Feliks Bańka, 323733

MNUM – Projekt 1

Zadanie 1

Napisać solwer, czyli uniwersalną procedurę, rozwiązującą układ n równań liniowych Ax = b, wykorzystując podaną metodę (parametry wejściowe: A, b, n /opcjonalnie, bo można odczytać za pomocą instrukcji size/; parametry wyjściowe: rozwiązanie ˜x, błąd rozwiązania ε1 = ‖A˜x − b‖2 oraz, jeśli zadanie tego dotyczy, macierze otrzymane w rezultacie faktoryzacji). Nie sprawdzać w środku solwera, czy macierz A jest kwadratowa, nieosobliwa, wektor b niezerowy, wymiary się zgadzają, itp. Proszę zastosować następnie swój solwer w programie do rozwiązania obydwu (jeśli można) lub jednego z podanych niżej układów równań dla n = 5, 10, 25, 50, 100, 200.

Metoda: eliminacji Gaussa z częściowym wyborem elementu głównego

Proszę wykonać wykres (wykresy) zależności błędu ε1 od liczby równań n.

Metoda eliminacji Gaussa jest jedną z metod skończonych rozwiązywania układów równań liniowych. Oznacza to, że wynik uzyskujemy po wykonaniu skończonej liczbie przekształceń, która zależy od rozmiaru układu równań.

Algorytm ten dzieli się na dwa kroki:

1. Eliminacja zmiennych
   Metoda ta polega na przekształceniu macierzy w taki sposób, aby uzyskać macierz górną trójkątną. Uzyskujemy ją poprzez wybranie jednego z równań elementu (równania wyjściowego), przy pomocy którego wyzerujemy współczynniki znajdujące się w tej same kolumnie poniżej równania wyjściowego, odejmując od kolejnych równań w_i znajdujących się poniżej iloczyn l_{i1} oraz równania wyjściowego w_1.

   $$l_{i1} = \frac{a_{i1}}{a_{11}}$$

   $$w_i = w_i - l_{i1} * w_1$$

   Stosujemy tą metodę, dla kolejnych kolumn zmieniając równanie wyjściowe i umieszczając je na samej górze spośród jeszcze niewybranych równań. W eliminacji Gaussa z częściowym wyborem elementu głównego, jako równanie wyjściowe wybieramy to, którego element w aktualnie zerowanej kolumnie ma największą wartość bezwzględną.

2. Postępowanie odwrotne

   Rozwiązujemy układ równań z macierzą odwrotną. Z ostatniego równania odczytujemy bezpośrednio $\mathbf{X}_n$. Podstawiamy tę wartość do równania n-1 i wyliczamy $\mathbf{X}_{n-1}$. Postępując analogicznie dla wszystkich równań otrzymamy rozwiązanie układu Ax=B.

Niepewność pomiarowa

Błąd rozwiązania obliczamy wzorem:

$$\varepsilon_1 = \| A\tilde{x} - b \|_2$$

Gdzie:

A - macierz współczynników

b - macierz dołączona

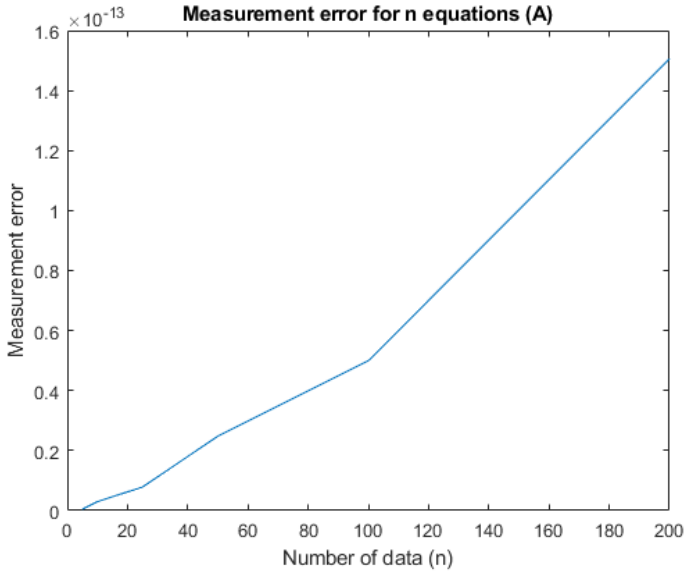
$\tilde{x}$ - wektor rozwiązań układu równań (wynik)

ε_1 – błąd rozwiązania

Wykresy i wartości zależności błędu ε_1 od liczby równań n.

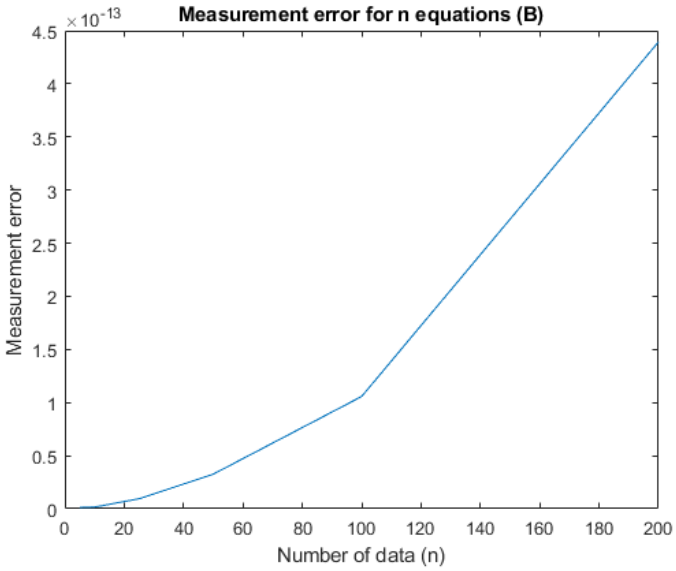
Podpunkt A

```
Time of solving 5 equations:
Elapsed time is 0.000028 seconds.
Measurement error: 4.440892e-16

Time of solving 10 equations:
Elapsed time is 0.000027 seconds.
Measurement error: 2.979041e-15

Time of solving 25 equations:
Elapsed time is 0.000046 seconds.
Measurement error: 7.869281e-15

Time of solving 50 equations:
Elapsed time is 0.000161 seconds.
Measurement error: 2.479354e-14

Time of solving 100 equations:
Elapsed time is 0.000946 seconds.
Measurement error: 4.992993e-14

Time of solving 200 equations:
Elapsed time is 0.007513 seconds.
Measurement error: 1.505855e-13
```



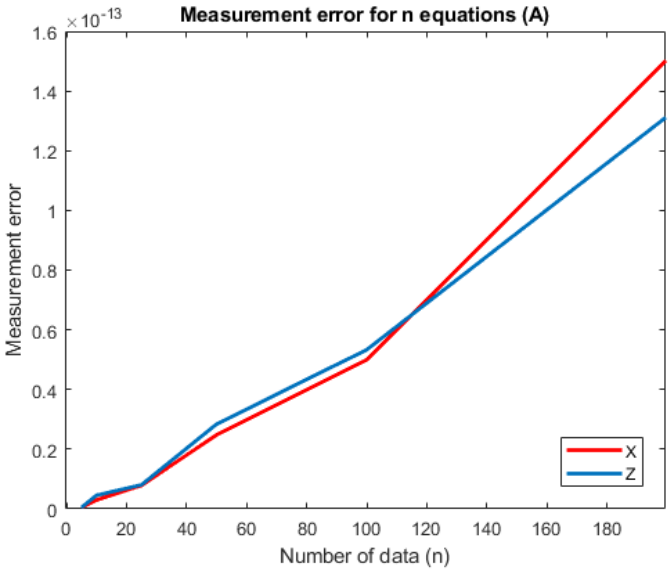
Podpunkt B

```
Time of solving 5 equations:
Elapsed time is 0.000033 seconds.
Measurement error: 1.332268e-15

Time of solving 10 equations:
Elapsed time is 0.000027 seconds.
Measurement error: 1.601186e-15

Time of solving 25 equations:
Elapsed time is 0.000082 seconds.
Measurement error: 9.483150e-15

Time of solving 50 equations:
Elapsed time is 0.000305 seconds.
Measurement error: 3.261260e-14

Time of solving 100 equations:
Elapsed time is 0.001641 seconds.
Measurement error: 1.057343e-13

Time of solving 200 equations:
Elapsed time is 0.007454 seconds.
Measurement error: 4.394612e-13
```



Jak widać, błąd rozwiązania rośnie szybciej niż wzrost liczby układów równań.

Ocenić nasz algorytm możemy również porównując go do już wbudowanego w Matlab'a algorytmu Z=linsolve(A,b) służącego do dowiązywania układów liniowych. Poniżej prezentuje się porównanie błędy pomiarowego dla danych z podpunktu A. Jak widać różnice w wydajności obu algorytmów są niewielkie. Mało tego przy małej liczbie danych nasz algorytm Gaussa jest bardziej efektywny.



Zadanie 2

Napisać solwer rozwiązujący układ n równań liniowych Ax = b, wykorzystując metodę iteracyjną Jacobiego. Jego parametry wejściowe powinny zawierać, poza

wymienionymi w p.1, także wartość graniczną ε 2 błędu między kolejnymi przybliżeniami rozwiązania, liczonego jako norma euklidesowa z ich różnicy. Przyjąć jako kryterium stopu warunek ε 2 = 10−6 ≙ 1e − 6.
Proszę zastosować swój solwer do rozwiązania właściwego układu równań spośród przedstawionych poniżej dla n = 5, 10, 25, 50, 100, 200.
Proszę sprawdzić dokładność rozwiązania licząc także błąd ε 1 i dla każdego układu równań wykonać rysunek zależności tego błędu od liczby równań n. Jeśli był rozwiązywany ten sam układ równań, co w p. 1, proszę porównać czasy obliczeń dla rożnych algorytmów i wymiarów zadań.

Metoda Jacobiego jest jedną z metod iteracyjnych rozwiązywania układów równań liniowych. Oznacza to, że zaczynając od założonego (lub danego) rozwiązania, poprawiamy je w kolejnych krokach (iteracjach), przez co zbliża się do rozwiązania. W tych metodach poprawiamy rozwiązanie tak długo, aż nie będzie ono wystarczająco dokładne, a tolerancję tą ustalamy odgórnie (ε_2). Używamy je najczęściej, gdy problem jest wielowymiarowy, a macierz A jest rozrzedzona.

Aby można było skorzystać z metody Jacobiego, musi być spełniony warunek dostateczny zbieżności macierzy. Oznacza to, iż macierz musi być silnie diagonalna wierszowo lub kolumnowo, co przedstawia się wzorem (i analogicznie dla kolumnowej):

$$|a_{ii}| > \sum_{k=1,\ k\neq 1}^{n} |a_{ik}|$$

Wyznaczania wyniki o dokładności ε

Na początku musimy zdekomponować macierz na sumę macierzy A=L+D+U, gdzie:

L - macierz poddiagonalna

D - macierz diagonalna

U - macierz naddiagonalna.

Wtedy po wykonaniu kilku przekształceń:

Ax=b

(L+D+U) x=b

Dx = − (L + U) x + b, i = 0, 1, 2, ...

$Dx_{(j)} = -(L + U)\, x_{(i)} + b$, gdzie j=i+1

$x_{(j)} = -D^{-1}(L + U)\, x_{(i)} + D^{-1}b$

W taki sposób uzyskujemy wzór na dokładniejszy wynik kolejnej iteracji algorytmu.

Wyznaczenie wartości granicznej ε_2

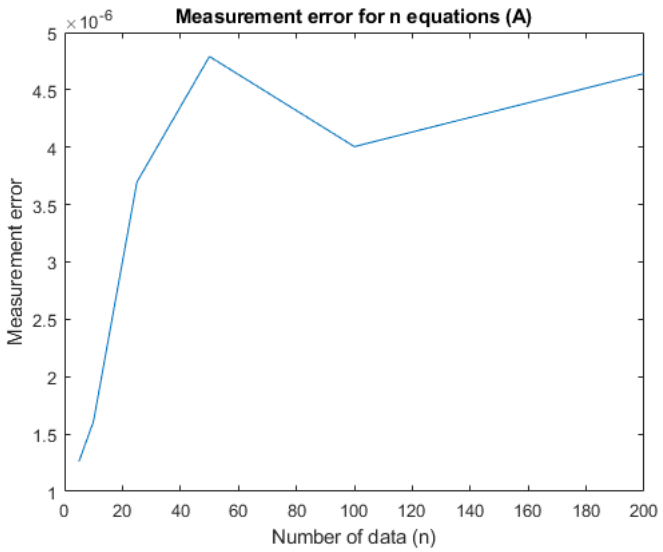
Wartość graniczna (warunek stopu iteracji) liczymy jako normę euklidesową z różnicy kolejnych przybliżeń wg wzoru:

$$\varepsilon_2 = \| x(j) - x(i) \|_2$$

Wykresy i wartości zależności błędu ε_1 od liczby równań n.

Podpunkt A

```
Time of solving 5 equations:
Elapsed time is 0.000048 seconds.
Measurement error: 1.261109e-06

Time of solving 10 equations:
Elapsed time is 0.000047 seconds.
Measurement error: 1.609824e-06

Time of solving 25 equations:
Elapsed time is 0.000082 seconds.
Measurement error: 3.697170e-06

Time of solving 50 equations:
Elapsed time is 0.000506 seconds.
Measurement error: 4.790611e-06

Time of solving 100 equations:
Elapsed time is 0.001805 seconds.
Measurement error: 4.005342e-06

Time of solving 200 equations:
Elapsed time is 0.009419 seconds.
Measurement error: 4.641916e-06
```



Podpunkt B

Macierz w podpunkcie B nie spełnia warunek dostateczny zbieżności algorytmu nie jest spełniony. Co pokazuje, że niestety, mimo iż jest to szybka metoda nie można jej zastosować w każdym przypadku.

Porównanie czasów obliczeń dla obu algorytmów

Jak widać na wykresie, w przypadku danych z podpunktu A metoda eliminacji Gaussa jest znacząco szybsza niezależnie od liczby danych.