LAPORAN PRAKTIKUM MATA KULIAH

METODE NUMERIK

PRAKTIKUM 4 – GALAT/ERROR

DISUSUN OLEH:

M0521003 – ADI PRASETYA

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SEBELAS MARET

2022

BAB I

ANALISIS *SOURCE CODE*

1. **Analisis *Source code* Metode Gauss**

```
% M0521003 - Adi Prasetya
function x = EliminasiGauss (A,b)
tic
[n,l] = size(A);
for i = 1 : n-1,
    [pivot,k] = max(abs(A(i:n, i)));
    if (k ~= 1)
        temp1 = A(i, :);
        temp2 = b(i, :);
        A(i,:) = A(i+k-1,:);
        b(i,:) = b(i+k-1,:);
        A(i+k-1,:) = temp1;
        b(i+k-1,:) = temp2;
    end
    for (h = i+1 : n),
        m = A(h,i)/A(i,i);
        A(h,:) = A(h,:) - m*A(i,:);
        b(h,:) = b(h,:) - m*b(i,:);
    end
end
fprintf('Hasil forward elimination:\n');
A
x(n,:) = b(n,:) / A(n,n);
for (i = n:-1:1),
    x(i,:) = (b(i,:)-A(i,i+1:n)*x(i+1:n,:)) / A(i,i);
end
toc
```

Dalam *source code* tersebut, terdapat perhitungan untuk mencari nilai **a, b, c,** dan **d** menggunakan metode Gauss. Pada **line 2**, terdapat ***function* x** yang berisi fungsi **EliminasiGauss (A,b)** yang akan dipanggil di *command window* kemudian terdapat (A,b) yang akan dideklarasikan nilainya dalam command window. Pada line **4** terdapat *assignment* yang berarti bahwa n adalah ukuran dari matriks A. Lalu, pada line **5 B** hingga **20** terdapat **for loop** yang dimulai dari i = 1 hingga n-1 dimana di dalamnya terdapat *source code* untuk melakukan perhitungan menjadi eselon baris. Terdapat **pivot** untuk menyimpan nilai maksimum dan kemudian dilakukan pertukaran matriks A dan b pada posisi tertentu dengan menggunakan *function* **swap**. Kemudian akan ditampilkan hasil forward elimination dan estimasi waktu komputasi yang digunakan.

2. Analisis *Source code* Metode Gauss-Jordan

```
% M0521003 - Adi Prasetya
function x = EliminasiGaussJordan(A,b)
tic
[n,1] = size(A);
for i = 1 : n-1,
    [pivot,k] = max(abs(A(i:n, i)));
    if (k ~= 1)
        temp1 = A(i, :);
        temp2 = b(i, :);
        A(i,:) = A(i+k-1,:);
        b(i,:) = b(i+k-1,:);
        A(i+k-1,:) = temp1;
        b(i+k-1,:) = temp2;
    end
    for (h = i+1 : n),
        m = A(h,i)/A(i,i);
        A(h,:) = A(h,:) - m*A(i,:);
        b(h,:) = b(h,:) - m*b(i,:);
    end
end
fprintf('Hasil forward elimination:\n')
A
for i = n:-1:2
    for h = i-1:-1:1
        m = A(h,i)/A(i,i);
        A(h,:) = A(h,:)-m*A(i,:);
        b(h,:) = b(h,:)-m*b(i,:);
    end
end
fprintf('Hasil backwards elimination:\n')
A
for i = 1:n
    x(i,:) = b(i,:)/A(i,i);
end
toc
```

Source code dalam Metode Gauss Jordan sama seperti *source code* pada metode Gauss hanya saja ditambahkan *code* untuk melakukan *backwards elimination***. Dalam** *source code* tersebut, terdapat perhitungan untuk mencari nilai **a, b, c,** dan **d** menggunakan metode Gauss. Pada **line 2**, terdapat ***function*** **x** yang berisi fungsi **EliminasiGaussJordan(A,b)** yang akan dipanggil di *command window* kemudian terdapat (A,b) yang akan dideklarasikan nilainya dalam command window. Pada line **4** terdapat *assignment* yang berarti bahwa n adalah ukuran dari matriks A. Lalu, pada line **5 B** hingga **20** terdapat **for loop** yang dimulai dari i = 1 hingga n-1 dimana di dalamnya terdapat *source code* untuk melakukan perhitungan menjadi eselon baris. Terdapat **pivot** untuk menyimpan nilai maksimum dan kemudian dilakukan pertukaran matriks A dan b pada posisi tertentu dengan menggunakan *function* **swap**. Kemudian akan dilakukan operasi *forward elimination*, lalu dilakukan operasi *backward elimination*. Setelah itu, hasil akhir akan ditampilkan ke layar disertai dengan estimasi waktu komputasinya.

3. Analisis *Source code* LU Dekomposisi

```
% M0521003 - Adi Prasetya
function x = LU_Solusi (A,b)
tic;
fprintf('Hasil dekomposisi LU\n')
[L,U,b] = LU_fwdelim(A,b)
[n,m] = size(L);
z = zeros(n,1);
x = zeros(n,1);
z(1) = b(1)/L(1,1);
for i = 2:n,
    z(i) = (b(i) - L(i, 1:i-1)*z(1:i-1)) / L(i,i);
end
x(n) = z(n)/U(n,n);
for i = n-1:-1:1,
    x(i) = (z(i) - U(i,i+1:n)*x(i+1:n)) / U(i,i);
end
toc;
```

Dalam *source code* tersebut dilakukan pencarian nilai utuk matriks Lower dan Upper terlebih dahulu. Terdapat *function* **LU_Solusi (A,b)** dengan dua parameter dimana A adalah matriks A dan b adalah matriks b. Kemudian terdapat *function* **LU_fwdelim(A,b)** untuk menentukan matriks *Lower* dan *Upper*. Kemudian terdapat matriks zeros dimana semua elemennya bernilai nol. Kemudian, dilakukan operasi untuk menentukan hasil dari operasi LU Dekomposisi.

4. Analisis *Source code* Metode Jacobi

```
% M0521003 - Adi Prasetya
function x = IterasiJacobi(A,b,tol,max_iter);
tic;
[n,m] = size(A);
for i = 1:n,
    xlama(i) = b(i)/A(i,i);
end
xlama = xlama';
C = -A;
for i = 1:n,
    C(i,i) = 0.0;
    C(i,:) = C(i,:)/A(i,i);
    d(i,1) = xlama(i);
end
i = 1;
while (i <= max_iter)
    xbaru = C*xlama + d;
    if (abs(xbaru - xlama) <= tol)
        x = xbaru;
        disp ('Jacobi method konverge');
        toc;
        return;
    else
        xlama = xbaru;
    end
##    disp([i xbaru']);
    i = i + 1;
end
disp ('Jacobi method not konverge');
toc;
```

Dalam *source code* tersebut, terdapat *function* **IterasiJacobi(A,b,tol, max_iter**) dimana A adalah matriks A, b adalah matriks b, tol adalah batas toleransi error yang dapat diterima, dan max_iter adalah jumlah iterasi yang dijalankan. Kemudian, akan mengatur x1 adalah konstanta dari b dibagi dengan elemen diagonalnya lalu baris berubah menjadi kolom. Kemudian, terdapat matriks C dimana C menyimpan masing-masing element dari element yang bukan diagonal lalu nilainya dijadikan negative. Lalu, elemen koefisien akan dibagi oleh elemen diagonal. Kemudian, terdapat **while** untuk melakukan iterasi yang akan berhenti jika memenuhi batas toleransi yang sudah ditentukan maka matriks tersebut akan dinyatakan **konvergen** dan jika sudah mencapai batas iterasi tetapi tidak memenuhi batas toleransi yang sudah ditentukan maka matriks tersebut dinyatakan **tidak konvergen.**

5. Analisis *Source code* Metode Gauss-Seidel

```
1  % M0521003 - Adi Prasetya
2  function x = IterasiGaussSeidel(A,b,tol,max_iter)
3  tic;
4  [n,m] = size(A);
5  for i = 1:n,
6      if (i == 1)
7          x(i) = b(i)/A(i,i);
8      else
9          x(i) = 0;
10     end
11 end
12 x = x';
13 C = -A;
14 for i = 1:n,
15     C(i,i) = 0.0;
16     C(i,:) = C(i,:)/A(i,i);
17     d(i,1) = b(i)/A(i,i);
18 end
19 i = 1;
20 while (i <= max_iter)
21     xlama = x;
22     for j = 1 : n,
23         x(j) = C(j,:)*x + d(j);
24     end
25     if (abs(xlama - x) <= tol)
26         disp ('Gauss Seidel method konverge');
27         toc;
28         return
29     end
30 ##    disp([i x']);
31     i = i + 1;
32 end
33 disp ('Gauss Seidel method not konverge');
34 toc;
```

Dalam *source code* tersebut, terdapat *function* **IterasiGaussSeidel(A,b,tol, max_iter**) dimana A adalah matriks A, b adalah matriks b, tol adalah batas toleransi error yang dapat diterima, dan max_iter adalah jumlah iterasi yang dijalankan. Pada **line 5** hingga **11** digunakan untuk mengasumsikan nilai **x1**. jika memenuhi batas toleransi yang sudah ditentukan maka matriks tersebut akan dinyatakan **konvergen** dan jika sudah mencapai batas iterasi tetapi tidak memenuhi batas toleransi yang sudah ditentukan maka matriks tersebut dinyatakan **tidak konvergen.**

BAB II

ANALISIS PRAKTIKUM

1. Analisis Praktikum

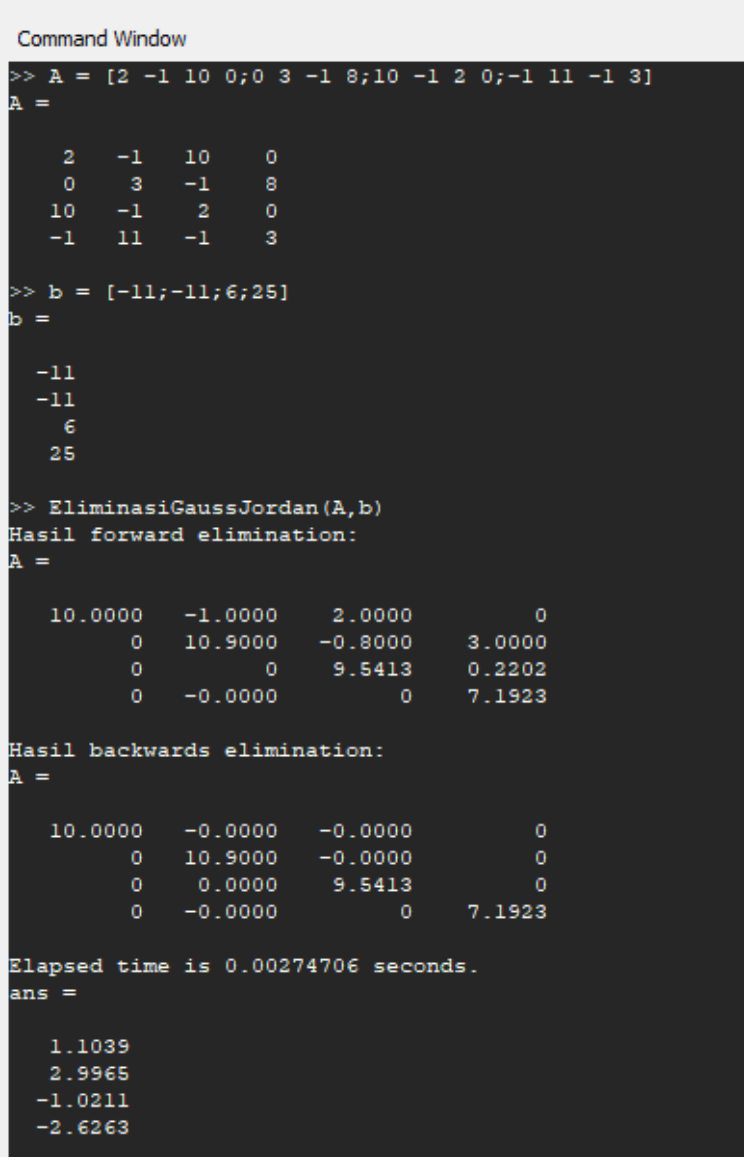
a) Soal A

- Metode Gauss

```
Command Window
>> A = [2 -1 10 0;0 3 -1 8;10 -1 2 0;-1 11 -1 3]
A =

    2    -1    10     0
    0     3    -1     8
   10    -1     2     0
   -1    11    -1     3

>> b = [-11;-11;6;25]
b =

  -11
  -11
    6
   25

>> EliminasiGauss(A,b)
Hasil forward elimination:
A =

   10.0000   -1.0000    2.0000         0
         0   10.9000   -0.8000    3.0000
         0         0    9.5413    0.2202
         0   -0.0000         0    7.1923

Elapsed time is 0.00165319 seconds.
ans =

   1.1039
   2.9965
  -1.0211
  -2.6263
```

Pada command window di atas, dilakukan pendeklarasi dan pengisialisasian dari *matrix* A dan *matrix* B. Kemudian, dilakukan pemanggilan *function* EliminasiGauss(A,b) dengan dua parameter dimana A adalah matriks A dan b adalah matriks B yang digunakan untuk menjalankan *source code* perhitungan sistem persamaan linear dengan metode Gauss. Lalu, akan ditampilkan hasil dari forward elimination dalam *matrix* A. Kemudian, hasil dari a, b, c, dan d akan ditampilkan dalam command window yang disertai dengan waktu komputasinya.

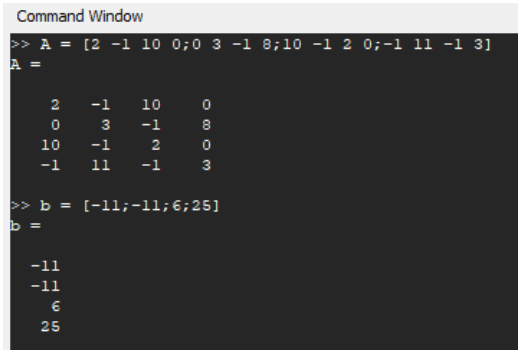
- Metode Gauss-Jordan

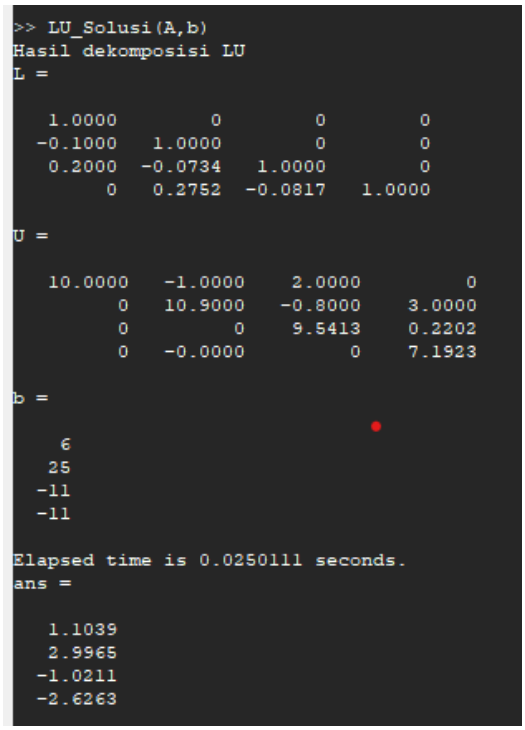


```
Command Window
>> A = [2 -1 10 0;0 3 -1 8;10 -1 2 0;-1 11 -1 3]
A =

    2    -1    10     0
    0     3    -1     8
   10    -1     2     0
   -1    11    -1     3

>> b = [-11;-11;6;25]
b =

  -11
  -11
    6
   25

>> EliminasiGaussJordan(A,b)
Hasil forward elimination:
A =

   10.0000   -1.0000    2.0000         0
         0   10.9000   -0.8000    3.0000
         0         0    9.5413    0.2202
         0   -0.0000         0    7.1923

Hasil backwards elimination:
A =

   10.0000   -0.0000   -0.0000         0
         0   10.9000   -0.0000         0
         0    0.0000    9.5413         0
         0   -0.0000         0    7.1923

Elapsed time is 0.00274706 seconds.
ans =

   1.1039
   2.9965
  -1.0211
  -2.6263
```

Pada command window di atas, dilakukan pendeklarasian dan penginisialisasian dari matriks A dan matriks B. Kemudian, dilakukan pemanggilan *function* EliminasiGaussJordan(A,b) dengan dua parameter dimana A adalah matriks A dan b adalah matriks B. Lalu, dilakukan operasi forward elimination dan hasilnya akan ditampilkan ke layar dalam matriks A. Setelah itu, akan dilakukan operasi backwards elimination dan hasilnya akan ditampilkan ke layar dalam matriks A. Kemudian, hasil berupa nilai dari a, b, c, dan d akan ditampilkan pada command window yang disertai dengan waktu komputasinya.

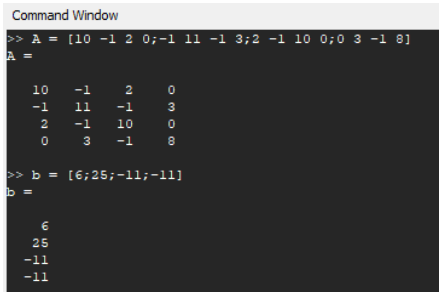
- Metode LU Dekomposisi

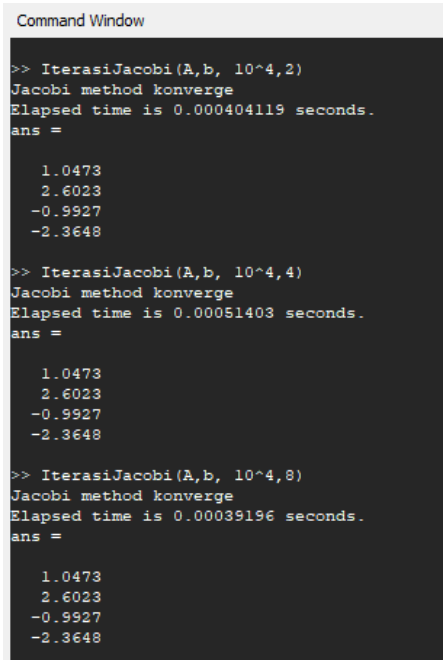




Pada command window di atas, dilakukan pendeklarasi dan pengisialisasian dari *matrix* A dan *matrix* B. Kemudian, dilakukan pemanggilan *function* LU_Solusi(A,b) dengan dua parameter dimana A adalah matriks A dan b adalah matriks B. Matriks A akan dipisah menjadi dua matriks yang berbeda, yaitu matriks L (Lower) dan U (Upper). Kemudian, dilakukan operasi untuk memisahkan matriks A menjadi matriks Lower dan Upper kemudian hasilnya akan ditampilkan ke layar. Setelah itu, dilakukan operasi [L][Z] = [C] untuk menghitung nilai [Z] yang ditampilkan sebagai nilai dari matriks B. Lalu, dilakukan operasi [U][X] = [Z] untuk mendapatkan nilai dari [X] yang kemudian value dari [X] sebagai matriks B yang berupa nilai dari a, b, c, dan d akan ditampilkan pada command window yang disertai dengan waktu komputasinya.

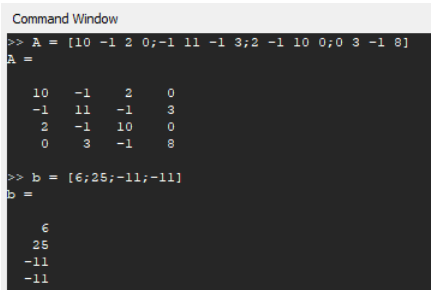
- Metode Jacobi



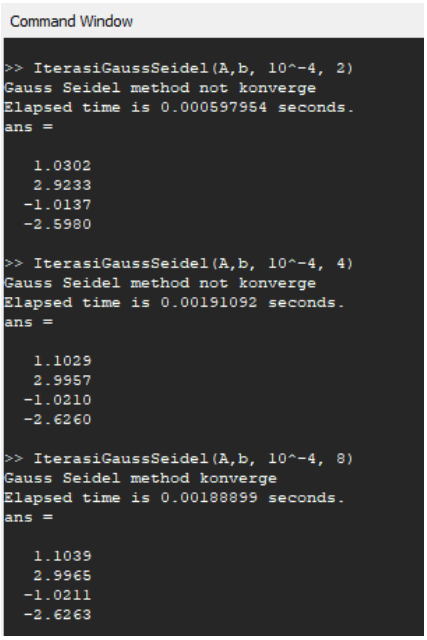


Pada command window di atas, dilakukan pendeklarasi dan pengisialisasian dari *matrix* A dan *matrix* B. Untuk 2 iterasi, dilakukan pemanggilan *function* IterasiJacobi(A,b,10^-4,2) dengan 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 2 adalah jumlah iterasi. Untuk 4 iterasi, dilakukan pemanggilan *function* IterasiJacobi(A,b,10^-4,4) dengan 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 4 adalah jumlah iterasi. Untuk 8 iterasi, dilakukan pemanggilan *function* IterasiJacobi(A,b,10^-4,8) iterasi 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 8 adalah jumlah iterasi. Kemudian, hasil operasi dari masing-masing pemanggilan *function* akan ditampilkan ke layar disertai dengan total waktu komputasinya. Pada pemanggilan *function* dengan 2, 4, dan 8 iterasi dengan menggunakan metode Jacobi **didapatkan hasil yang konvergen sejak iterasi ke-2.**

- Metode Gauss-Seidel



```
Command Window
>> A = [10 -1 2 0;-1 11 -1 3;2 -1 10 0;0 3 -1 8]
A =

   10   -1    2    0
   -1   11   -1    3
    2   -1   10    0
    0    3   -1    8

>> b = [6;25;-11;-11]
b =

    6
   25
  -11
  -11
```



```
Command Window

>> IterasiGaussSeidel(A,b, 10^-4, 2)
Gauss Seidel method not konverge
Elapsed time is 0.000597954 seconds.
ans =

   1.0302
   2.9233
  -1.0137
  -2.5980

>> IterasiGaussSeidel(A,b, 10^-4, 4)
Gauss Seidel method not konverge
Elapsed time is 0.00191092 seconds.
ans =

   1.1029
   2.9957
  -1.0210
  -2.6260

>> IterasiGaussSeidel(A,b, 10^-4, 8)
Gauss Seidel method konverge
Elapsed time is 0.00188899 seconds.
ans =

   1.1039
   2.9965
  -1.0211
  -2.6263
```

Pada command window di atas, dilakukan pendeklarasi dan pengisialisasian dari *matrix* A dan *matrix* B. Untuk 2 iterasi, dilakukan pemanggilan *function* IterasiGaussSeidel(A,b,10^-4,2) dengan 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 2 adalah jumlah iterasi. Untuk 4 iterasi, dilakukan pemanggilan *function* IterasiGaussSeidel(A,b,10^-4,4) dengan 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 4 adalah jumlah iterasi. Untuk 8 iterasi, dilakukan pemanggilan *function* IterasiGaussSeidel(A,b,10^-4,8) iterasi 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 8 adalah jumlah iterasi. Kemudian, hasil operasi dari masing-masing pemanggilan *function* akan ditampilkan ke layar disertai dengan total waktu komputasinya. Pada pemanggilan *function* dengan 2, 4, dan 8 iterasi dengan menggunakan metode Jacobi **baru didapatkan hasil yang konvergen pada iterasi ke-8.**

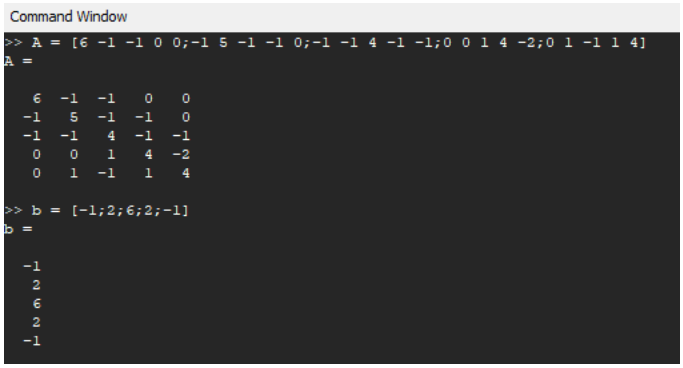
b) Soal B

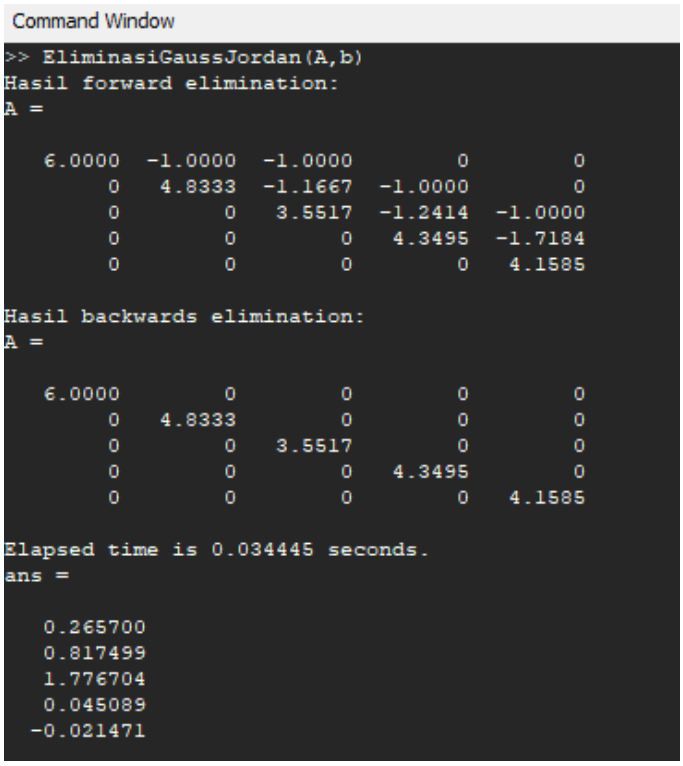
- Metode Gauss

```
Command Window
>> A = [6 -1 -1 0 0;-1 5 -1 -1 0;-1 -1 4 -1 -1;0 0 1 4 -2;0 1 -1 1 4]
A =

   6  -1  -1   0   0
  -1   5  -1  -1   0
  -1  -1   4  -1  -1
   0   0   1   4  -2
   0   1  -1   1   4

>> b = [-1;2;6;2;-1]
b =

  -1
   2
   6
   2
  -1

>> EliminasiGauss(A,b)
Hasil forward elimination:
A =

   6.0000  -1.0000  -1.0000        0        0
        0   4.8333  -1.1667  -1.0000        0
        0        0   3.5517  -1.2414  -1.0000
        0        0        0   4.3495  -1.7184
        0        0        0        0   4.1585

Elapsed time is 0.015702 seconds.
ans =

   0.265700
   0.817499
   1.776704
   0.045089
  -0.021471

>>
```

Pada command window di atas, dilakukan pendeklarasi dan pengisialisasian dari *matrix* A dan *matrix* B. Kemudian, dilakukan pemanggilan *function* **EliminasiGauss(A,b)** dengan dua parameter dimana **A** adalah **matriks A** dan **b** adalah **matriks B** yang digunakan untuk menjalankan *source code* perhitungan sistem persamaan linear dengan metode Gauss. Lalu, akan ditampilkan hasil dari forward elimination dalam *matrix* A. Kemudian, hasil dari **a, b, c**, dan **d** akan ditampilkan dalam command window yang disertai dengan waktu komputasinya.

- Metode Gauss Jordan





Pada command window di atas, dilakukan pendeklarasian dan penginisialisasian dari **matriks A** dan **matriks B**. Kemudian, dilakukan pemanggilan *function* **EliminasiGaussJordan(A,b)** dengan dua parameter dimana A adalah **matriks A** dan b adalah **matriks B**. Lalu, dilakukan operasi **forward elimination** dan hasilnya akan ditampilkan ke layar dalam matriks A. Setelah itu, akan dilakukan operasi backwards elimination dan hasilnya akan ditampilkan ke layar dalam matriks A.

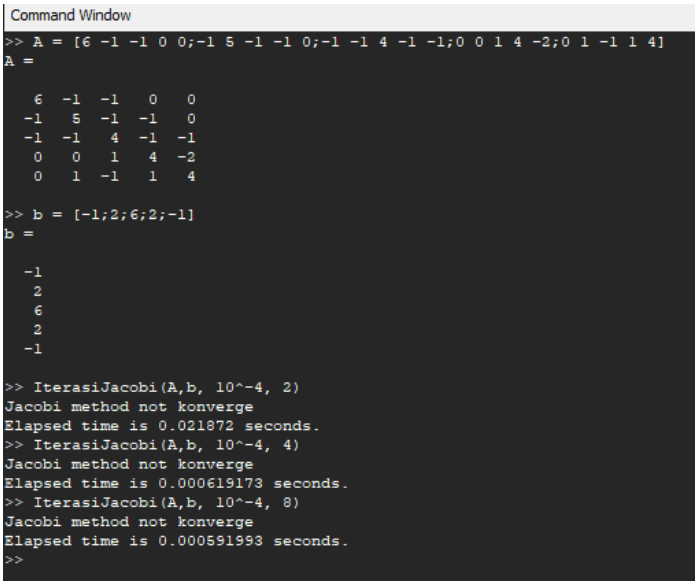
- Metode LU Dekomposisi

```
Command Window
>> A = [6 -1 -1 0 0;-1 5 -1 -1 0;-1 -1 4 -1 -1;0 0 1 4 -2;0 1 -1 1 4]
A =

   6  -1  -1   0   0
  -1   5  -1  -1   0
  -1  -1   4  -1  -1
   0   0   1   4  -2
   0   1  -1   1   4

>> b = [-1;2;6;2;-1]
b =

  -1
   2
   6
   2
  -1
```

```
Command Window
>> LU_Solusi(A,b)
Hasil dekomposisi LU
L =

   1.0000         0         0         0         0
  -0.1667    1.0000         0         0         0
  -0.1667   -0.2414    1.0000         0         0
        0         0    0.2816    1.0000         0
        0    0.2069   -0.2136    0.2165    1.0000

U =

   6.0000   -1.0000   -1.0000         0         0
        0    4.8333   -1.1667   -1.0000         0
        0         0    3.5517   -1.2414   -1.0000
        0         0         0    4.3495   -1.7184
        0         0         0         0    4.1585

b =

  -1
   2
   6
   2
  -1

Elapsed time is 0.0504501 seconds.
ans =

   0.265700
   0.817499
   1.776704
   0.045089
  -0.021471
```

Pada command window di atas, dilakukan pendeklarasian dan pengisialisasian dari ***matrix* A** dan ***matrix* B.** Kemudian, dilakukan pemanggilan *function* **LU_Solusi(A,b)** dengan dua parameter dimana **A** adalah **matriks A** dan **b** adalah **matriks B**. Matriks A akan dipisah menjadi dua matriks yang berbeda, yaitu matriks L (Lower) dan U (Upper). Kemudian, dilakukan operasi untuk memisahkan matriks A menjadi matriks Lower dan Upper kemudian hasilnya akan ditampilkan ke layar. Setelah itu, dilakukan operasi **[L][Z] = [C]** untuk menghitung nilai **[Z]** yang ditampilkan sebagai nilai dari **matriks B**. Lalu, dilakukan operasi **[U][X] = [Z]** untuk mendapatkan nilai dari **[X]** yang kemudian value dari **[X]** sebagai **matriks B** yang berupa nilai dari **a, b, c,** dan **d** akan ditampilkan pada command window yang disertai dengan waktu komputasinya.

- Metode Jacobi



Pada command window di atas, dilakukan pendeklarasian dan pengisialisasian dari ***matrix* A** dan ***matrix* B**. Untuk 2 iterasi, dilakukan pemanggilan *function* **IterasiJacobi(A,b,10^-4,2)** dengan 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 2 adalah jumlah iterasi. Untuk 4 iterasi, dilakukan pemanggilan *function* **IterasiJacobi(A,b,10^-4,4)** dengan 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 4 adalah jumlah iterasi. Untuk 8 iterasi, dilakukan pemanggilan *function* **IterasiJacobi(A,b,10^-4,8)** iterasi 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 8 adalah jumlah iterasi. Kemudian, hasil operasi dari masing-masing pemanggilan *function* akan ditampilkan ke layar disertai dengan total waktu komputasinya. Pada pemanggilan *function* dengan 2, 4, dan 8 iterasi dengan menggunakan metode Jacobi **tidak didapatkan hasil yang konvergen.**

```
>> IterasiJacobi(A,b, 10^-4,9)
Jacobi method not konverge
Elapsed time is 0.00112104 seconds.
>> IterasiJacobi(A,b, 10^-4,10)
Jacobi method konverge
Elapsed time is 0.000937939 seconds.
ans =

   0.265675
   0.817469
   1.776665
   0.045093
  -0.021480
```

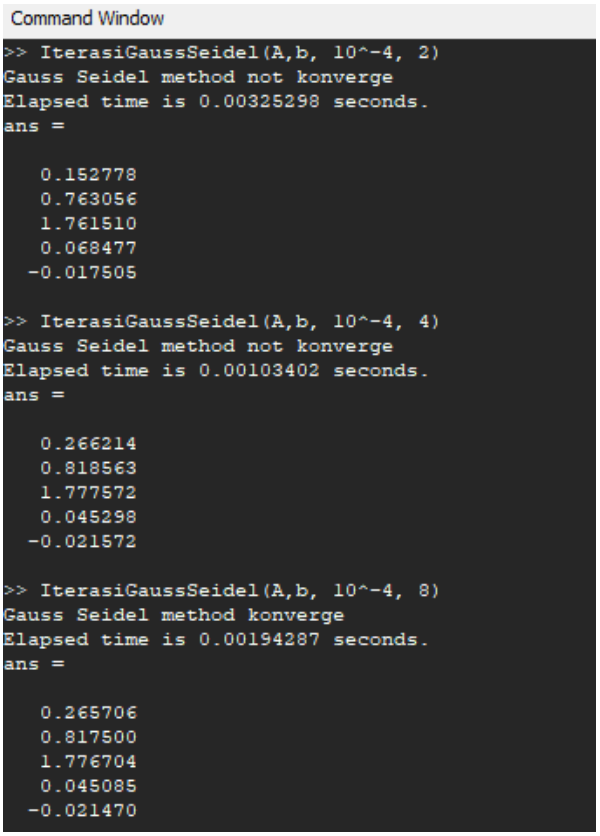
Ketika dilakukan pemanggilan *function* untuk **iterasi ke-9**, hasil yang didapatkan **tidak konvergen**. Akan tetapi, ketika dilakukan pemanggilan *function* untuk **iterasi ke-10 yang didapatkan konvergen.**

- Metode Gauss Seidel

Command Window

```
>> A = [6 -1 -1 0 0;-1 5 -1 -1 0;-1 -1 4 -1 -1;0 0 1 4 -2;0 1 -1 1 4]
A =

   6  -1  -1   0   0
  -1   5  -1  -1   0
  -1  -1   4  -1  -1
   0   0   1   4  -2
   0   1  -1   1   4

>> b = [-1;2;6;2;-1]
b =

  -1
   2
   6
   2
  -1
```

Pada command window di atas, dilakukan pendeklarasi dan pengisialisasian dari ***matrix* A** dan ***matrix* B.**



Untuk 2 iterasi, dilakukan pemanggilan *function* **IterasiGaussSeidel(A,b,10^-4,2)** dengan 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 2 adalah jumlah iterasi. Untuk 4 iterasi, dilakukan pemanggilan *function* **IterasiGaussSeidel(A,b,10^-4,4)** dengan 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 4 adalah jumlah iterasi. Untuk 8 iterasi, dilakukan pemanggilan *function* **IterasiGaussSeidel(A,b,10^-4,8)** iterasi 4 parameter dimana A adalah matriks A, b adalah matriks b, 10^-4 adalah toleransi dari error, dan 8 adalah jumlah iterasi. Kemudian, hasil operasi dari masing-masing pemanggilan *function* akan ditampilkan ke layar disertai dengan total waktu komputasinya. Pada pemanggilan *function* dengan 2, 4, dan 8 iterasi dengan menggunakan metode Gauss-Seidel **didapatkan hasil yang konvergen pada iterasi ke-8.**

2. Perbandingan Galat/Error

a) SOAL A

Dikarenakan untuk ketiga **metode langsun**g didapatkan nilai [a;b;c;d] yang sama sehingga hanya salah satu dari metode tersebut yang akan digunakan.

- Metode Gauss

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1.1039 \\ 2.9965 \\ -1.0211 \\ -2.6263 \end{bmatrix}$$

- Metode Gauss-Jordan

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1.1039 \\ 2.9965 \\ -1.0211 \\ -2.6263 \end{bmatrix}$$

- Metode LU Dekomposisi

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1.1039 \\ 2.9965 \\ -1.0211 \\ -2.6263 \end{bmatrix}$$

- Metode Jacobi

Pada metode jacobi nilai dari [a;b;c;d] sudah konvergen sejak iterasi ke-2 sebagai berikut.

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1.0473 \\ 2.6023 \\ -0.9927 \\ -2.3648 \end{bmatrix}$$

Galat :

$$\left\| \begin{bmatrix} 1.1039 \\ 2.9965 \\ -1.0211 \\ -2.6263 \end{bmatrix} - \begin{bmatrix} 1.0473 \\ 2.6023 \\ -0.9927 \\ -2.3648 \end{bmatrix} \right\| = \begin{bmatrix} 0.0566 \\ 0.3942 \\ 0.0248 \\ 0.2615 \end{bmatrix}$$

- Metode Gauss-Seidel

Pada metode Gauss-Seidel mengalami konvergensi ketika iterasi ke-8 sehingga nilai [a;b;c;d] yang digunakan adalah nilai ketika iterasi ke-8.

a = 1.1039 ; b = 2.9965 ; c = -1.0211 ; d = -2.6263

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1.1039 \\ 2.9965 \\ -1.0211 \\ -2.6263 \end{bmatrix}$$

Galat :

$$\left\| \begin{bmatrix} 1.1039 \\ 2.9965 \\ -1.0211 \\ -2.6263 \end{bmatrix} - \begin{bmatrix} 1.1039 \\ 2.9965 \\ -1.0211 \\ -2.6263 \end{bmatrix} \right\| = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Berdasarkan perhitungan galat di atas, dapat disimpulkan bahwa metode Gauss-Seidel memiliki nilai galat yang lebih kecil daripada metode Jacobi.

b) SOAL B

Dikarenakan untuk ketiga **metode langsun**g didapatkan nilai [p;q;r;s;t] yang sama sehingga hanya salah satu dari metode tersebut yang akan digunakan.

- Metode Gauss

$$\begin{bmatrix} p \\ q \\ r \\ s \\ t \end{bmatrix} = \begin{bmatrix} 0.265700 \\ 0.817499 \\ 1.776704 \\ 0.045089 \\ -0.021471 \end{bmatrix}$$

- Metode Gauss-Jordan

$$\begin{bmatrix} p \\ q \\ r \\ s \\ t \end{bmatrix} = \begin{bmatrix} 0.265700 \\ 0.817499 \\ 1.776704 \\ 0.045089 \\ -0.021471 \end{bmatrix}$$

- Metode LU Dekomposisi

$$\begin{bmatrix} p \\ q \\ r \\ s \\ t \end{bmatrix} = \begin{bmatrix} 0.265700 \\ 0.817499 \\ 1.776704 \\ 0.045089 \\ -0.021471 \end{bmatrix}$$

- Metode Jacobi

  Pada metode Jacobi hasil yang konvergen tidak didapatkan pada iterasi ke-8 tetapi baru didapatkan pada **iterasi ke-10.**

$$\begin{bmatrix} p \\ q \\ r \\ s \\ t \end{bmatrix} = \begin{bmatrix} 0.265675 \\ 0.817469 \\ 1.776665 \\ 0.045093 \\ -0.021480 \end{bmatrix}$$

Galat :

$$\left\| \begin{bmatrix} 0.265700 \\ 0.817499 \\ 1.776704 \\ 0.045089 \\ -0.021471 \end{bmatrix} - \begin{bmatrix} 0.265675 \\ 0.817469 \\ 1.776665 \\ 0.045093 \\ -0.021480 \end{bmatrix} \right\| = \begin{bmatrix} 0.000025 \\ 0.00003 \\ 0.0001054 \\ 0.000004 \\ 0.000009 \end{bmatrix}$$

- Metode Gauss-Seidel

  Pada metode Gauss-Seidel mengalami konvergensi ketika iterasi ke-8 sehingga nilai [p;q;r;s;t] yang digunakan adalah nilai ketika iterasi ke-8.

$$\begin{bmatrix} p \\ q \\ r \\ s \\ t \end{bmatrix} = \begin{bmatrix} 0.265706 \\ 0.817500 \\ 1.776704 \\ 0.045085 \\ -0.021470 \end{bmatrix}$$

  Galat :

$$\left\| \begin{bmatrix} 0.265700 \\ 0.817499 \\ 1.776704 \\ 0.045089 \\ -0.021471 \end{bmatrix} - \begin{bmatrix} 0.265706 \\ 0.817500 \\ 1.776704 \\ 0.045085 \\ -0.021470 \end{bmatrix} \right\| = \begin{bmatrix} 0.000006 \\ 0.000001 \\ 0 \\ 0.000004 \\ 0.000001 \end{bmatrix}$$

  Berdasarkan perhitungan galat di atas, dapat disimpulkan bahwa metode Gauss-Seidel memiliki nilai galat yang lebih kecil daripada metode Jacobi.

3. Perbandingan Bentuk Komputasi

a) SOAL A

- Metode Langsung (eksak)

Waktu komputasi	
Metode Gauss	0,00165319
Metode Gauss-Jordan	0,00274706
Dekomposisi LU	0,0250111

- Metode Iteratif

N	Waktu Komputasi	
	Metode Jacobi	Metode Gauss-Seidel
2	0,000404119	0,000597954
4	0,00051403	0,00191092
8	0,00039196	0,00188899

Pada soal A, metode Jacobi adalah metode dengan membutuhkan komputasi waktu yang paling sedikit di antara keempat metode lainnya.

b) SOAL B

- Metode Langsung (eksak)

Waktu komputasi	
Metode Gauss	0,015702
Metode Gauss-Jordan	0,034445
Dekomposisi LU	0,0504501

- Metode Iteratif

N	Waktu Komputasi	
	Metode Jacobi	Metode Gauss-Seidel
2	0,021872	0,00325298
4	0,000619173	0,00103402
8	0,000591993	0,00194287

Pada soal B, metode Jacobi adalah metode dengan membutuhkan komputasi waktu yang paling sedikit di antara keempat metode lainnya.