ΣΥΣΤΗΜΑΤΑ ΓΡΑΜΜΙΚΩΝ ΕΞΙΣΩΣΕΩΝ

Matlab-Octave

```
function x = GaussNaive(A,b)
% GaussNaive: naive Gauss elimination
% A = coefficient matrix
% b = right hand side vector
% x = solution vector
[m,n] = size(A);
if m~=n, error('Matrix A must be square'); end
nb = n+1;
Aug = [A b];
% forward elimination
for k = 1:n-1
for i = k+1:n
factor = Aug(i,k)/Aug(k,k);
Aug(i,k:nb) = Aug(i,k:nb)-factor*Aug(k,k:nb);
end
end
% back substitution
x = zeros(n,1);
x(n) = Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
x(i) = (Aug(i,nb)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
end
```

Παράδειγμα

>> a= [2 1 2; 1 2 3; 1 2 2]

a =

2 1 2

1 2 3

1 2 2

>> b= [5; 6; 5]

b =

5

6

5

>> x = GaussNaive(a,b)

x =

1

1

1

```
function x = GaussPivot(A,b)
% x = GaussPivot(A,b): Gauss elimination with pivoting.
% A = coefficient matrix
% b = right hand side vector
% x = solution vector
[m,n]=size(A);
if m~=n, error('Matrix A must be square'); end
nb=n+1;
Aug=[A b];
% forward elimination
for k = 1:n-1
% partial pivoting
[big,i]=max(abs(Aug(k:n,k)));
ipr=i+k-1;
if ipr~=k
Aug([k,ipr],:)=Aug([ipr,k],:);
end
for i = k+1:n
factor=Aug(i,k)/Aug(k,k);
Aug(i,k:nb)=Aug(i,k:nb)-factor*Aug(k,k:nb);
end
end
% back substitution
x=zeros(n,1);
x(n)=Aug(n,nb)/Aug(n,n);
for i = n-1:-1:1
x(i)=(Aug(i,nb)-Aug(i,i+1:n)*x(i+1:n))/Aug(i,i);
end
```

Παράδειγμα

```
>> a= [2 1 2; 1 2 3; 1 2 2]
a =
   2   1   2
   1   2   3
   1   2   2
>> b= [ 5; 6; 5]
b =
   5
   6
   5
>> x = GaussPivot(a,b)
x =
   1
   1
   1
```

Η Συνάρτηση rref δίνει την ανηγμένη κλιμακωτή μορφή

```
>> A=[1 2 3;4 5 6;7 8 0]
A =
   1   2   3
   4   5   6
   7   8   0
>> b=[1;2;3]
b =
   1
   2
   3

>> rref([A b])
ans =
   1.00000   0.00000   0.00000  -0.33333
   0.00000   1.00000   0.00000   0.66667
   0.00000   0.00000   1.00000  -0.00000
```

Η λύση στην τελευταία Στήλη του αποτελέσματος

Παράδειγμα (2)

```
>> A=[1 1;2 2]
A =
   1   1
   2   2
>> b=[2;3]
b =
   2
   3
>> rref([A b])
ans =
   1   1   0
   0   0   1
```

Το σύστημα είναι ασυμβίβαστο
και δεν υπάρχει λύση

Ενσωματωμένη συνάρτηση επίλυση συστήματος

```
>> A=[1 2 3;4 5 6;7 8 0]
A =
1 2 3
4 5 6
7 8 0
>> b=[1;2;3]
b =
1
2
3
>> x=A\b
x =
-0.3333
0.6667
0
```

Παραγοντοποίηση πινάκων DooLittle

```
function [L U] = Doolittle(A)
n = size(A,1);
L = zeros(n,n);            % initilize L and U to be n by n
  U = zeros(n,n);            % matrices with all zero entries.
  for k = 1:n
    L(k,k) = 1;
    for j = k:n
        U(k,j) = A(k,j);
      for s = 1:(k-1)
        U(k,j) = U(k,j) - L(k,s)*U(s,j);
      end
    end
    for i = (k+1):n
        L(i,k) = A(i,k);
      for s = 1:(k-1)
        L(i,k) = L(i,k) - L(i,s)*U(s,k);
      end
        L(i,k) = L(i,k) / U(k,k);
    end
  end
end
```

Παράδειγμα

```
>> A=[1 2 3;4 5 6;7 8 0]
A =
   1   2   3
   4   5   6
   7   8   0
>> [L U] = Doolittle(A)
L =
   1   0   0
   4   1   0
   7   2   1
U =
   1   2   3
   0  -3  -6
   0   0  -9
```

```
function x = DoolittleSystemSolution(A,b)
n = size(A,1);
L = zeros(n,n);             % initilize L and U to be n by n
  U = zeros(n,n);             % matrices with all zero entries.
  for k = 1:n
    L(k,k) = 1;
    for j = k:n
        U(k,j) = A(k,j);
      for s = 1:(k-1)
        U(k,j) = U(k,j) - L(k,s)*U(s,j);
      end
    end
    for i = (k+1):n
        L(i,k) = A(i,k);
      for s = 1:(k-1)
        L(i,k) = L(i,k) - L(i,s)*U(s,k);
      end
        L(i,k) = L(i,k) / U(k,k);
    end
  end
```

Συνέχεια…

```
% Step 2: Forward subsitiution to solve  L * y = b
y = zeros(n,1);              % Initialize y to be a column vector
y(1) = b(1);
  for i = 2:n
      y(i) = b(i);
    for j = 1:(i-1)
      y(i) = y(i) - L(i,j)*y(j);
    end
  end
% Step 3: Back subsitiution to solve  U * x = y
x = zeros(n,1);              % Initialize x to be a column vector
  x(n) = y(n) / U(n,n);
  for i = (n-1):-1:1
      x(i) = y(i);
    for j = (i+1):n
      x(i) = x(i) - U(i,j)*x(j);
    end
      x(i) = x(i) / U(i,i);
  end
end
```

Παράδειγμα

```
>> A=[1 2 3;4 5 6;7 8 0]
A =
   1   2   3
   4   5   6
   7   8   0
>> b=[1;2;3]
b =
   1
   2
   3
>> x = DoolittleSystemSolution(A,b)
x =
  -0.33333
   0.66667
  -0.00000
```

Παραγοντοποίηση πινάκων Crout

```
function [L,U] = crout(A)
[m, n] = size(A);
L=zeros(n,n);
U=zeros(n,n);
    for k = 1 : n
        L(k,k)=A(k,k)-L(k,1:k-1)*U(1:k-1,k);
            for j = k : n
                U(k,j) = ( A(k,j) - L(k,1:k-1)*U(1:k-1,j) )/ L(k,k);
            end;
         for i = k+1 : n
                L(i,k) = (A(i,k) - L(i,1:k-1)*U(1:k-1,k) ) / U(k,k);
         end
    end
```

Παράδειγμα

```
>> A=[1 2 3;4 5 6;7 8 0]
A =
   1   2   3
   4   5   6
   7   8   0
>> [L,U] = crout(A)
L =
   1   0   0
   4  -3   0
   7  -6  -9
U =
   1   2   3
   0   1   2
   0   0   1
```

Ενσωματωμένη συνάρτηση LU
[L, U, P] = lu(A)

A=[1 2 3;4 5 6;7 8 0]

A =

1 2 3

4 5 6

7 8 0

>> [L, U, P] = lu(A)

L =

1.00000 0.00000 0.00000

0.14286 1.00000 0.00000

0.57143 0.50000 1.00000

U =

7.00000 8.00000 0.00000

0.00000 0.85714 3.00000

0.00000 0.00000 4.50000

P =

Permutation Matrix

0 0 1

1 0 0

0 1 0

Εναλλαγή γραμμής

Παραγοντοποίηση Cholesky

```
>> A = [2,-1,0;-1,2,-1;0,-1,2]
A =
   2  -1   0
  -1   2  -1
   0  -1   2
>> chol(A)
ans =
   1.41421  -0.70711   0.00000
   0.00000   1.22474  -0.81650
   0.00000   0.00000   1.15470
```

```
function [x, iflag, itnum] = Jacobi(A,b,x0,delta,max_it)
%   x0: initial guess
%   delta: error tolerance for the relative difference between two consecutive iterates
%   max_it: maximum number of iterations to be allowed
% %   iflag: 1 if a numerical solution satisfying the error
%   itnum: the number of iterations used to compute x
n = length(b);
iflag = 1;
k = 0;
diagA = diag(A);
A = A-diag(diag(A));
while k < max_it
  k = k+1;
  x = (b-A*x0)./diagA
  relerr = norm(x-x0,inf)/(norm(x,inf)+eps);
  x0 = x;
  if relerr < delta
    break
  end
end
itnum = k;
if (itnum == max_it)
  iflag = -1
end
```

```
function x = GaussSeidel(A,b,es,maxit)
% A = coefficient matrix
% b = right hand side vector
% es = stop criterion (default = 0.00001%)
% maxit = max iterations (default = 50)
if nargin<2,error('at least 2 input arguments required'),end
if nargin<4|isempty(maxit),maxit=50;end
if nargin<3|isempty(es),es=0.00001;end
[m,n] = size(A);
if m~=n, error('Matrix A must be square'); end
C = A;
for i = 1:n
C(i,i) = 0;
x(i) = 0;
end
x = x';
```

Συνέχεια…

```
for i = 1:n
C(i,1:n) = C(i,1:n)/A(i,i);
end
for i = 1:n
d(i) = b(i)/A(i,i);
end
iter = 0;
while (1)
xold = x;
for i = 1:n
x(i) = d(i)-C(i,:)*x;
if x(i) ~= 0
ea(i) = abs((x(i) - xold(i))/x(i)) * 100;
end
end
iter = iter+1;
if max(ea)<=es | iter >= maxit, break, end
end
```

Νόρμες διανύσματος

Η p-νόρμα ($1 \le p < \infty$) διανύσματος $x \in \mathbb{R}^n$ ορίζεται ως εξής:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

Στην πράξη χρησιμοποιούνται συνήθως οι 1-, η 2- και η ∞-νόρμα για τις οποίες ισχύουν:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \qquad \|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}, \qquad \|x\|_\infty = \max_{1 \le i \le n} |x_i|$$

Παράδειγμα

```
>> x=[1 2 3 4]
x =
   1   2   3   4
>> norm(x,1)
ans =  10
>> norm(x,2)
ans =  5.4772
>> norm(x,inf)
ans =  4
```

Νόρμες Πινάκων

$$\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}|,$$

$$\|A\|_2 = \sqrt{r_\sigma\left(A^T A\right)},$$

$$\|A\|_\infty = \max_{1\le i\le n} \sum_{j=1}^{n} |a_{ij}|$$

Παράδειγμα

```
>> A= [ 2 -1 4;-2 3 -1;6 -1 4]
A =
   2  -1   4
  -2   3  -1
   6  -1   4
>> norm(A,1)
ans =  10
>> norm(A,2)
ans =  8.8141
>> norm(A,inf)
ans =  11
```

Δείκτης Κατάστασης

Ο **δείκτης κατάστασης** (condition number) ενός αντιστρέψιμου πίνακα A ορίζεται ως εξής:

$$\kappa_p(A) = \|A\|_p \left\|A^{-1}\right\|_p$$

Αποδεικνύεται ότι

$$\kappa_p(A) \geq 1$$

αν η χρησιμοποιούμενη νόρμα πίνακα είναι **φυσική**. Αν $\kappa_p(A) >> 1$ λέμε ότι ο A είναι **κακής κατάστασης** (ill or badly conditioned). Διαφορετικά λέμε ότι ο A είναι **καλής κατάστασης** (well conditioned).

Παράδειγμα

Η λύση του γραμμικού συστήματος

$$\begin{bmatrix} .780 & .563 \\ .913 & .659 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} .217 \\ .254 \end{bmatrix}$$

είναι $x_1 = 1$ και $x_2 = -1$. Πράγματι με τη MATLAB παίρνουμε:

```
>> A=[.780 .563; .913 .659]

A =
    0.7800    0.5630
    0.9130    0.6590

>> b=[.217; .254]

b =
    0.2170
    0.2540

>> x=A\b

 x =
     1.0000
    -1.0000
```

Παράδειγμα

Αν διαταράξουμε πολύ λίγο τον πίνακα των συντελεστών, και λύσουμε το σύστημα

$$\begin{bmatrix} .780 & .563001 \\ .913 & .659 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} .217 \\ .254 \end{bmatrix}$$

```
>> format long

>> A(1,2)=.563001
A =
   0.78000000000000   0.56300100000000
   0.91300000000000   0.65900000000000

>> x=A\b

x =
   8.57471264411556
 -11.49425287416920

>> cond(A,1)

ans =
  2.6614e+006
```