}$	[1 -1 1 -1]	[1 -1 1 -1]	[1. -1. 1. -1.]	[1. -1. 1. -1.]
4	<class 'numpy.float64'>	$3.018e^{-13}$	$2.482e^{-16}$	[1 -1 1 -1]	[1 -1 1 -1]	[1. -1. 1. -1.]	[1. -1. 1. -1.]
4	<class 'numpy.float128'>	$4.187e^{-17}$	$1.212e^{-19}$	[1 -1 1 -1]	[1 -1 1 -1]	[1. -1. 1. -1.]	[1. -1. 1. -1.]
5	<class 'numpy.float32'>	$7.282e^{-13}$	$2.482e^{-16}$	[1 -1 1 -1 1]	[1 -1 1 -1 1]	[1. -1. 1. -1. 1.]	[1. -1. 1. -1. 1.]
5	<class 'numpy.float64'>	$9.229e^{-12}$	$4.154e^{-16}$	[1 -1 1 -1 1]	[1 -1 1 -1 1]	[1. -1. 1. -1. 1.]	[1. -1. 1. -1. 1.]
5	<class 'numpy.float128'>	$8.671e^{-16}$	$2.484e^{-19}$	[1 -1 1 -1 1]	[1 -1 1 -1 1]	[1. -1. 1. -1. 1.]	[1. -1. 1. -1. 1.]
6	<class 'numpy.float32'>	$5.440e^{-11}$	$3.140e^{-16}$	[1 -1 1 -1 1 -1]	[1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1.]	[1. -1. 1. -1. 1. -1.]
6	<class 'numpy.float64'>	$3.637e^{-10}$	$9.742e^{-16}$	[1 -1 1 -1 1 -1]	[1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1.]	[1. -1. 1. -1. 1. -1.]
6	<class 'numpy.float128'>	$3.012e^{-16}$	$1.626e^{-19}$	[1 -1 1 -1 1 -1]	[1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1.]	[1. -1. 1. -1. 1. -1.]
7	<class 'numpy.float32'>	$3.695e^{-09}$	$2.482e^{-16}$	[1 -1 1 -1 1 -1 1]	[1 -1 1 -1 1 -1 1]	[1. -1. 1. -1. 1. -1. 1.]	[1. -1. 1. -1. 1. -1. 1.]
7	<class 'numpy.float64'>	$1.360e^{-08}$	$1.694e^{-15}$	[1 -1 1 -1 1 -1 1]	[1 -1 1 -1 1 -1 1]	[1. -1. 1. -0.99999999 0.99999999 -0.99999999 1.]	[1. -1. 1. -1. 1. -1. 1.]
7	<class 'numpy.float128'>	$8.865e^{-13}$	$1.131e^{-18}$	[1 -1 1 -1 1 -1 1]	[1 -1 1 -1 1 -1 1]	[1. -1. 1. -1. 1. -1. 1.]	[1. -1. 1. -1. 1. -1. 1.]

8	<class 'numpy.float32'>	$6.747e^{-09}$	$6.080e^{-16}$	[1 -1 1 -1 1 -1 1 -1]	[1 -1 1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1. 1. -1.]	[1. -1. 1. -1. 1. -1. 1. -1.]
8	<class 'numpy.float64'>	$1.203e^{-07}$	$4.672e^{-15}$	[1 -1 1 -1 1 -1 1 -1]	[1 -1 1 -1 1 -1 1 -1]	[1. -1. 1.00000001 -1.00000003 1.00000007 -1.00000008 1.00000005 -1.00000001]	[1. -1. 1. -1. 1. -1. 1. -1.]
8	<class 'numpy.float128'>	$5.106e^{-11}$	$1.280e^{-18}$	[1 -1 1 -1 1 -1 1 -1]	[1 -1 1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1. 1. -1.]	[1. -1. 1. -1. 1. -1. 1. -1.]

Tabela 3: 18 pierwszych wyników z porównania wyników z zadania pierwszego z zadaniem drugim

Wnioski

Dzięki tabelom możemy zauważyć, że jedną z przyczyn złych wyników dla układu równań z pierwszego problemu jest metoda wybierania elementu wiodącego. W implementacji z Problemu 1 algorytm eliminacji Gaussa jako zawsze jako element wiodący wybiera kolejne elementy przekątnej macierzy. W przypadku macierzy z Problemu 1, elementy na przekątnej redukują się na tyle, że osiągają wartość rzędu nawet 10^{-18}. W przypadku macierzy z Problemu 2 jest to wielkość rzędu 10^{-2}, więc jest tutaj znaczna różnica. Mała wartość w przypadku Problemu 2 stanowi problem, ponieważ poszczególne wiersze w metodzie eliminacji Gaussa są dzielone przez element wiodący, to znaczy mnożone przez jego odwrotność. Jeżeli element ten ma małą wartość (< 1), to wiersze mnożone są przez duża wartość , więc błędy stają się duże w stosunku do współczynników oryginalnej macierzy. Możemy zauważyć, że wskaźniki uwarunkowania dla macierzy w problemie 1 są znacznie większe od wskaźników uwarunkowania w problemie 2. Oznacza to, że niewielki błąd znacznie wpływa na wyniki.

Problem 3:

Powtórz eksperyment dla jednej z macierzy zadanej wzorem poniżej (macierz i parametry podane w zadaniu indywidualnym). Następnie rozwiąż układ metodą przeznaczoną do rozwiązywania układów z macierzą trójdiagonalną. Porównaj wyniki otrzymane dwoma metodami (czas, dokładność obliczeń i zajętość pamięci) dla różnych rozmiarów układu. Przy porównywaniu czasów należy pominąć czas tworzenia układu. Opisz, jak w metodzie dla układów z macierzą trójdiagonalną przechowywano i wykorzystywano macierz A.

$(m,k$ - parametry zadania):

$$\begin{cases} a_{i,i} = -m \cdot i - k \\ a_{i,i+1} = i \\ a_{i,i-1} = \dfrac{m}{i} \quad dla \;\; i > 1 \\ a_{i,j} = 0 \qquad dla \;\; j < i-1 \quad oraz \quad j > i+1 \end{cases} \qquad i, j = 1, \ldots, n$$

parametry zadania: **k = 8, m = 3**.

Rozmiary układu, które zostały przetestowane w tym zadaniu to:
3, 4, …, 80.

Wyniki do zadania trzeciego

n	given X	gaussian X	gaussian norm	gaussian time	thomas X	thomas norm	thomas time
3	[1 -1 1]	[1. -1. 1.]	2.220e^{-16}	0,0002269	[1. -1. 1.]	2.220e^{-16}	0,00003558
4	[1 -1 1 -1]	[1. -1. 1. -1.]	2.220e^{-16}	0,0002043	[1. -1. 1. -1.]	2.220e^{-16}	0,00002810
5	[1 -1 1 -1 1]	[1. -1. 1. -1. 1.]	2.220e^{-16}	0,0007462	[1. -1. 1. -1. 1.]	2.220e^{-16}	0,00003398
6	[1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,0004420	[1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,00005049
7	[1 -1 1 -1 1 -1 1]	[1. -1. 1. -1. 1. -1. 1.]	2.482e^{-16}	0,0004010	[1. -1. 1. -1. 1. -1. 1.]	2.482e^{-16}	0,00003878
8	[1 -1 1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,0003461	[1. -1. 1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,00004120
9	[1 -1 1 -1 1 -1 1 -1 1]	[1. -1. 1. -1. 1. -1. 1. -1. 1.]	2.220e^{-16}	0,0003666	[1. -1. 1. -1. 1. -1. 1. -1. 1.]	2.220e^{-16}	0,00004317
10	[1 -1 1 -1 1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,0004321	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,00004760
11	[1 -1 1 -1 1 -1 1 -1 1 -1 1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	2.220e^{-16}	0,0004914	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	2.220e^{-16}	0,00004898
12	[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,0005996	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,00005383
13	[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	2.220e^{-16}	0,0006640	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	2.220e^{-16}	0,0000570
14	[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,0005951	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	2.220e^{-16}	0,0000304
15	[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	3.140e^{-16}	0,0004147	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	3.140e^{-16}	0,00003125
16	[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	3.510e^{-16}	0,0005396	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	3.510e^{-16}	0,00003511
17	[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	4.710e^{-16}	0,0005404	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	4.710e^{-16}	0,00003630
18	[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	4.299e^{-16}	0,0006642	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	4.299e^{-16}	0,00003610
19	[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	4.710e^{-16}	0,0006317	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1.]	4.710e^{-16}	0,00003860
20	[1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1]	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	4.154e^{-16}	0,0008322	[1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1. 1. -1.]	4.154e^{-16}	0,00004184

Tabela 4: 18 pierwszych wyników z zadania trzeciego

Wnioski

Z Tabeli 4 możemy zaobserwować, że niezależnie której metody użyjemy to błędy (norm) są takie same. Wyniki te są przewidywalne, ponieważ metody Gaussa i Thomasa są bardzo podobne, a w niektórych miejscach identyczne. Metoda Thomasa ogranicza się tylko do działania na elementach trójdiagonali. Jeśli chodzi o czasy działania to metoda Thomasa jest znacznie szybsza od metody eliminacji Gaussa. Nie jest to jednak zaskakujące, ponieważ wykonuje ona znacznie mniej operacji, ograniczając się tylko do trójdiagonali macierzy. Metoda gaussa jest skutecznym i prostym sposobem na rozwiązywanie układów równań liniowych. W niektórych przypadkach jest ona jednak wolna z powodu błędów dokładności, które są spowodowane słabym uwarunkowaniem układów wejściowych lub sposobu wybierania elementu wejściowego. W przypadku, gdy macierz jest trójdiagonalna, to warto stosować zamiast metody eliminacji Gaussa metodę Thomasa. Jest to uproszczona wersja algorytmu Gaussa, która daje wyniki o tej samej dokładności jednak działa znacznie szybciej.

Literatura

- Wykład nr 8 dr Rycerz z przedmiotu MOwNiT
- Wikipedia na temat algorytmu Gaussa oraz algorytmu Thomasa