Rozwiązywanie układów równań CD

opracował dr. inż. Robert JAKUBOWSKI, Politechnika Rzeszowska

Table of Contents

W tej części zostaną przedstawione kolejne metody rozwiązywania układów równań liniowych.

Kolejnym podejściem do rozwiązywania układu równań jest rozkład układu: $A * X = B$ na układ w postaci:

$L * U * X = B$, gdzie U - jest macierzą trójkątną górną, a L jest macierzą trójkątną dolną. Wtedy otrzymuje się układ dwóch równań w postaci, gdzie pierwszy stanowi układ z macierzą trójkątną dolną L – rozwiązywany prze podstawienie w przód i drugi jest z macierzą trójkątną górną rozwiązywany prze podstawienie w tył.

$$L * Y = B$$
$$U * X = Y$$

Problem w tym wypadku polega na rozłożeniu macierzy A na macierz trjókątną dolną i górną.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ & & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ & & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} * \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ & & \ddots & \\ 0 & 0 & \cdots & u_{nn} \end{bmatrix}$$

Gdybyśmy się temu przyglądnęli to otrzymujemy następujące równania:

Dla pierwszego wiersza:

$$l_{11} * u_{11} = a_{11}$$
$$l_{11} * u_{12} = a_{12}$$
$$l_{11} * u_{13} = a_{13}$$
$$\vdots$$
$$l_{11} * u_{1n} = a_{1n}$$

Dla drugiego wiersza:

$$l_{21} * u_{11} = a_{21}$$
$$l_{21} * u_{12} + l_{22} * u_{22} = a_{22}$$
$$l_{21} * u_{13} + l_{22} * u_{23} = a_{23}$$
$$\vdots$$
$$l_{21} * u_{1n} + l_{22} * u_{2n} = a_{2n}$$

Gdy dalej rozpiszemy te równania, to możemy zauważyć, że w pierwszym równaniu dla pierwszego równania, drugim dla drugiego itd. występują dwie niewiadome l_{ii}, u_{ii} (znamy tylko wartości a_{ij}). Gdyby za jedną z tycz niewiadomych przyjąć jakąś wartość, to pozostałe elementy równań stają się proste do wyznaczenia. Np. przyjmując że l_{11}=1 to bez problemu wyznaczymy wartości u_{1i} dla pierwszego wiersza. Podobnie będzie w przypadku kolejnych równań. Gdy postawimy wartość za l_{ii} =1, to pozostałe równania stają się możliwe do rozwiązania.

Można zastosować w tej kwestii m.in, jedną z metod: rozkład Dolittle'a lub rozkład Crouta.

Rozkład Dolittle'a

Rozkład Dolittle'a polega na założeniu, że na głównej przekątnej macierzy dolnej L występują jedynki. Pozostałe elementy macierzy L i U wyznacza się dla kolejnych wierszy od pierwszego do ostatniego wg zależności:

$$i, j \in \{1 \cdots n\}$$
$$l_{ii} = 1$$
$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \right),$$
$$u_{ij} = a_{ij} - \sum_{k=i}^{n} l_{ik} u_{kj}$$

W Matlabie można to zapisać w postaci dwóch pętli iteracji. Pierwszej iterującej numery wierszy (zmienna i) i drugiej wewnętrznej iterującej numery kolumn j, rozbitej na dwie części. W tym wypadku sumy iloczynów iterowanych po zmiennej k zastępuje mnożenie wektorowe stąd:

```
for  i=1:n
L(i,i)=1
for j=1:i-1
L(i,j)=(A(i,j)-L(i,:)*U(:,j))/U(j,j);
end
for j=i:n
U(i,j)=A(i,j)-L(i,:)*U(:,j);
end
end
```

Przetestujmy to na konkretnym przypadku macierzy A. Dla późniejszych obliczeń od razu zdefiniujemy wektor B:

```
A=[1 2 5 1 2; ...
   2 7 1 1 2; ...
  -1 1 3 1 3; ...
   2 3 1 0 5;...
  -2 2 5 -1 0];
```

```
B=[2; 4; 5; -1; 3];
```

Przystępując do rozwiązania zadania przygotujmu zerową macierz L i U o wymiarze rwónym wymiarowi macierzy A

```
L=zeros(size(A))
```

```
L = 5×5
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```
U=zeros(size(A))
```

```
U = 5×5
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

Wypełnianie macierzy L i U odbywa się wierszami w dół. Dla pierwszej linii zrealizowanego kodu otrzymujemy

```
L(1,1)=1
```

```
L = 5×5
     1     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```
[n,~]=size(A);
% Wyznaczamy elementy 1 .. n w pierwszym wierszu wektora U. Ponieważ macierz L
% zawiera elementy rowne 0 dlatego do pierwszego wiersza macierzy U wysraczy
% przepisac elemty z pierwszego wiersza macierzy A
U(1,1:n)=A(1,1:n)
```

```
U = 5×5
     1     2     5     1     2
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

Dla drugiedo wiersza otrzymujemy:

```
L(2,2)=1;
% Wyznaczamy element pierwszy w drugim wierszu wektora l
L(2,1)=(A(2,1)-L(2,:)*U(:,1))/U(1,1)
```

```
L = 5×5
     1     0     0     0     0
     2     1     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```
     0     0     0     0     0
```

```
% Wyznaczamy elementy 2 .. n w drugim wierszu wektora U
for j=2:n
    U(2,j)=A(2,j)-L(2,:)*U(:,j);
end
U
```

```
U = 5×5
     1     2     5     1     2
     0     3    -9    -1    -2
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

W kolejnym kroku wyznaczamy wiersz trzeci macierzy L i U:

```
L(3,3)=1;
% Wyznaczamy elementy 1,2 w trzecim wierszu wektora L
for j=1:2
    L(3,j)=(A(3,j)-L(3,:)*U(:,j))/U(j,j);
end
L
```

```
L = 5×5
     1     0     0     0     0
     2     1     0     0     0
    -1     1     1     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```

% Wyznaczamy elementy 3 .. n w trzecim wierszu wektora u
for j=3:n
    U(3,j)=A(3,j)-L(3,:)*U(:,j);
end
U
```

```
U = 5×5
     1     2     5     1     2
     0     3    -9    -1    -2
     0     0    17     3     7
     0     0     0     0     0
     0     0     0     0     0
```

```

```

Dla kolejnych lini proces ten zautomatyzujemy. Poczynając od czwartego wiersza do ostatniego mamy:

```
for  i=4:n
    L(i,i)=1;
    for j=1:i-1
        L(i,j)=(A(i,j)-L(i,:)*U(:,j))/U(j,j);
    end
    for j=i:n
        U(i,j)=A(i,j)-L(i,:)*U(:,j);
    end
end
```

```
L
```

```
L = 5×5
    1.0000         0         0         0         0
    2.0000    1.0000         0         0         0
   -1.0000    1.0000    1.0000         0         0
    2.0000   -0.3333   -0.7059    1.0000         0
   -2.0000    2.0000    1.9412   13.0909    1.0000
```

```
U
```

```
U = 5×5
    1.0000    2.0000    5.0000    1.0000    2.0000
         0    3.0000   -9.0000   -1.0000   -2.0000
         0         0   17.0000    3.0000    7.0000
         0         0         0   -0.2157    5.2745
         0         0         0         0  -74.6364
```

Żeby sprawdzić poprawność dokonanego rozkładu należy porównać macierz A z iloczynem macierzy L i U

```
A
```

```
A = 5×5
     1     2     5     1     2
     2     7     1     1     2
    -1     1     3     1     3
     2     3     1     0     5
    -2     2     5    -1     0
```

```
SPR=L*U
```

```
SPR = 5×5
    1.0000    2.0000    5.0000    1.0000    2.0000
    2.0000    7.0000    1.0000    1.0000    2.0000
   -1.0000    1.0000    3.0000    1.0000    3.0000
    2.0000    3.0000    1.0000    0.0000    5.0000
   -2.0000    2.0000    5.0000   -1.0000         0
```

Dostaliśmy dwie identyczne macierze, co potwierdza poprawność wykonanych obliczeń

Rozwiązywanie układu równań po rozłożeniu na macierz trójkątną

Po rozłożeniu macierzy A na macierze trójkątną dolną L i górną U układ równań można przetransponować na dwa układy:

$$L * Y = B$$
$$U * X = Y$$

Rozpoczniemy rozwiązywanie od wyznaczenia wartości wektora Y rozwiązując pierwsze równanie. Jest to układ z macierzą trójkątną dolną, a zatem jego rozwiązanie będzie prowadzone z wykorzystaniem metody przez podstawienie w przód. Skopijuemy funkcję przygotowaną w poprzednim wykładzie do rozwiązywania układ równań z macierzą trójkątną dolną. Następnie wykorzystamy tę funkcję

```
[Y]=uklad_rown_MTD(L,B)
```

```
Y = 5×1
    2.0000
         0
```

```
     7.0000
    -0.0588
    -5.8182
```

W ostatnim kroku otrzymujemy ostateczne rozwiązanie rozwiązując drugi z układów. Tym razem układ zawiera macierz trójkątną górną, wiec rozwiązanie odbywa się przez podstawienie w stecz. Znowu wykorzystamy funkcję przygotowaną wcześniej którą skopiujemy i wykorzystamy

```
[X]=uklad_rown_MTG(U,Y)
```

```
X = 5×1
   -1.8380
    0.7637
   -0.0049
    2.1790
    0.0780
```

Na koniec pozostaje sprawdzenie poprawności rozwiązania. Zrobimy to jak poprzednio - obliczymy iloczyn A*X i porównamy z wektorem B

```
SPR=A*X
```

```
SPR = 5×1
    2.0000
    4.0000
    5.0000
   -1.0000
    3.0000
```

```
B
```

```
B = 5×1
     2
     4
     5
    -1
     3
```

Zgodność obydwu wektorów potwierdza poprawność rozwiązania.

Rozkład Crouta

Rozwiązywanie układu równań z rozkładem Crouta na macierz trójkątną górną i dolną jest realizowane w ten sam sposób. Jedyna różnica to sposób uzyskania macierzy trójkątnej górnej i dolnej.

W rozkładzie Crouta przyjmuje się, że jedynki występują na głównej przekątnej macierzy U, a zatem model obliczeń przedstawia się następująco:

$$i, j \in \{1 \cdots n\}$$
$$u_{ii} = 1$$

$$l_{ij} = a_{ij} - \sum_{k=1}^{i} l_{ik} u_{kj},$$

$$u_{ij} = \frac{1}{l_{ii}} \left(a_{ij} - \sum_{k=i+1}^{n} l_{ik} u_{kj} \right)$$

Rozwiązywanie układów równań z macierzą wstęgową trójelementową

Metodę rozkładu na macierz trójkątną górną i dolną można zastosować do rozwiązywania układów równań z macierzą wstęgową. Macierz wstęgowa, to taka, która zawiera elementy wzdłuż głównej przekątnej oraz taką samą ilość elementów w jedną i w drugą stronę od głównej przekątnej. Stąd nazwa macierz wstęgowa trójelementowa (zawiera element na głównej przekątnej i po jednym elemencie z prawej i z lewej) pięcioelementowa itp. Z zadaniami, gdzie pojawiają się duże macierze wstęgowe, często mamy do czynienia w zagadnieniach inżynierskich, dlatego aby przyśpieszyć rozwiązanie opracowane zostały specjalne algorytmy ukierunkowane na ten typ zadań. W opracowaniu zostanie przedstawiona metoda postępowania dla układu zawierającego macierz wstęgową trójelementową w oparciu o rozkład Dolittlea. Przykładowa macierz 8-o elementowa jest przedstawiona poniżej:

```
n=8;
A=zeros(n);
A(1,1)=rand();
A(1,2)=rand;
for i=2:n-1
    A(i,[i-1,i+1])=rand;
    A(i,i)=rand;
end
A(n,n-1:n)=rand(2,1)
```

```
A = 8×8
    0.1712    0.7060         0         0         0         0         0         0
    0.0318    0.2769    0.0318         0         0         0         0         0
         0    0.0462    0.0971    0.0462         0         0         0         0
         0         0    0.8235    0.6948    0.8235         0         0         0
         0         0         0    0.3171    0.9502    0.3171         0         0
         0         0         0         0    0.0344    0.4387    0.0344         0
         0         0         0         0         0    0.3816    0.7655    0.3816
         0         0         0         0         0         0    0.7952    0.1869
```

Do tego dołóżmy 10-o elementowy wektor B stanowiący drugą stronę układu równań:

```
B=rand(n,1)
```

```
B = 8×1
    0.4898
    0.4456
    0.6463
    0.7094
    0.7547
    0.2760
    0.6797
    0.6551
```

Najpierw zastosujmy metodę rozkładu macierzy wstęgowej trójelementowej w oparciu o metodę Dolittle'a.

W tym celu przygotujemy 10-o elementowe macierze zerowe L i U:

```
L=zeros(n);
```

```
U=zeros(n);
```

Następnie wg wzoru Dolittle'a zajmiemy się tylko operacjami wokół głównej przekątnej, bowiem elementy macierzy A zawierające 0 też będą zerowe w macierzach L i U. Podany poniżej przykład kodu pozwala zrealizować rozkład na macierz trójkątną górną i dolną dla macierzy wstęgowej trójelementowej

```
%Pierwszy wiersz
L(1,1)=1;
U(1,1:2)=A(1,1:2);
%drugi wiersz
L(2,2)=1;
L(2,1)=A(2,1)/U(1,1);
U(2,2)=A(2,2)-L(2,1)*U(1,2);
U(2,3)=A(2,3);

for i=3:n-1
    L(i,i)=1;
    L(i,i-1)=A(i,i-1)/U(i-1,i-1);
    U(i,i)=A(i,i)-L(i,i-1)*U(i-1,i);
    U(i,i+1)=A(i,i+1);
end
% wiersz ostatni
L(n,n)=1;
L(n,n-1)=A(n,n-1)/U(n-1,n-1);
U(n,n)=A(n,n)-L(n,n-1)*U(n-1,n);

U
```

```
U = 8×8
    0.1712    0.7060         0         0         0         0         0         0
         0    0.1456    0.0318         0         0         0         0         0
         0         0    0.0870    0.0462         0         0         0         0
         0         0         0    0.2580    0.8235         0         0         0
         0         0         0         0   -0.0618    0.3171         0         0
         0         0         0         0         0    0.6154    0.0344         0
         0         0         0         0         0         0    0.7442    0.3816
         0         0         0         0         0         0         0   -0.2209
```

```
L
```

```
L = 8×8
    1.0000         0         0         0         0         0         0         0
    0.1860    1.0000         0         0         0         0         0         0
         0    0.3170    1.0000         0         0         0         0         0
         0         0    9.4608    1.0000         0         0         0         0
         0         0         0    1.2290    1.0000         0         0         0
         0         0         0         0   -0.5573    1.0000         0         0
         0         0         0         0         0    0.6200    1.0000         0
         0         0         0         0         0         0    1.0686    1.0000
```

Spraawdzmy czy zrobiliśmy to poprawnie porównując iloczyn LxU do wyjściowej macierzy A

```
SPR=L*U
```

```
SPR = 8×8
    0.1712    0.7060         0         0         0         0         0         0
    0.0318    0.2769    0.0318         0         0         0         0         0
         0    0.0462    0.0971    0.0462         0         0         0         0
```

```
         0         0    0.8235    0.6948    0.8235         0         0         0
         0         0         0    0.3171    0.9502    0.3171         0         0
         0         0         0         0    0.0344    0.4387    0.0344         0
         0         0         0         0         0    0.3816    0.7655    0.3816
         0         0         0         0         0         0    0.7952    0.1869
```

```
A
```

```
A = 8×8
    0.1712    0.7060         0         0         0         0         0         0
    0.0318    0.2769    0.0318         0         0         0         0         0
         0    0.0462    0.0971    0.0462         0         0         0         0
         0         0    0.8235    0.6948    0.8235         0         0         0
         0         0         0    0.3171    0.9502    0.3171         0         0
         0         0         0         0    0.0344    0.4387    0.0344         0
         0         0         0         0         0    0.3816    0.7655    0.3816
         0         0         0         0         0         0    0.7952    0.1869
```

Rozwiązanie układu równań też możemy istotnie uprościć. Stosując podejście jak do rozwiązania układu równań z rozkładem na macierz trójkątną górną i dolną otrzymamy z równania L*y=B, gdy macierz L ma dwa elementy wektora Y

$$y_1 = b_1$$
$$i \in \{2 \cdots n\}$$
$$y_i = b_i - y_{i-1} l_{i,i-1}$$

czyli w zapisie Matlab:

```
Y=zeros(n,1);
Y(1,1)=B(1);
for i=2:n
    Y(i,1)=B(i)-Y(i-1)*L(i,i-1);
end
Y
```

```
Y = 8×1
    0.4898
    0.3545
    0.5339
   -4.3419
    6.0909
    3.6702
   -1.5957
    2.3602
```

W kolejnym kroku rozwiązuje się układ z macierzą trójkątną górną U*X=Y. Ponieważ U jest ograniczona do 2 elementów, dlatego wystarczy cały algorytm uprościć do następującego postępowania realizowanego od ostatniego do pierwszego elementu:

$$x_n = \frac{y_n}{u_{\mathrm{nn}}}$$
$$i \in \{n-1 \cdots 1\}$$
$$x_i = \frac{(y_i - x_{i+1} u_{i,i+1})}{u_{\mathrm{ii}}}$$

stąd

```
X=zeros(n,1);
X(n,1)=Y(n)/U(n,n);
for i=n-1:-1:1
    X(i,1)=(Y(i)-X(i+1)*U(i,i+1))/U(i,i);
end
```

Rozwiązaniem jest wektor X.

Sprawdźmy poprawność rozwiązania. Pomnóżmy macierz A przez X i porównajmy z wektorem B.

```
X
```

```
X = 8×1
  -98.7663
   24.6404
 -101.5897
  203.0742
  -68.9016
    5.7768
    3.3352
  -10.6868
```

```
SPR=A*X
```

```
SPR = 8×1
    0.4898
    0.4456
    0.6463
    0.7094
    0.7547
    0.2760
    0.6797
    0.6551
```

```
B
```

```
B = 8×1
    0.4898
    0.4456
    0.6463
    0.7094
    0.7547
    0.2760
    0.6797
    0.6551
```

Wykorzystanie macierzy odwrotnej w rozwiązywaniu układów równań

Kolejną metodą rozwiązywania układów równań jest wykorzystanie macierzy odwrotnej. Jeżeli układ równań zapisany jest w postaci macierzowej:

$$A * X = B$$

to X można wyznaczyć bezpośrednio, z równania:

$X = A^{-1} * B$

W Matlabie wyznaczenie macierzy odwrotnej jest bardzo proste. Załóżmy, że macierz A i wektor B są następujące:

```
A=[2  3  5  4;...
  -1  2 -2  3;...
   3  1  1 -2;...
   2  1  3 -1];
B=[2 3 4 1]';
```

Macierz odwrotna MO w Matlabie wylicza się z zależności:

```
MO=A^-1
```

```
MO = 4×4
    0.3333   -0.3333    0.6667   -1.0000
   -0.3333    0.6111   -0.2778    1.0556
    0.0000   -0.0556   -0.2778    0.3889
    0.3333   -0.2222    0.2222   -0.7778
```

Rozwiązanie jest więc

```
X=MO*B
```

```
X = 4×1
    1.3333
    1.1111
   -0.8889
    0.1111
```

Sprawdźmy poprawność rozwiązania:

```
SPR=A*X
```

```
SPR = 4×1
    2.0000
    3.0000
    4.0000
    1.0000
```

```
B
```

```
B = 4×1
     2
     3
     4
     1
```

Wynik wyszedł poprawny.

Jak otrzymać macierz odwrotną, gdy nie mamy takiego narzędzia jak Matlab? Jedną z metod jest wykorzystanie eliminacji Gaussa-Jordana. Metoda polega na tym, że porównuje się macierz A z macierzą jednostkową o tym samym wymiarze. Następnie wykonuje się operacje jak w eliminacji Gaussa-Jordana. Po sprowadzeniu macierzy A do macierzy jednostkowej z drugiej strony otrzymuje się macierz odwrotną do macierzy A.

Skopiujmy do funkcji funkcję do eliminacji Gaussa-Jordana. Następnie przygotujmy macierz jednostkową MJ o wymiarze macierzy A. Po podstawieniu obydwu macierzy i uruchomieniu funkcji otrzymamy z MJ macierz odwrotną MO

```
[n,~]=size(A);
MJ=eye(n)
```

```
MJ = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
[AJ,MO]=eliminacja_G_J(A,MJ);
MO
```

```
MO = 4×4
    0.3333   -0.3333    0.6667   -1.0000
   -0.3333    0.6111   -0.2778    1.0556
         0   -0.0556   -0.2778    0.3889
    0.3333   -0.2222    0.2222   -0.7778
```

Otrzymaną w ten sposób macierz odwrotną MO możemy porównać z macierzą odwrotną wyznaczoną z funkcji Matlaba

```
A^-1
```

```
ans = 4×4
    0.3333   -0.3333    0.6667   -1.0000
   -0.3333    0.6111   -0.2778    1.0556
    0.0000   -0.0556   -0.2778    0.3889
    0.3333   -0.2222    0.2222   -0.7778
```

Teraz aby dokończyć rozwiązywanie wystarczy pomnożyć MO przez wektor B.

```
X=MO*B
```

```
X = 4×1
    1.3333
    1.1111
   -0.8889
    0.1111
```

Pokazane metody nie wyczerpują tematu. Przedstawiono tylko wybrane metody, oczywiście są inne podejścia do rozwiązywania układów równań, w tym metody iteracyjne, których w ogóle nie poruszono w tym opracowaniu.

Funkcje dodatkowe

Funkcja do rozwiązywania układów równań z macierzą trójkątną dolną

```
function [X]=uklad_rown_MTD(L,B)
%Funkcja wyznacza rozwiązanie dla ukadu rwnań z macierzą trjkątną dolną

    [i,~]=size(L);
    X=zeros(i,1);
```

```
    for k=1:i
        c=L(k,:)*X;
        X(k)=(B(k)-c)/L(k,k);
    end
end
```

Funkcja do rozwiązywania układów równań z macierzą trójkątną górną

```
function [X]=uklad_rown_MTG(U,B)
%Funkcja wyznacza rozwiązanie dla ukadu rwnań z macierzą trjkątną grną

    [i,~]=size(U);
    X=zeros(i,1);
    for k=i:-1:1
        c=U(k,:)*X;
        X(k)=(B(k)-c)/U(k,k);
    end
end
```

Funkcja realizująca eliminację Gaussa-Jordana

```
function [a1,b1]=eliminacja_G_J(a,b)
n=length(b);
    for i=1:n
        % Część funkcji realizująca zamianę elementu z gwnej przekątnej w i-tej kolumnie
        % na element o najwyższej wartości bezwzględnej
        [~,im]=max(abs(a(i:n,i)));
        im=im+i-1;
        if im~=i
            am=a(im,:);
            a(im,:)=a(i,:);
            a(i,:)=am;
            am=b(im,:);
            b(im,:)=b(i,:);
            b(i,:)=am;
        end

        % Waściwa część eliminacji Gaussa-Jordana
        b(i,:)=b(i,:)/a(i,i);
        a(i,:)=a(i,:)/a(i,i);

        c=a(:,i);
        c(i,1)=0;
        a=a-c*a(i,:);
        b=b-c*b(i,:);
    end
    a1=a;
    b1=b;
end
```