## (I) Copy of Code With The Value N1

```
% GAUSSIAN ELIMINATION WITH BACKWARD SUBSTITUTION ALGOTITHM 6.1
%
% To solve the n by n linear system
%
% E1:  A(1,1) X(1) + A(1,2) X(2) +...+ A(1,n) X(n) = A(1,n+1)
% E2:  A(2,1) X(1) + A(2,2) X(2) +...+ A(2,n) X(n) = A(2,n+1)
% :
% .
% EN:  A(n,1) X(1) + A(n,2) X(2) +...+ A(n,n) X(n) = A(n,n+1)
%
% INPUT:   number of unknowns and equations n; augmented
%          matrix A = (A(I,J)) where 1<=I<=n and 1<=J<=n+1.
%
% OUTPUT:  solution x(1), x(2),...,x(n) or a message that the
%          linear system has no unique solution.
%   syms('AA', 'NAME', 'INP', 'OK', 'N', 'I', 'J', 'A', 'NN', 'M');
%   syms('ICHG', 'IP', 'JJ', 'C', 'XM', 'K', 'X', 'SUM');
%   syms('KK', 'FLAG', 'OUP');
TRUE = 1;
FALSE = 0;
fprintf(1,'This is Gaussian Elimination to solve a linear system.\n');
fprintf(1,'The array and right-hand-side shall be written in the code in
advance.\n');
fprintf(1,'Have the array and right-hand-side been given? - enter Y or N.\n');
AA = input(' ','s');
OK = FALSE;
if AA == 'Y' | AA == 'y'
while OK == FALSE
% Matrix dimension N
N=2130;
A = zeros(N,N+1);
X = zeros(1,N);
for I = 1:N
for J = 1:N
  if I==J,
   A(I,J) =5; %A(1,1), A(2,2), ..., A(N,N) entries
  end;
  if J==I+1,
   A(I,J)=-1; %A(1,2), A(2,3), ..., A(N-1,N) entries
  end;
  if J==I-1,
   A(I,J)=-1; %A(2,1), A(3,2), ..., A(N,N-1) entries
  end;
end;
end;
% right-hand-side B(I)
  A(1,N+1)=4;  %B(1)
  for I = 2:N-1
   A(I,N+1)=3; %B(I)
```

```
  end;
  A(N,N+1)=4;  %B(N)

OK = TRUE;
end;
else
fprintf(1,'The program will end so the matrix can be written.\n');
end;
if OK == TRUE
fprintf(1,'The code is executing now.\n');
t = cputime; %Starting time
% STEP 1
% Elimination Process
NN = N-1;
M = N+1;
ICHG = 0;
I = 1;
while OK == TRUE & I <= NN
% STEP 2
% use IP instead of p
IP = I;
while abs(A(IP,I)) <= 1.0e-20 & IP <= N
IP = IP+1;
end;
if IP == M
OK = FALSE;
else
% STEP 3
if IP ~= I
for JJ = 1:M
C = A(I,JJ);
A(I,JJ) = A(IP,JJ);
A(IP,JJ) = C;
end;
ICHG = ICHG+1;
end;
% STEP 4
JJ = I+1;
for J = JJ:N
% STEP 5
% use XM in place of m(J,I)
XM = A(J,I)/A(I,I);
% STEP 6
for K = JJ:M
A(J,K) = A(J,K) - XM * A(I,K);
end;
% Multiplier XM could be saved in A(J,I).
A(J,I) = 0;
end;
```

```
end;
I = I+1;
end;
if OK == TRUE
% STEP 7
if abs(A(N,N)) <= 1.0e-20
OK = FALSE;
else
% STEP 8
% start backward substitution
X(N) = A(N,M) / A(N,N);
% STEP 9
for K = 1:NN
I = NN-K+1;
JJ = I+1;
SUM = 0;
for KK = JJ:N
SUM = SUM - A(I,KK) * X(KK);
end;
X(I) = (A(I,M)+SUM) / A(I,I);
end;
eltime = cputime-t; %total elapse time
OUP = 1;
%Find the maximal error of computationed solution
ERRMAX=0.0;
for I=1:N
   TMP=abs(X(I)-1);    %exact solution value Xi
   if TMP > ERRMAX
      ERRMAX=TMP;
   end;
end;
fprintf(OUP, '\nGAUSSIAN ELIMINATION\n');
fprintf (OUP, 'With %d row interchange(s)\n', ICHG);
fprintf(OUP, '\nThe matrix dimension =  %d X %d \n', N,N);
fprintf(OUP, 'MAX ERROR=  %12.8e\n', ERRMAX);
if ERRMAX<0.00001,
fprintf(OUP, '\nCpu time is %12.8e\n',eltime);
else
 fprintf(OUP, 'Something is wrong. \n');
end;
end;
end;
end;
```

**A) M = 71 | N1 = 30 * 71 = 2130 | Here is the code ran 4 times**

Command Window

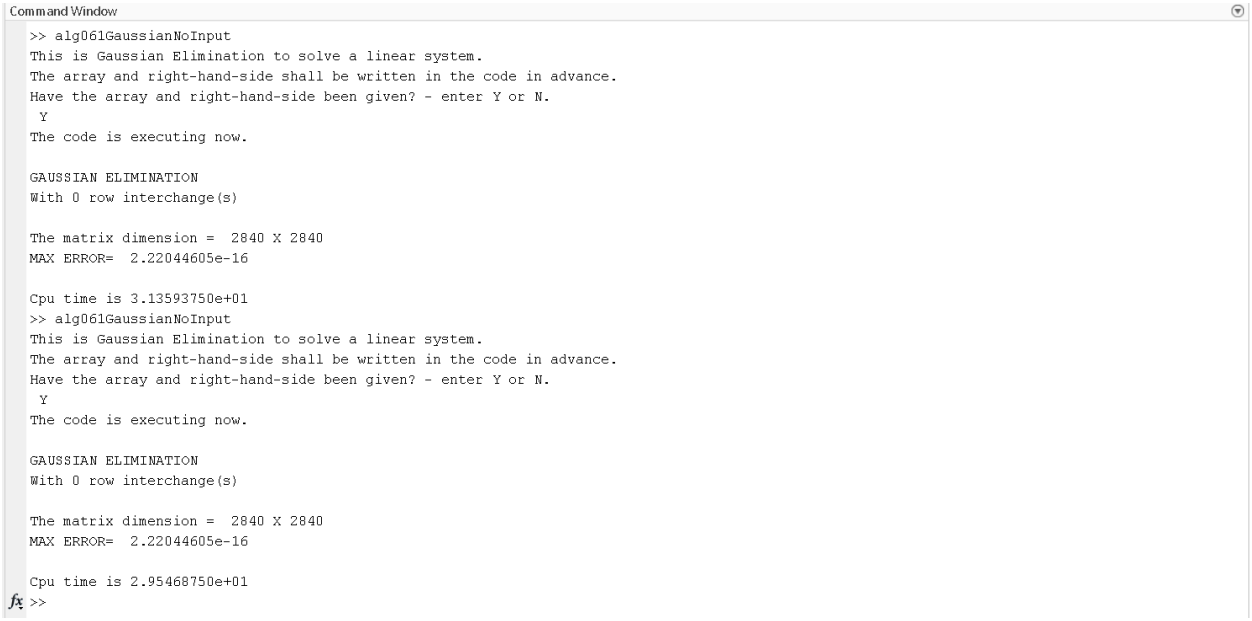
```
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  2130 X 2130
MAX ERROR=  2.22044605e-16

Cpu time is 1.49531250e+01
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  2130 X 2130
MAX ERROR=  2.22044605e-16

Cpu time is 1.40625000e+01
fx >>
```

Command Window

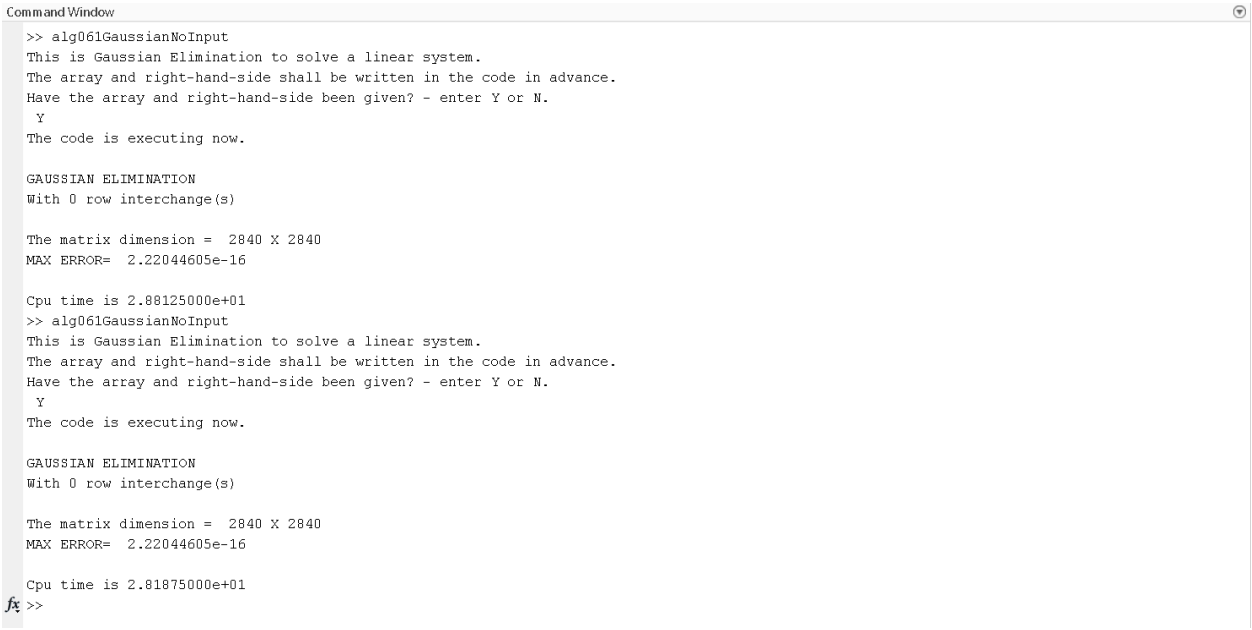
```
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  2130 X 2130
MAX ERROR=  2.22044605e-16

Cpu time is 1.40000000e+01
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  2130 X 2130
MAX ERROR=  2.22044605e-16

Cpu time is 1.38750000e+01
fx >>
```

**E1 = 56.890625 / 4**
**E1 = 14.22265625**

**M = 71 | N2 = 40 * 71 = 2840 | Here is the code ran 4 times**



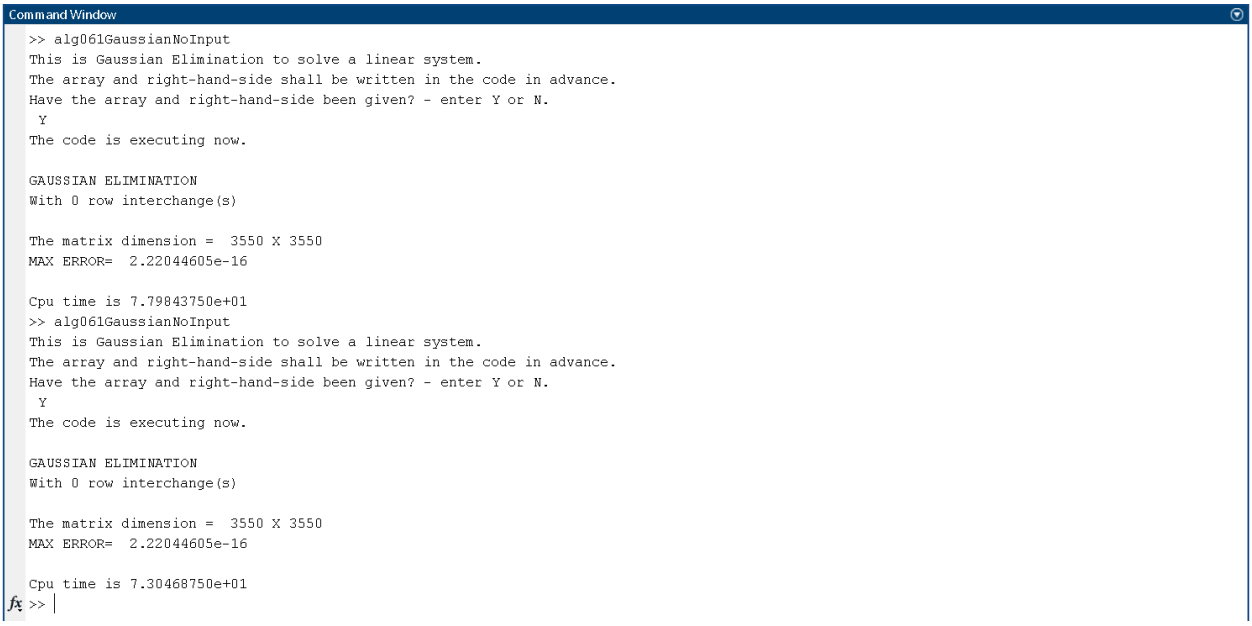
```
Command Window
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  2840 X 2840
MAX ERROR=  2.22044605e-16

Cpu time is 3.13593750e+01
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  2840 X 2840
MAX ERROR=  2.22044605e-16

Cpu time is 2.95468750e+01
fx >>
```

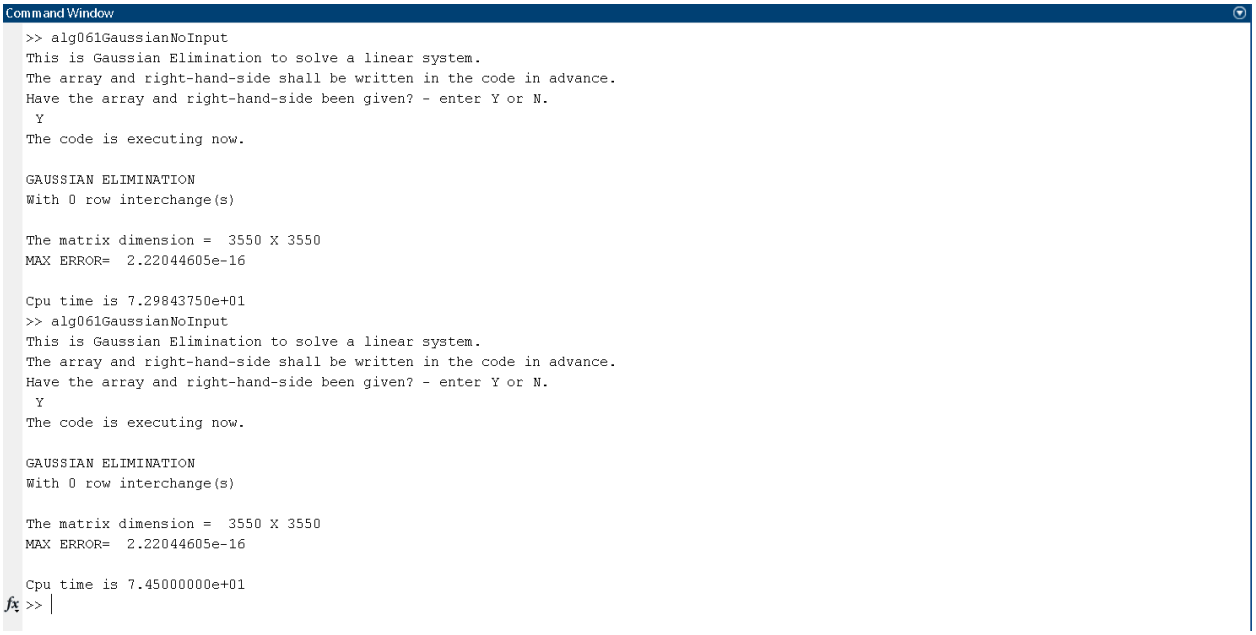


```
Command Window
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  2840 X 2840
MAX ERROR=  2.22044605e-16

Cpu time is 2.88125000e+01
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  2840 X 2840
MAX ERROR=  2.22044605e-16

Cpu time is 2.81875000e+01
fx >>
```

**E2 = 117.90625 / 4**
**E2 = 29.4765625**

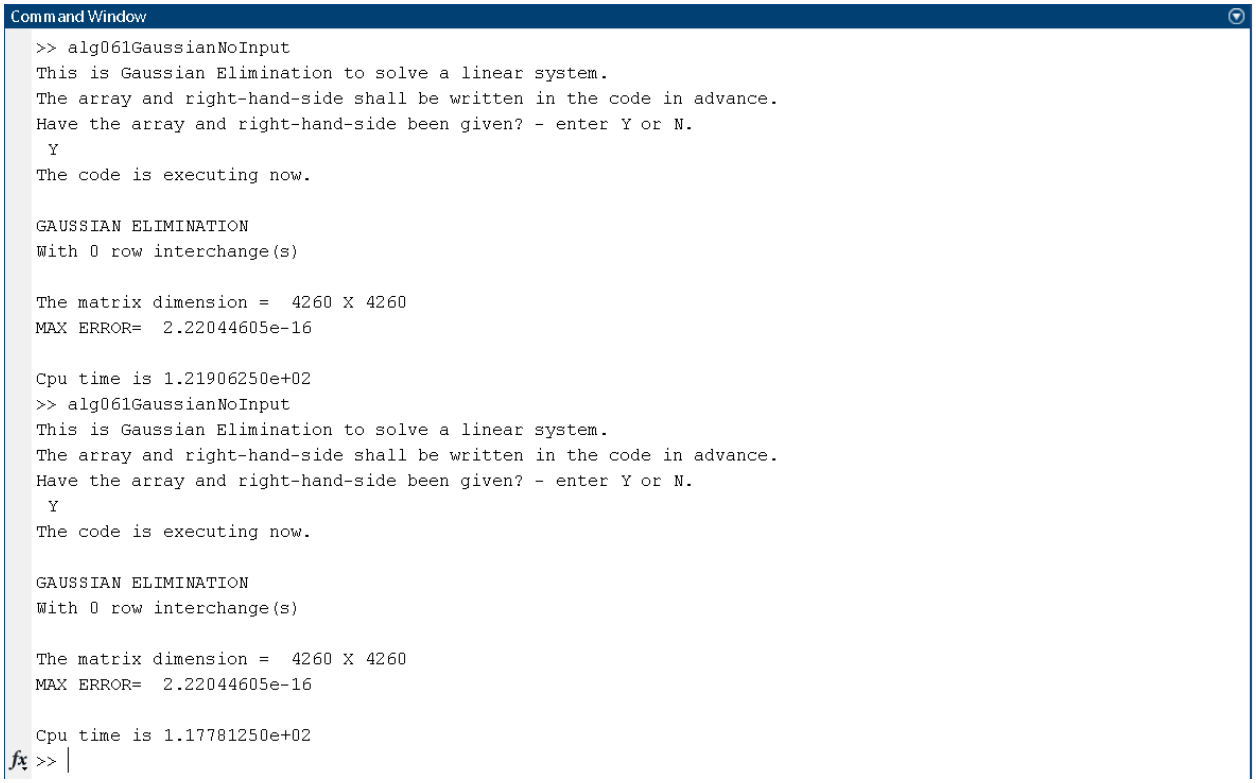
**M = 71 | N3 = 50 * 71 = 3550 | Here is the code ran 4 times**



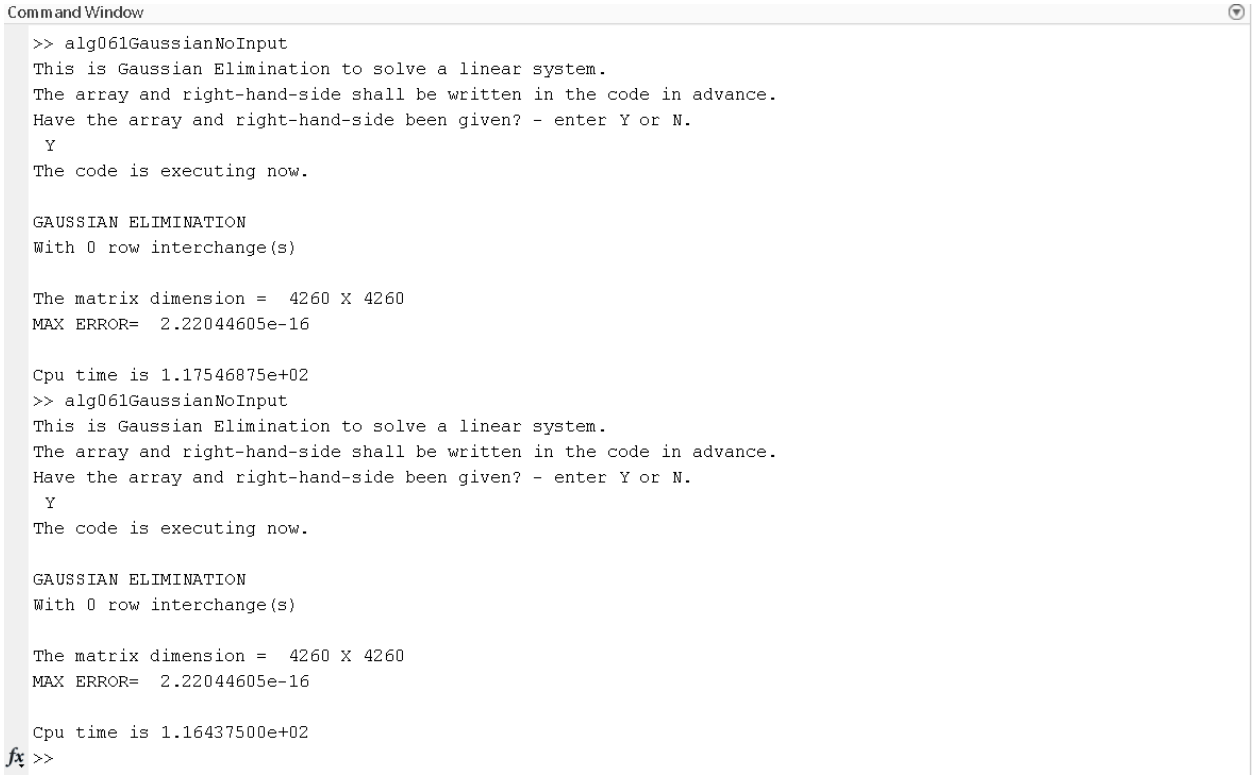




**E3 = 298.515625 / 4**
**E3 = 74.62890625**

**M = 71 | N4 = 60 * 71 = 4260 | Here is the code ran 4 times**



```
Command Window
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  4260 X 4260
MAX ERROR=  2.22044605e-16

Cpu time is 1.21906250e+02
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  4260 X 4260
MAX ERROR=  2.22044605e-16

Cpu time is 1.17781250e+02
fx >>
```

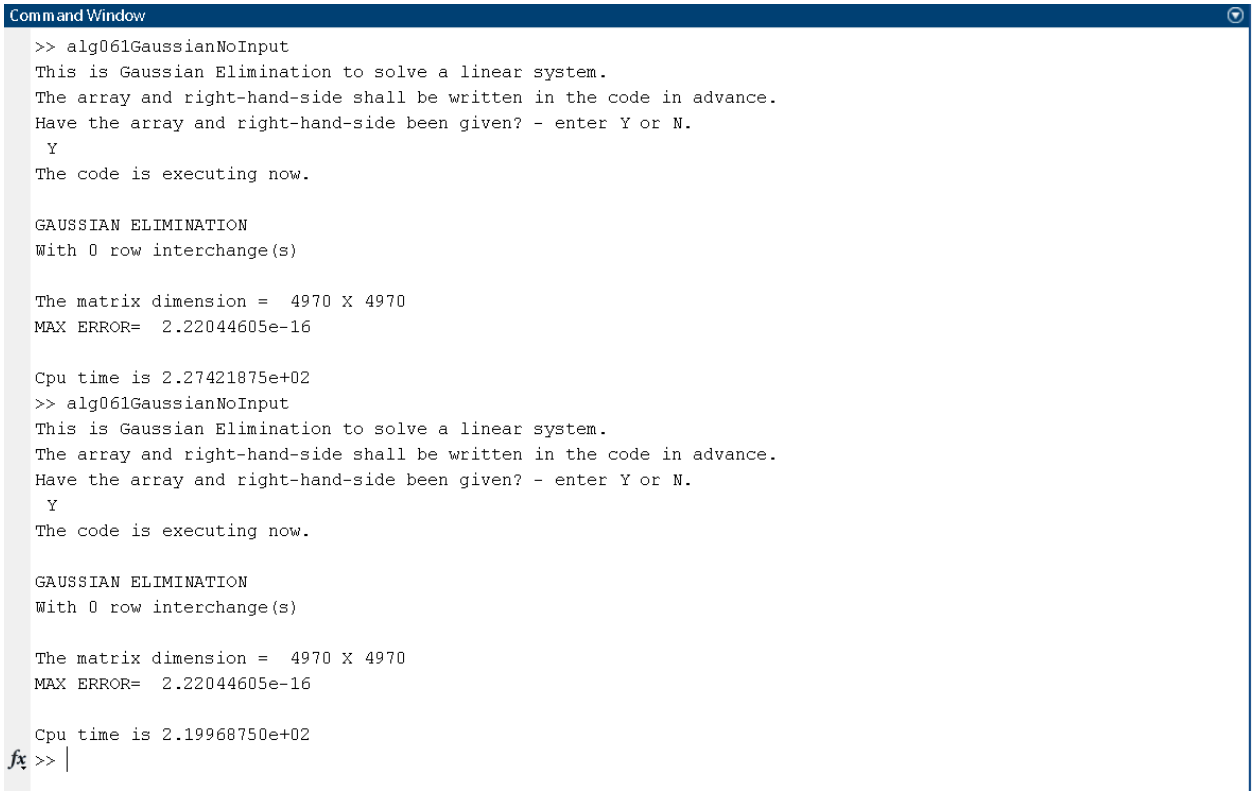


```
Command Window
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  4260 X 4260
MAX ERROR=  2.22044605e-16

Cpu time is 1.17546875e+02
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  4260 X 4260
MAX ERROR=  2.22044605e-16

Cpu time is 1.16437500e+02
fx >>
```

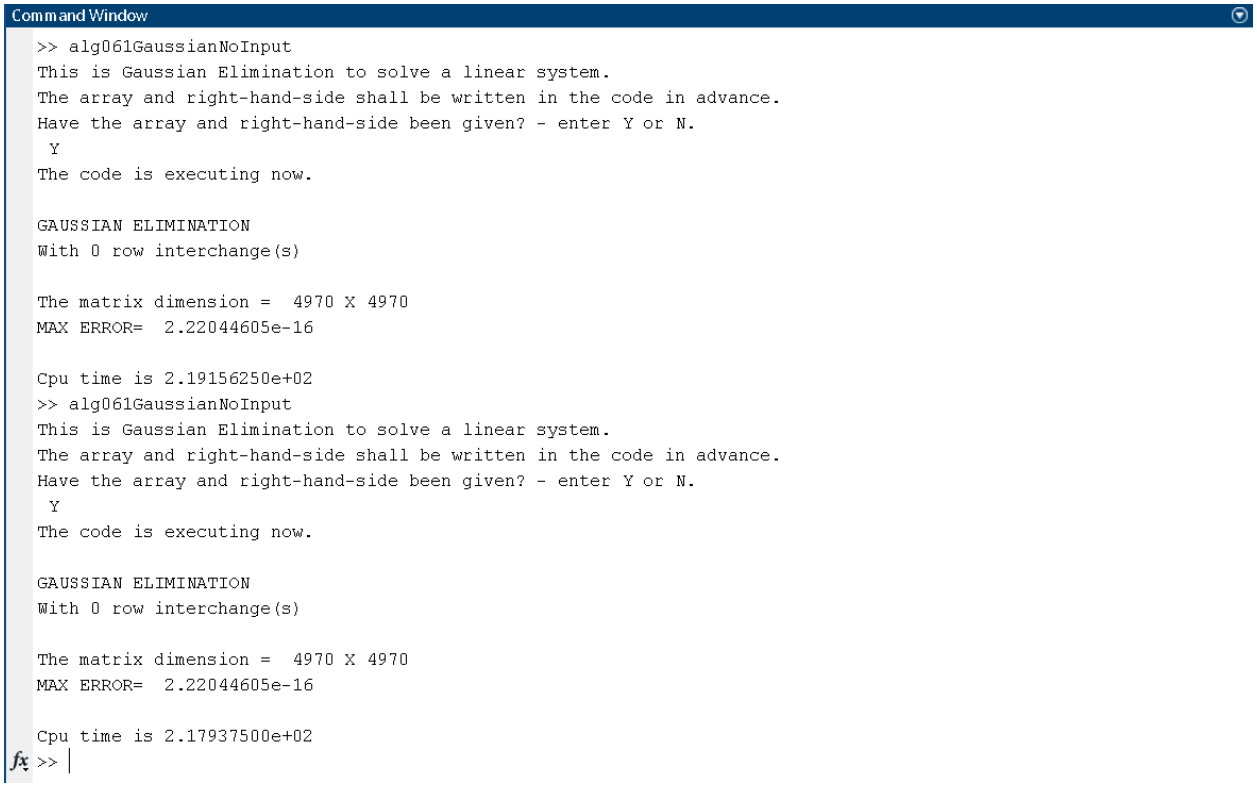
**E4 = 473.671875 / 4**

**E4 = 118.4179688**

**M = 71 | N5 = 70 * 71 = 4970 | Here is the code ran 4 times**



```
Command Window
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  4970 X 4970
MAX ERROR=  2.22044605e-16

Cpu time is 2.27421875e+02
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  4970 X 4970
MAX ERROR=  2.22044605e-16

Cpu time is 2.19968750e+02
fx >>
```



```
Command Window
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  4970 X 4970
MAX ERROR=  2.22044605e-16

Cpu time is 2.19156250e+02
>> alg061GaussianNoInput
This is Gaussian Elimination to solve a linear system.
The array and right-hand-side shall be written in the code in advance.
Have the array and right-hand-side been given? - enter Y or N.
 Y
The code is executing now.

GAUSSIAN ELIMINATION
With 0 row interchange(s)

The matrix dimension =  4970 X 4970
MAX ERROR=  2.22044605e-16

Cpu time is 2.17937500e+02
fx >>
```

**E5 = 884.484375 / 4**
**E5 = 221.1210938**

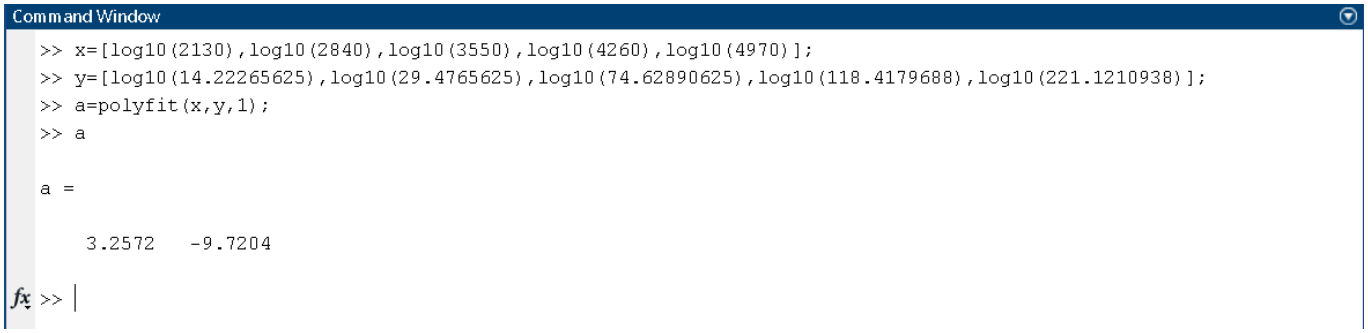
### i) Create Tables to Report Average Time Ei

| xxxxxxxxxxx | N1 = 2130 | N2 = 2840 | N3 = 3550 | N4 = 4260 | N5 = 4970 |
|---|---|---|---|---|---|
| Time 1 | 14.9531250 | 31.359375 | 77.9843750 | 121.90625 | 227.421875 |
| Time 2 | 14.0625 | 29.546875 | 73.0468750 | 117.78125 | 219.968750 |
| Time 3 | 14.000 | 28.8125 | 72.9843750 | 117.546875 | 219.156250 |
| Time 4 | 13.875 | 28.1875 | 74.5000 | 116.437500 | 217.937500 |
| Average Time (Ei) | 14.22265625 | 29.4765625 | 74.62890625 | 118.4179688 | 221.1210938 |

**E1 = 14.22265625**
**E2 = 29.4765625**
**E3 = 74.62890625**
**E4 = 118.4179688**
**E5 = 221.1210938**

---

### ii) Line Fitting On Data Set {log10(Ni),log10(Ei))}

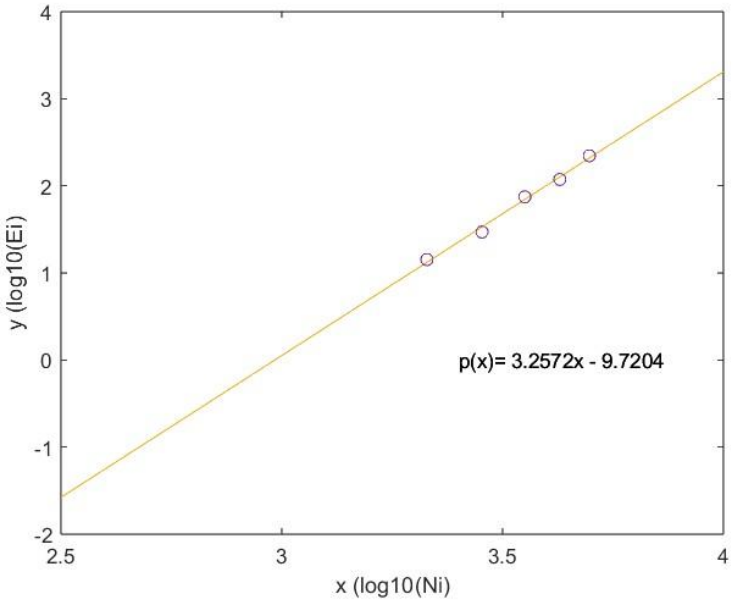



**Polynomial of Degree One Obtained:**

**p(x) = 3.2572x - 9.7204**

---

### iii) Plot: a) the line obtained by line fitting & b) the data pairs




### Code That Produced the Graph:

```
>> t=linspace(2.5,4,101);
>> s=polyval(a,t);
>> plot(t,s);
>> hold on;
>> plot(x,y,'o');
>> xlabel('x')
>> ylabel('y')
>> text(3.4,0,'p(x)= 3.2572x - 9.7204')
>> t=linspace(2.5,4,101);
>> s=polyval(a,t);
>> plot(t,s);
>> hold on;
>> plot(x,y,'o');
>> xlabel('x (log10(Ni)')
>> ylabel('y (log10(Ei)')
>> text(3.4,0,'p(x)= 3.2572x - 9.7204')
>>
```

---

### iv) What is the Slope of the Line You Obtained? Is it Close to 3?

**Slope = 3.2572**

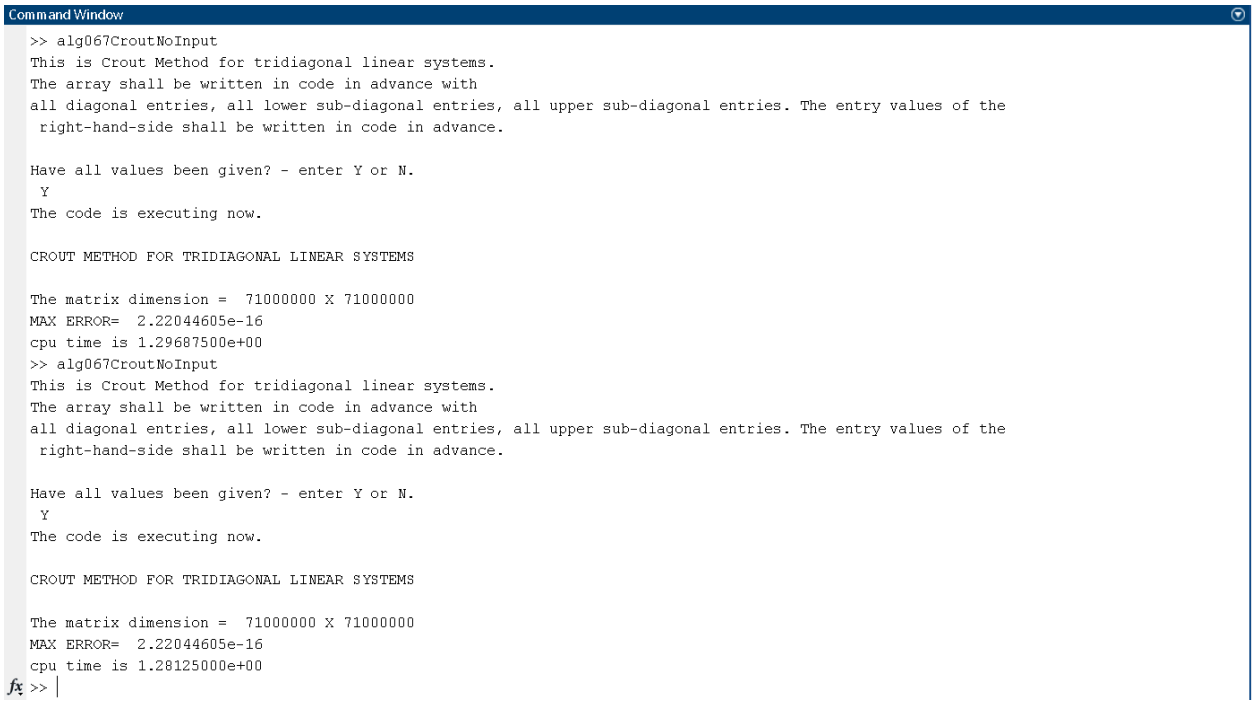
**The slope found is close to 3.**
**It is only slightly larger.**

## b) Copy of Code With The Value N1

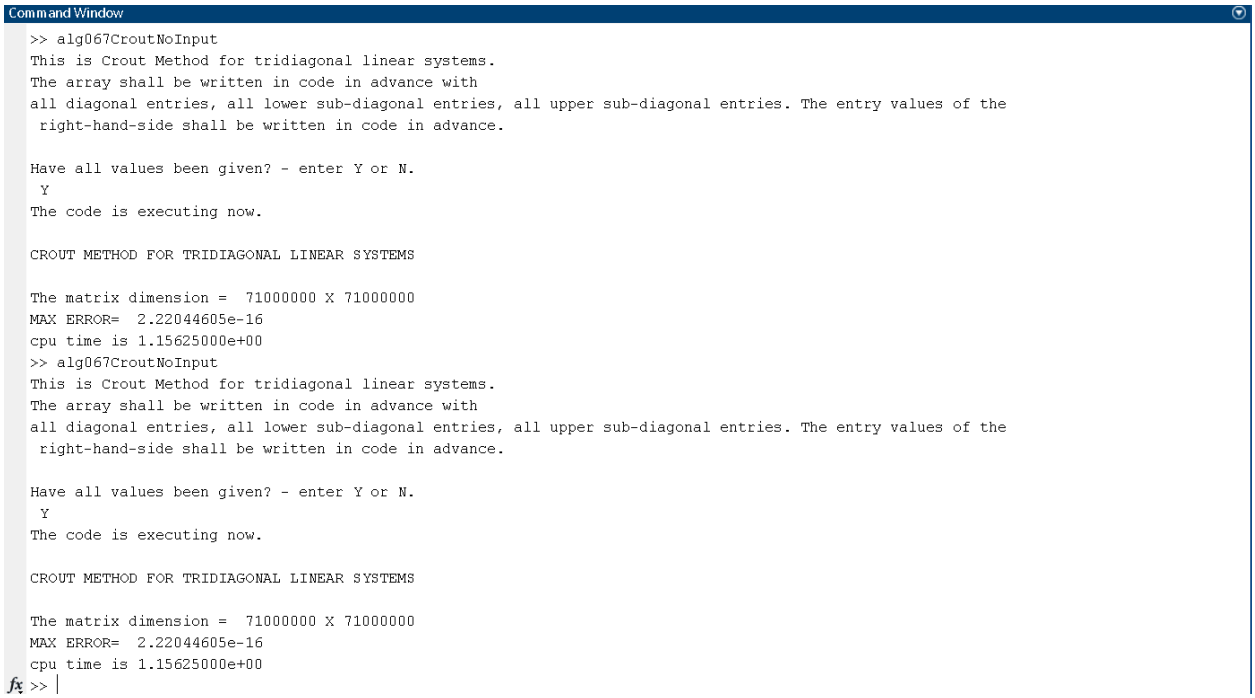
```
% CROUT FACTORIZATION FOR TRIDIAGONAL LINEAR SYSTEMS ALGORITHM 6.7
%
% To solve the n x n linear system
%
% E1:  A(1,1) X(1) + A(1,2) X(2)                   = A(1,n+1)
% E2:  A(2,1) X(1) + A(2,2) X(2) + A(2,3) X(3)    = A(2,n+1)
% :
% .
% E(n):          A(n,n-1) X(n-1) + A(n,n) X(n)    = A(n,n+1)
%
% INPUT:   the dimension n; the entries of A.
%
% OUTPUT:  the solution X(1), ..., X(N).
%   syms('AA', 'OK', 'NAME', 'INP', 'N', 'I', 'A', 'B', 'NN');
%   syms('C', 'BB', 'Z', 'X', 'II', 'FLAG', 'OUP', 's');
TRUE = 1;
FALSE = 0;
fprintf(1,'This is Crout Method for tridiagonal linear systems.\n');
fprintf(1,'The array shall be written in code in advance with\n');
fprintf(1,'all diagonal entries, all lower sub-diagonal entries, all ');
fprintf(1,'upper sub-diagonal entries. The entry values of the\n');
fprintf(1,' right-hand-side shall be written in code in advance.\n\n');
fprintf(1,'Have all values been given? - enter Y or N.\n');
AA = input(' ','s');
OK = FALSE;
if AA == 'Y' | AA == 'y'
while OK == FALSE
% Matrix dimension N
N=71000000;
A = zeros(1,N);
B = zeros(1,N);
C = zeros(1,N);
BB = zeros(1,N);
X = zeros(1,N);
Z = zeros(1,N);
% A(I,I) is stored in A(I), 1 <= I <= n */
 for I = 1 : N
  A(I) = 5;
 end;
% the lower sub-diagonal A(I,I-1) is stored
%in B(I), 2 <= I <= n */
 for I = 2 : N
  B(I) = -1;
 end;
% the upper sub-diagonal A(I,I+1) is stored
%in C(I), 1 <= I <= n-1 */
 NN = N-1;
 for I = 1 : NN
  C(I) = -1;
 end;
 % right-hand-side B(I)  is stored in BB(I), 1 <= I <= n */
```

```
 BB(1)= 4; %B(1)
 for I = 2 : N-1
  BB(I) = 3; %B(I)
 end;
 BB(N)= 4; %B(N)
OK = TRUE;
end;
else
 fprintf(1,'The program will end so the matrix can be written in code.\n')
end;
if OK == TRUE
fprintf(1,'The code is executing now.\n');
t = cputime; %Starting time
% Steps 1-3 set up and solve LZ = B
% STEP 1
% the entries of U overwrite C and the entries of  L overwrite A
A(1);
C(1) = C(1)/A(1);
Z(1) = BB(1)/A(1);
% STEP 2
for I = 2 : NN
A(I) = A(I)-B(I)*C(I-1);
C(I) = C(I)/A(I);
Z(I) = (BB(I)-B(I)*Z(I-1))/A(I);
end;
% STEP 3
A(N) = A(N)-B(N)*C(N-1);
Z(N) = (BB(N)-B(N)*Z(N-1))/A(N);
% STEP 4
% STEPS 4, 5 solve UX = Z
X(N) = Z(N);
% STEP 5
for II = 1 : NN
I = NN-II+1;
X(I) = Z(I)-C(I)*X(I+1);
end;
eltime = cputime -t; %total elapse time
%
OUP = 1;
%Find the maximal error of computationed solution
ERRMAX=0.0;
for I=1:N
   TMP=abs(X(I)-1); %exact solution value
   if TMP > ERRMAX
      ERRMAX=TMP;
   end;
end;
 fprintf(OUP, '\nCROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS\n');
fprintf(OUP, '\nThe matrix dimension =  %d X %d \n', N,N);
fprintf(OUP, 'MAX ERROR=  %12.8e\n', ERRMAX);
if ERRMAX<0.00001,
fprintf(OUP, 'cpu time is %12.8e\n',eltime);
else
 fprintf(OUP, 'Something is wrong. \n');
end;
```

**b) M = 71 | N1 = 1,000,000 * 71 = 71,000,000 | Here is the code ran 4 times**



```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  71000000 X 71000000
MAX ERROR=  2.22044605e-16
cpu time is 1.29687500e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  71000000 X 71000000
MAX ERROR=  2.22044605e-16
cpu time is 1.28125000e+00
fx >>
```

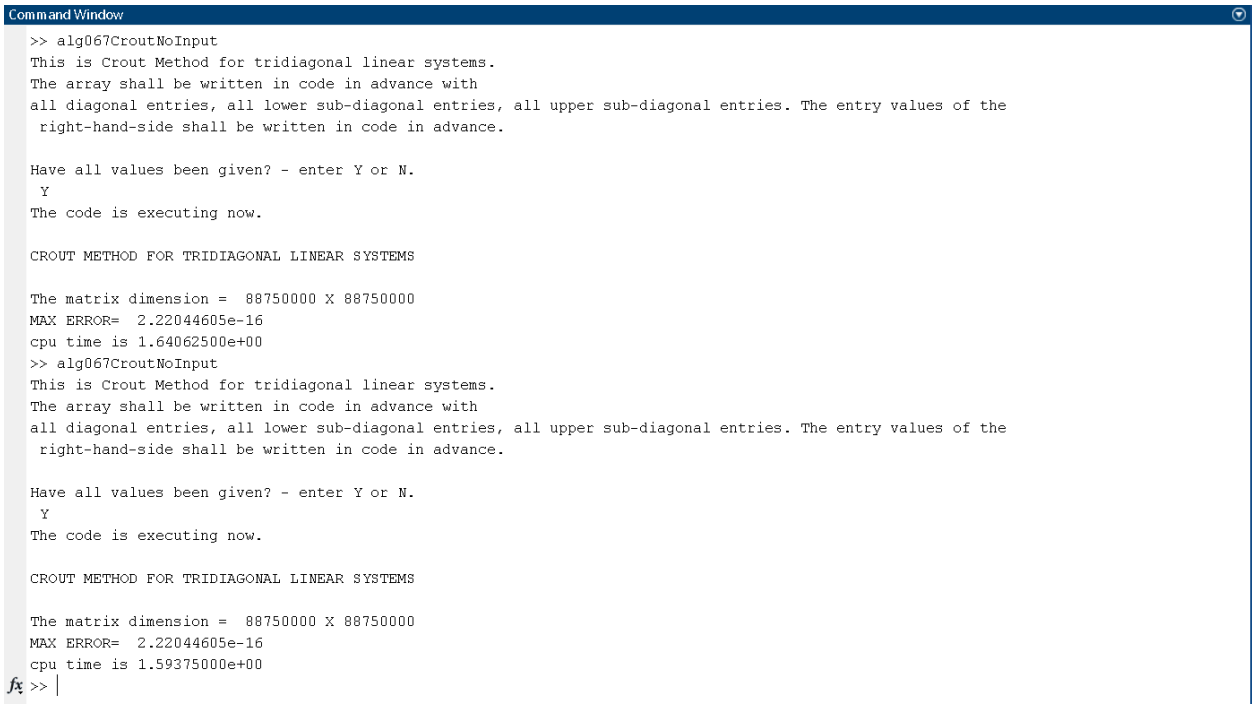


```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  71000000 X 71000000
MAX ERROR=  2.22044605e-16
cpu time is 1.15625000e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  71000000 X 71000000
MAX ERROR=  2.22044605e-16
cpu time is 1.15625000e+00
fx >>
```

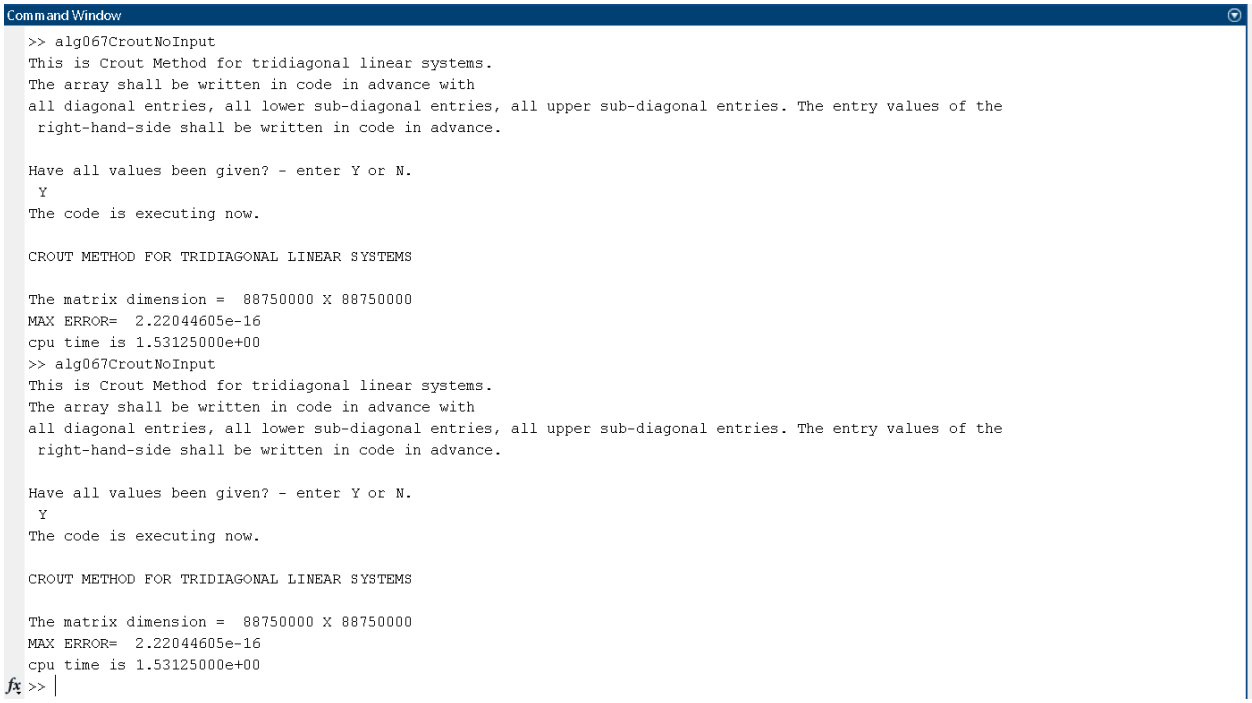
**E1 = 4.890625 / 4**

**E1 = 1.22265625**

**M = 71 | N2 = 1,250,000 * 71 = 88,750,000 | Here is the code ran 4 times**



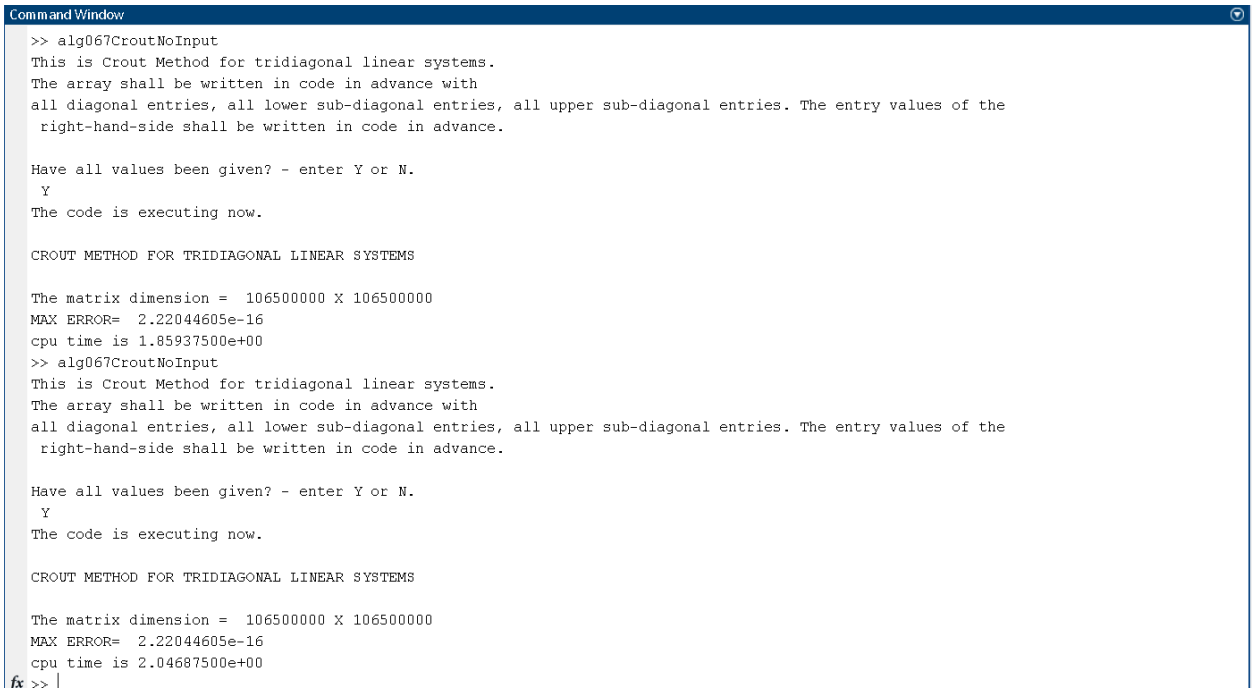
```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  88750000 X 88750000
MAX ERROR=  2.22044605e-16
cpu time is 1.64062500e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  88750000 X 88750000
MAX ERROR=  2.22044605e-16
cpu time is 1.59375000e+00
fx >>
```



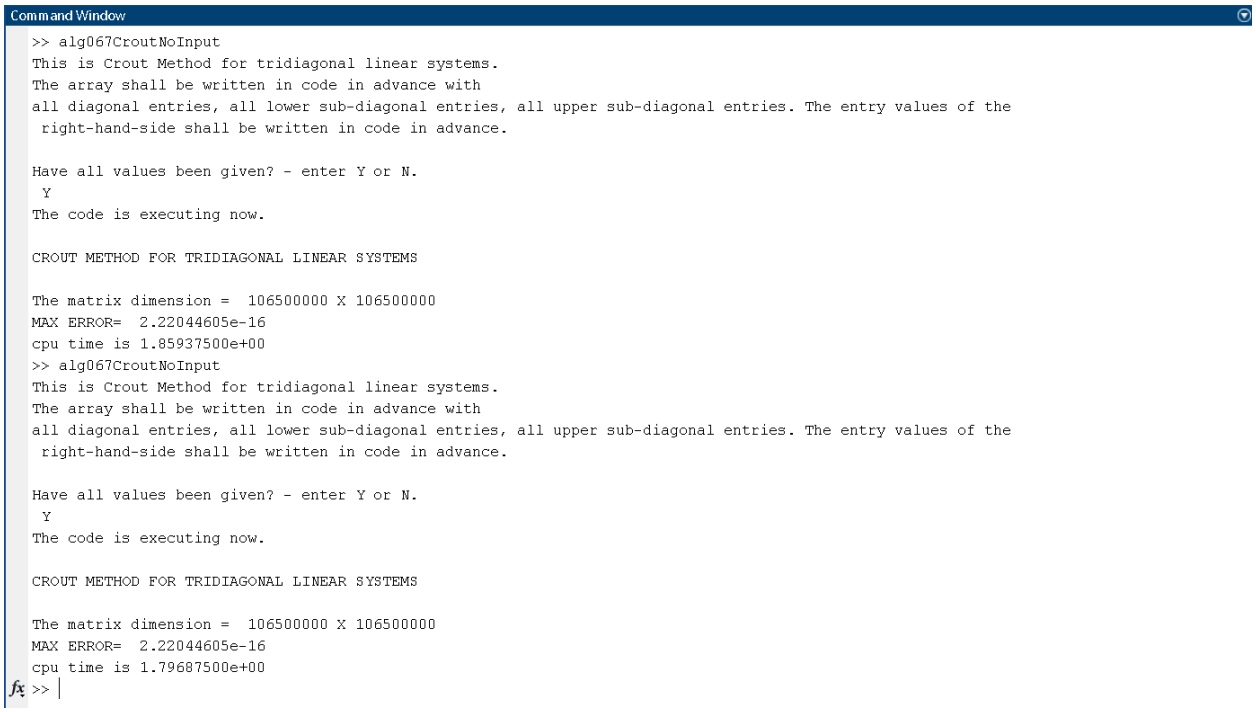
```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  88750000 X 88750000
MAX ERROR=  2.22044605e-16
cpu time is 1.53125000e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  88750000 X 88750000
MAX ERROR=  2.22044605e-16
cpu time is 1.53125000e+00
fx >>
```

**E2 = 6.296875 / 4**
**E2 = 1.57421875**

**M = 71 | N3 = 1,500,000 * 71 = 106,500,000 | Here is the code ran 4 times**



```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  106500000 X 106500000
MAX ERROR=  2.22044605e-16
cpu time is 1.85937500e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  106500000 X 106500000
MAX ERROR=  2.22044605e-16
cpu time is 2.04687500e+00
fx >>
```

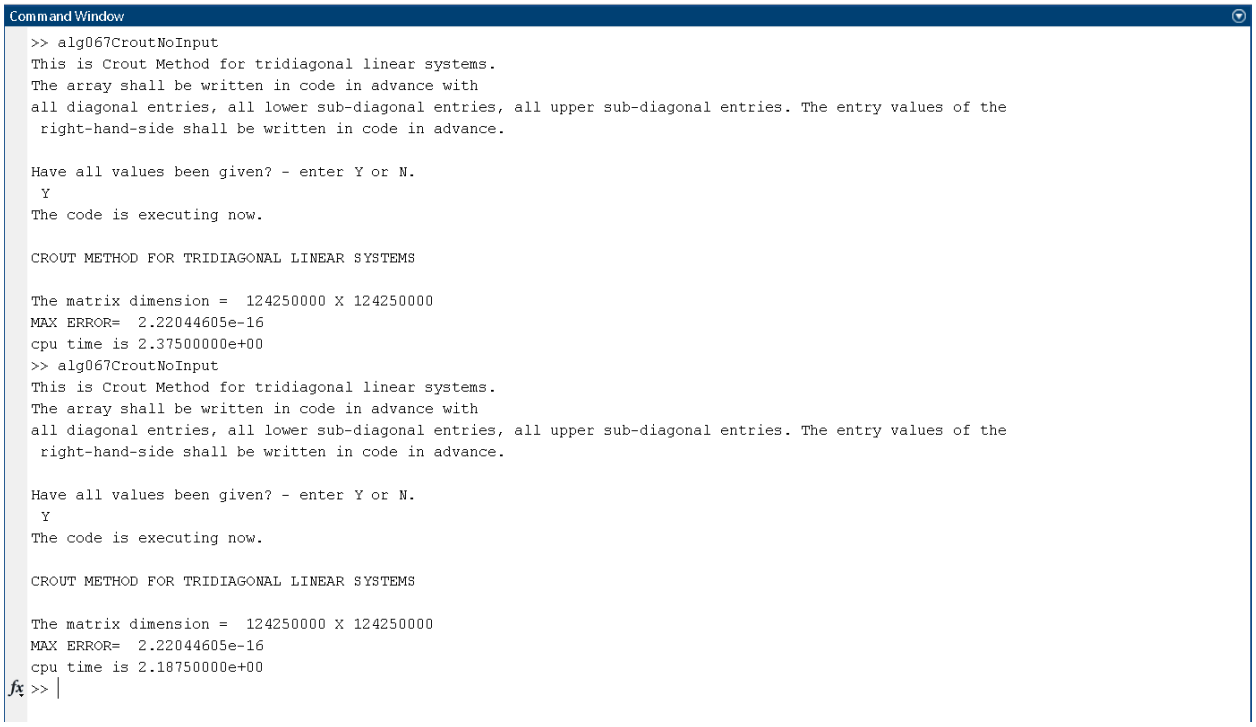


```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  106500000 X 106500000
MAX ERROR=  2.22044605e-16
cpu time is 1.85937500e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  106500000 X 106500000
MAX ERROR=  2.22044605e-16
cpu time is 1.79687500e+00
fx >>
```

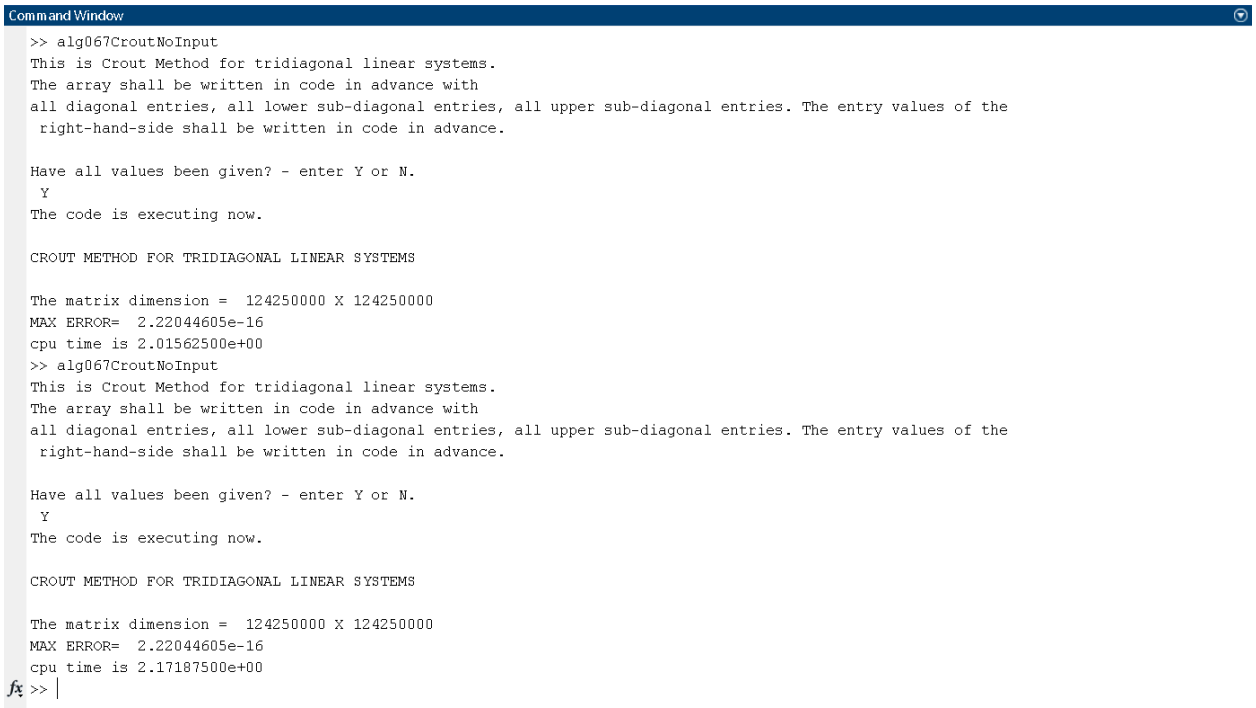
**E3 = 7.5625 / 4**

**E3 = 1.890625**

**M = 71 | N4 = 1,750,000 * 71 = 124,250,000 | Here is the code ran 4 times**



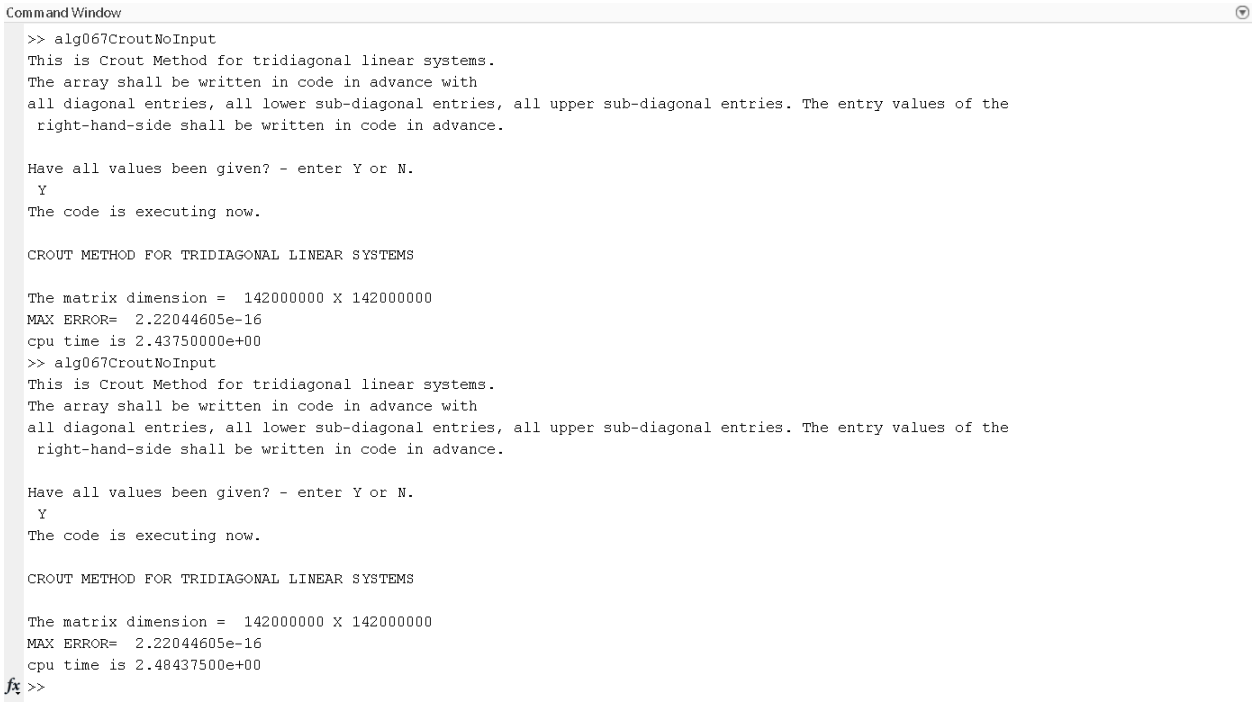
```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  124250000 X 124250000
MAX ERROR=  2.22044605e-16
cpu time is 2.37500000e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  124250000 X 124250000
MAX ERROR=  2.22044605e-16
cpu time is 2.18750000e+00
fx >>
```



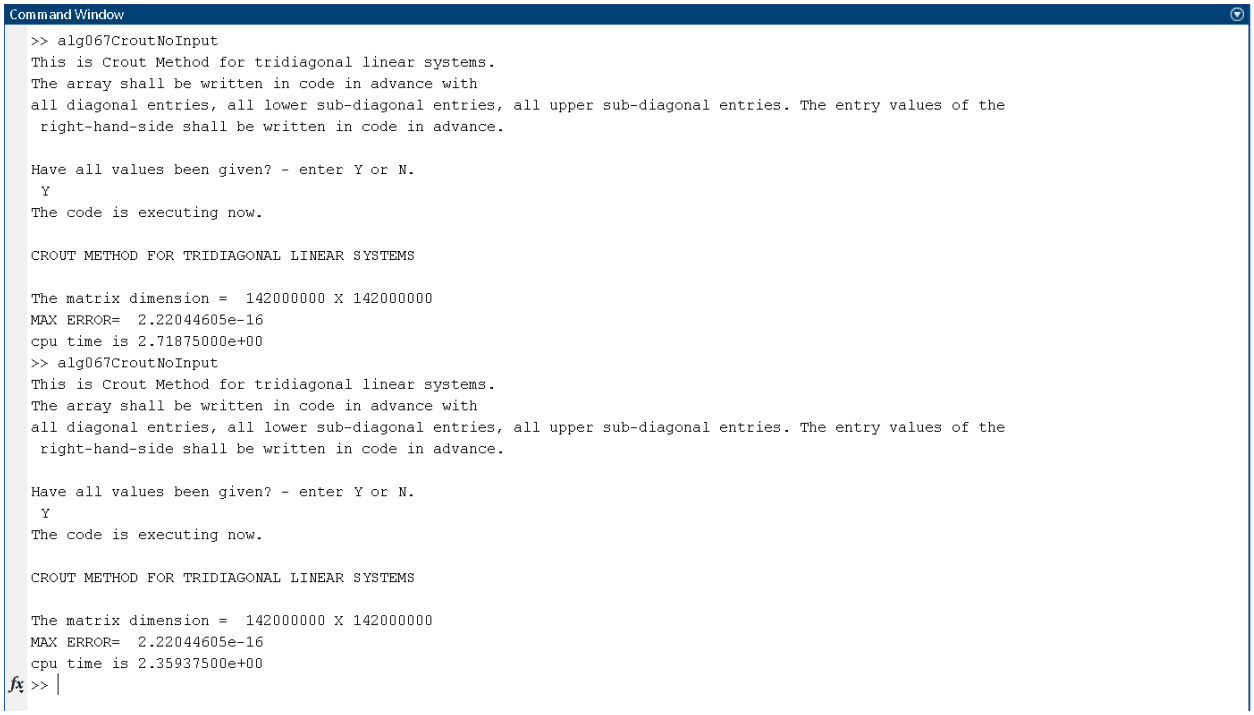
```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  124250000 X 124250000
MAX ERROR=  2.22044605e-16
cpu time is 2.01562500e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  124250000 X 124250000
MAX ERROR=  2.22044605e-16
cpu time is 2.17187500e+00
fx >>
```

**E4 = 8.75 / 4**
**E4 = 2.1875**

**M = 71 | N5 = 2,000,000 * 71 = 142,000,000 | Here is the code ran 4 times**



```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  142000000 X 142000000
MAX ERROR=  2.22044605e-16
cpu time is 2.43750000e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  142000000 X 142000000
MAX ERROR=  2.22044605e-16
cpu time is 2.48437500e+00
fx >>
```



```
Command Window
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  142000000 X 142000000
MAX ERROR=  2.22044605e-16
cpu time is 2.71875000e+00
>> alg067CroutNoInput
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  142000000 X 142000000
MAX ERROR=  2.22044605e-16
cpu time is 2.35937500e+00
fx >>
```

**E5 = 10 / 4**

**E5 = 2.5**

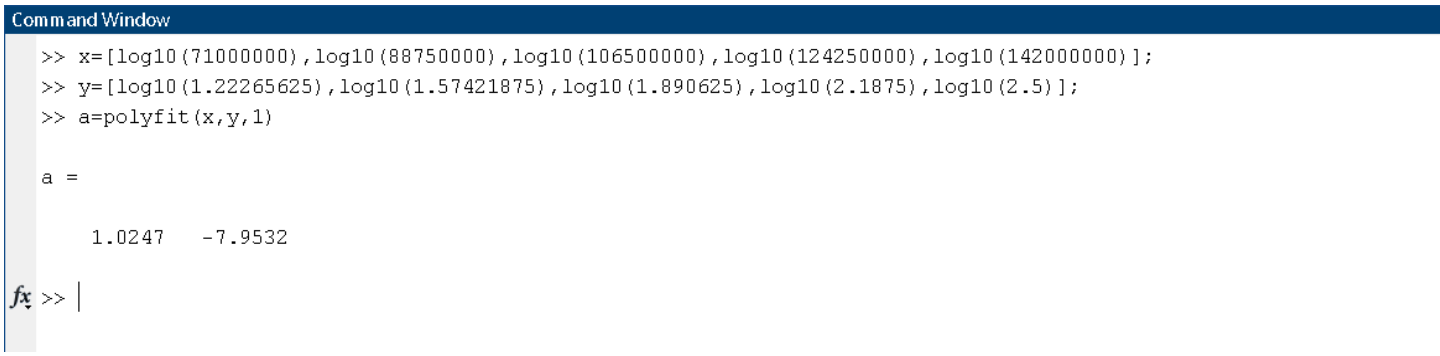
### i) Create Tables to Report Average Time Ei

| xxxxxxxxxxx | N1 = 71,000,000 | N2 = 88,750,000 | N3 = 106,500,000 | N4 = 124,250,000 | N5 = 142,000,000 |
|---|---|---|---|---|---|
| Time 1 | 1.296875 | 1.640625 | 1.859375 | 2.375 | 2.4375 |
| Time 2 | 1.28125 | 1.59375 | 2.046875 | 2.1875 | 2.484375 |
| Time 3 | 1.15625 | 1.53125 | 1.859375 | 2.015625 | 2.71875 |
| Time 4 | 1.15625 | 1.53125 | 1.796875 | 2.171875 | 2.359375 |
| Average Time (Ei) | 1.22265625 | 1.57421875 | 1.890625 | 2.1875 | 2.5 |

**E1 = 1.22265625**
**E2 = 1.57421875**
**E3 = 1.890625**
**E4 = 2.1875**
**E5 = 2.5**

---

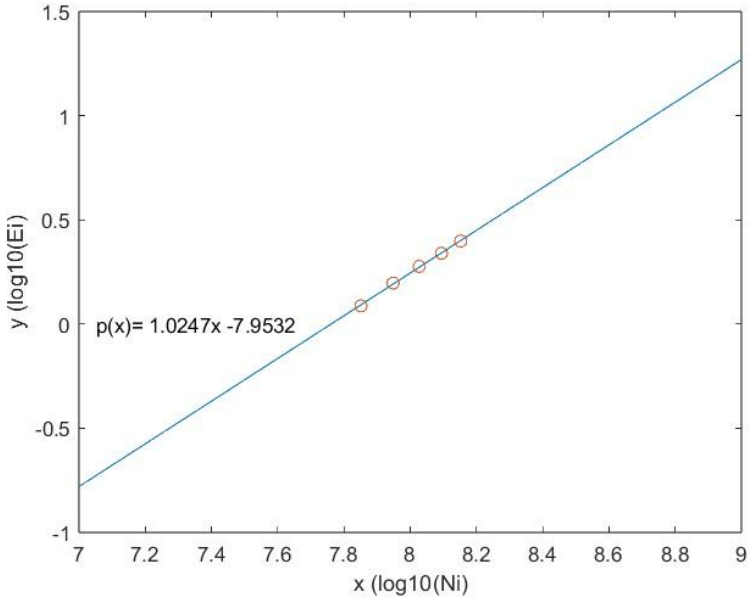
### ii) Line Fitting On Data Set {log10(Ni),log10(Ei))}




**p(x) = 1.0247x -7.9532**

---

**iii) Plot: a) the line obtained by line fitting & b) the data pairs**




**Code That Produced the Graph:**

```
>> t=linspace(7,9,101);
>> s=polyval(a,t);
>> plot(t,s);
>> hold on;
>> plot(x,y,'o');
>> xlabel('x (log10(Ni)')
>> ylabel('y (log10(Ei)')
>> text(7.05,0,'p(x)= 1.0247x -7.9532')
>>
```

---

**iv) What is the Slope of the Line You Obtained? Is it Close to 1?**

**Slope = 1.0247**

**This slope is close to 1. It is only slightly larger.**

---

**C) When Solving the Linear System, Explain Why Crouts Can Handle Very Large Scale Linear Systems Easily:**

**Gaussian has multiple 0 entries and the code does not understand this. The code still works with the 0; however, Crout ignores the 0's and only uses the diagonals. Thus, it is able to run faster.**

## II) Copy of Code With Value N1 = 10M

```
% CROUT FACTORIZATION FOR TRIDIAGONAL LINEAR SYSTEMS ALGORITHM 6.7
%
% To solve the n x n linear system
%
% E1:  A(1,1) X(1) + A(1,2) X(2)                  = A(1,n+1)
% E2:  A(2,1) X(1) + A(2,2) X(2) + A(2,3) X(3)    = A(2,n+1)
% :
% .
% E(n):          A(n,n-1) X(n-1) + A(n,n) X(n)    = A(n,n+1)
%
% INPUT:   the dimension n; the entries of A.
%
% OUTPUT:  the solution X(1), ..., X(N).
%   syms('AA', 'OK', 'NAME', 'INP', 'N', 'I', 'A', 'B', 'NN');
%   syms('C', 'BB', 'Z', 'X', 'II', 'FLAG', 'OUP', 's');
TRUE = 1;
FALSE = 0;
fprintf(1,'This is Crout Method for tridiagonal linear systems.\n');
fprintf(1,'The array shall be written in code in advance with\n');
fprintf(1,'all diagonal entries, all lower sub-diagonal entries, all ');
fprintf(1,'upper sub-diagonal entries. The entry values of the\n');
fprintf(1,' right-hand-side shall be written in code in advance.\n\n');
fprintf(1,'Have all values been given? - enter Y or N.\n');
AA = input(' ','s');
OK = FALSE;
if AA == 'Y' | AA == 'y'
while OK == FALSE
% Matrix dimension N
N=710;
A = zeros(1,N);
B = zeros(1,N);
C = zeros(1,N);
BB = zeros(1,N);
X = zeros(1,N);
Z = zeros(1,N);
% A(I,I) is stored in A(I), 1 <= I <= n */
 for I = 1 : N
  A(I) = 4;
 end;
% the lower sub-diagonal A(I,I-1) is stored
%in B(I), 2 <= I <= n */
 for I = 2 : N
  B(I) = -2;
 end;
% the upper sub-diagonal A(I,I+1) is stored
%in C(I), 1 <= I <= n-1 */
 NN = N-1;
 for I = 1 : NN
  C(I) = -2;
 end;
 % right-hand-side B(I)  is stored in BB(I), 1 <= I <= n */
 BB(1)= 20; %B(1)
```

```
 for I = 2 : N-1
  BB(I) = 0; %B(I)
 end;
 BB(N)= 20; %B(N)
OK = TRUE;
end;
else
 fprintf(1,'The program will end so the matrix can be written in code.\n')
end;
if OK == TRUE
fprintf(1,'The code is executing now.\n');
t = cputime; %Starting time
% Steps 1-3 set up and solve LZ = B
% STEP 1
% the entries of U overwrite C and the entries of  L overwrite A
A(1);
C(1) = C(1)/A(1);
Z(1) = BB(1)/A(1);
% STEP 2
for I = 2 : NN
A(I) = A(I)-B(I)*C(I-1);
C(I) = C(I)/A(I);
Z(I) = (BB(I)-B(I)*Z(I-1))/A(I);
end;
% STEP 3
A(N) = A(N)-B(N)*C(N-1);
Z(N) = (BB(N)-B(N)*Z(N-1))/A(N);
% STEP 4
% STEPS 4, 5 solve UX = Z
X(N) = Z(N);
% STEP 5
for II = 1 : NN
I = NN-II+1;
X(I) = Z(I)-C(I)*X(I+1);
end;
eltime = cputime -t; %total elapse time
%
OUP = 1;
%Find the maximal error of computationed solution
ERRMAX=0.0;
for I=1:N
   TMP=abs(X(I)-1); %exact solution value
   if TMP > ERRMAX
      ERRMAX=TMP;
   end;
end;
 fprintf(OUP, '\nCROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS\n');
fprintf(OUP, '\nThe matrix dimension =  %d X %d \n', N,N);
fprintf(OUP, 'MAX ERROR=  %12.8e\n', ERRMAX);
if ERRMAX<0.00001,
fprintf(OUP, 'cpu time is %12.8e\n',eltime);
else
 fprintf(OUP, 'Something is wrong. \n');
end;
```

Command Window

```
>> alg067CroutNoIn
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  710 X 710
MAX ERROR=  9.00000000e+00
Something is wrong.
```

Command Window

```
>> alg067CroutNoIn
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  71000 X 71000
MAX ERROR=  9.00000000e+00
Something is wrong.
fx >>
```

Command Window

```
>> alg067CroutNoIn
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  7100000 X 7100000
MAX ERROR=  9.00001392e+00
Something is wrong.
fx >>
```

```
Command Window
>> alg067CroutNoIn
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  71000000 X 71000000
MAX ERROR=  9.02755170e+00
Something is wrong.
fx >>
```

```
Command Window
>> alg067CroutNoIn
This is Crout Method for tridiagonal linear systems.
The array shall be written in code in advance with
all diagonal entries, all lower sub-diagonal entries, all upper sub-diagonal entries. The entry values of the
 right-hand-side shall be written in code in advance.

Have all values been given? - enter Y or N.
 Y
The code is executing now.

CROUT METHOD FOR TRIDIAGONAL LINEAR SYSTEMS

The matrix dimension =  124250000 X 124250000
MAX ERROR=  8.99999999e+00
Something is wrong.
```

**i) What Is The Maximum Error? Does The Error Grow As the Matrix Size Increases?**

**The Maximum error was from the N4 code which had an error of 9.0275517**

**From the N1 to N2 code, the error remained the same. From the N2 code to the N4 code, the error appeared to grow as the matrix size increased. However, the N5 code had the least error, which means that the error decreased from N4 to N5 code.**

**ii) Determine If the Matrix A of linear system S is strictly diagonally dominant:**

**Matrix A of the linear system (S) is strictly diagonally dominant. This is because each diagonal entry has a magnitude greater than all other numbers combined in that same row as the individual diagonal. In other words, |a11| > |a12| + |a13|... This statement is true for all diagonal entries.**