MATLAB PROJECT 2

GROUP # 27

1. **David Rowe**
2. **Jake Sanchez**
3. **Charles Richardson**
4. **Brandon Miguel**
5. **Nicolas Santiago**
6. **Nic Morita**

Part 1. Elementary Row Operations

Exercise 1

```
% Display functions
type ele1
```

```
function E1 = ele1(n, r, i, j)
E1 = eye(n);
E1(j,:) = (E1(j,:) + (E1(i,:)*r));
end
```

```
type ele2
```

```
function E2 = ele2(n, i, j)
E2 = eye(n);
temp = E2(j,:);
E2(j,:) = E2(i,:);
E2(i,:) = temp;
end
```

```
type ele3
```

```
function E3 = ele3(n, j, k)
E3 = eye(n);
E3(j,:) = E3(j,:)*k;
end
```

```
format
format compact
% Reduce matrix A with functions ele1, ele2, and ele3.
A = [0 1 3 1; 2 4 6 -2; 3 1 4 2]
```

```
A = 3×4
     0     1     3     1
     2     4     6    -2
     3     1     4     2
```

```
E2 = ele2(3,1,2);
A1 = E2*A
```

```
A1 = 3×4
     2     4     6    -2
```

```
     0     1     3     1
     3     1     4     2
```

```
E2 = ele2(3,1,3);
A2 = E2*A1
```

```
A2 = 3×4
     3     1     4     2
     0     1     3     1
     2     4     6    -2
```

```
E1 = ele1(3,-2/3,1,3);
A3 = E1*A2
```

```
A3 = 3×4
    3.0000    1.0000    4.0000    2.0000
         0    1.0000    3.0000    1.0000
         0    3.3333    3.3333   -3.3333
```

```
E2 = ele2(3,2,3);
A4 = E2*A3;
E1 = ele1(3,-3/10,2,3);
A5 = E1*A4
```

```
A5 = 3×4
    3.0000    1.0000    4.0000    2.0000
         0    3.3333    3.3333   -3.3333
         0         0    2.0000    2.0000
```

```
E3 = ele3(3,3,1/2);
A6 = E3*A5
```

```
A6 = 3×4
    3.0000    1.0000    4.0000    2.0000
         0    3.3333    3.3333   -3.3333
         0         0    1.0000    1.0000
```

```
E1 = ele1(3,-10/3,3,2);
A7 = E1*A6
```

```
A7 = 3×4
    3.0000    1.0000    4.0000    2.0000
         0    3.3333         0   -6.6667
         0         0    1.0000    1.0000
```

```
E1 = ele1(3,-4,3,1);
A8 = E1*A7
```

```
A8 = 3×4
    3.0000    1.0000         0   -2.0000
         0    3.3333         0   -6.6667
         0         0    1.0000    1.0000
```

```
E3 = ele3(3,2,3/10);
A9 = E3*A8
```

```
A9 = 3×4
    3.0000    1.0000         0   -2.0000
         0    1.0000         0   -2.0000
         0         0    1.0000    1.0000
```

```
E1 = ele1(3,-1,2,1);
A10 = E1*A9
```

```
A10 = 3×4
    3.0000         0         0   -0.0000
         0    1.0000         0   -2.0000
         0         0    1.0000    1.0000
```

```
E3 = ele3(3,1,1/3);
A11 = E3*A10
```

```
A11 = 3×4
    1.0000         0         0   -0.0000
         0    1.0000         0   -2.0000
         0         0    1.0000    1.0000
```

```
% Use a logical statement to check if matrix A11 matches built-in function of MATLAB
check = rref(A);
if round(A11)==check
    disp(' Reduced echelon form matrices match.')
else
    disp('Matrices do not match, check work.')
end
```

```
 Reduced echelon form matrices match.
```

Part 2. Basic Operations

Exercise 2

```
format short
type inverses
```

```
function F=inverses(A)
    [row,column]=size(A);
    k=rank(A);
    F=[];
    B=A;
    if row==column
        if row==k
            A= [A eye(row)];
            R=rref(A);
            F= R(:,row+1:end);
        else
            disp("matrix A is not invertible")
            F=[];
        end
    else
        disp("Matrix A is not Invertible")
        F=[];
    end
    if ~isempty(F)
        P=inv(B);
        F=round(F,5);
        P=round(P,5);
        P
        disp("P is the inverse of A calculated using the built in MATLAB function")

        if F==P
```

```
            disp("F and P are equal")
        else
            disp("F and P aren't equal")
        end
    end
end
```

```
type closetozeroroundoff
```

```
function B = closetozeroroundoff(A,p)
A(abs(A)<10^-p)=0;
B=A;
end
```

(a)

```
A=[4 0 -7 -7;-6 1 11 9;7 -5 10 19;-1 2 3 -1]
```

```
A = 4×4
     4     0    -7    -7
    -6     1    11     9
     7    -5    10    19
    -1     2     3    -1
```

```
inverses(A)
```

```
P = 4×4
   -19   -14     0     7
  -549  -401    -2   196
   267   195     1   -95
  -278  -203    -1    99
P is the inverse of A calculated using the built in MATLAB function
F and P are equal
ans = 4×4
   -19   -14     0     7
  -549  -401    -2   196
   267   195     1   -95
  -278  -203    -1    99
```

(b)

```
A=[1 -3 2 -4;-3 9 -1 5;2 -6 4 -3;-4 12 2 7]
```

```
A = 4×4
     1    -3     2    -4
    -3     9    -1     5
     2    -6     4    -3
    -4    12     2     7
```

```
inverses(A)
```

```
matrix A is not invertible

ans =

     []
```

(c)

```
A=magic(3)
```

```
A = 3×3
     8     1     6
     3     5     7
     4     9     2
```

```
inverses(A)
```

```
P = 3×3
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028
P is the inverse of A calculated using the built in MATLAB function
F and P are equal
ans = 3×3
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028
```

(d)

```
A=hilb(5);
inverses(A)
```

```
P = 5×5
          25        -300        1050       -1400         630
        -300        4800      -18900       26880      -12600
        1050      -18900       79380     -117600       56700
       -1400       26880     -117600      179200      -88200
         630      -12600       56700      -88200       44100
P is the inverse of A calculated using the built in MATLAB function
F and P are equal
ans = 5×5
          25        -300        1050       -1400         630
        -300        4800      -18900       26880      -12600
        1050      -18900       79380     -117600       56700
       -1400       26880     -117600      179200      -88200
         630      -12600       56700      -88200       44100
```

(e)

```
A=magic(6);
inverses(A)
```

```
matrix A is not invertible

ans =

     []
```

Part 3. Solving Equations

Exercise 3

Part 1

```
type solvesys
```

```
function [C,N] = solvesys(A,b)
[~,n]=size(A);
```

```
format long

%Check if matrix is not invertible = # pivot pos <  # cols in A
if (rank(A) ~= n)
   disp('A is not invertible')
   C = [];
   N = [];
   if (rank(A) < rank([A,b]))   % if there is a pivot pos in b
       disp('A is inconsistent');
   elseif (rank([A,b]) < n)         % if there is a free variable
       disp('[A b] has infinitely many solutions');
   end
   return
end

%Matrix is invertible, thus solve for each x using different solving
%methods
x = A\b;
x1 = x(1);
x = inv(A)*b;
x2 = x(2);
x = rref([A,b]);
x3 = x(3, end);
C = [x1, x2, x3];
disp('Solutions obtained by different methods are the columns of');
C

%Return a column vector
n1 = norm(x1 - x2);
n2 = norm(x2 - x3);
n3 = norm(x3 - x1);
N = [n1;n2;n3];
disp('Norms of differences between solutions are the entires of');
N
end
```

(a)

```
A=magic(4);I=eye(size(A,1));b=I(:,end)
```

```
b = 4×1
     0
     0
     0
     1
```

```
[C,N]=solvesys(A,b);
```

```
A is not invertible
A is inconsistent
```

(b)

```
A=magic(4),b=A(:,end)
```

```
A = 4×4
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
b = 4×1
    13
     8
```

```
    12
     1
```

```
[C,N]=solvesys(A,b);
```

```
A is not invertible
[A b] has infinitely many solutions
```

(c)

```
A=magic(5);b=fix(10*rand(size(A,1),1))
```

```
b = 5×1
     8
     2
     9
     3
     1
```

```
[C,N]=solvesys(A,b);
```

```
Solutions obtained by different methods are the columns of
C = 1×3
  -0.248782051282051   0.268525641025641  -0.162561576354680
Norms of differences between solutions are the entires of
N = 3×1
   0.517307692307692
   0.431087217380321
   0.086220474927371
```

(d)

```
A=eye(6);b=fix(10*rand(size(A,1),1))
```

```
b = 6×1
     2
     6
     4
     3
     8
     5
```

```
[C,N]=solvesys(A,b);
```

```
Solutions obtained by different methods are the columns of
C = 1×3
     2     6     4
Norms of differences between solutions are the entires of
N = 3×1
     4
     2
     2
```

(e)

```
A=magic(7)
```

```
A = 7×7
    30    39    48     1    10    19    28
    38    47     7     9    18    27    29
    46     6     8    17    26    35    37
     5    14    16    25    34    36    45
```

```
    13    15    24    33    42    44     4
    21    23    32    41    43     3    12
    22    31    40    49     2    11    20
```

```
b=fix(10*rand(size(A,1),1))
```

```
b = 7×1
     5
     9
     2
     7
     7
     3
     5
```

```
[C,N]=solvesys(A,b);
```

```
Solutions obtained by different methods are the columns of
C = 1×3
  -0.083182249644044   0.185090175605126  -0.047468354430380
Norms of differences between solutions are the entires of
N = 3×1
   0.268272425249169
   0.232558530035505
   0.035713895213664
```

(f)

```
A=hilb(7)
```

```
A = 7×7
   1.000000000000000   0.500000000000000   0.333333333333333   0.250000000000000 ···
   0.500000000000000   0.333333333333333   0.250000000000000   0.200000000000000
   0.333333333333333   0.250000000000000   0.200000000000000   0.166666666666667
   0.250000000000000   0.200000000000000   0.166666666666667   0.142857142857143
   0.200000000000000   0.166666666666667   0.142857142857143   0.125000000000000
   0.166666666666667   0.142857142857143   0.125000000000000   0.111111111111111
   0.142857142857143   0.125000000000000   0.111111111111111   0.100000000000000
```

```
b=fix(10*rand(size(A,1),1))
```

```
b = 7×1
     0
     0
     5
     7
     9
     1
     5
```

```
[C,N]=solvesys(A,b);
```

```
Solutions obtained by different methods are the columns of
C = 1×3
```
10^8 ×
```
   0.002961420010407  -0.120743280511942   1.181073600000000
Norms of differences between solutions are the entires of
N = 3×1
```
10^8 ×
```
   0.123704700522349
   1.301816880511942
   1.178112179989593
```

% The results of **x1, x2, x3** are identical to **b1, b2** and **b3,** respectively. Implying there is a 1 to 1 relationship between A and B.

% For parts D and F, methods 1 and 2 do not produce results that are "close" to the results of method 3. This is likely due to the inaccuracies of the **rref()** function, given it is used to help students learn Linear Algebra and is strongly advised against for real life computation.

Part 2

```
disp('Computing condition number...')
```

```
Computing condition number...
```

```
c1=cond(magic(7))
```

```
c1 =
   7.111323446624705
```

```
c2=cond(hilb(7))
```

```
c2 =
     4.753673563768867e+08
```

% The condition number for c1 > c2 > 1. The condition number is the upper bound for error in computing the result. Thus, the condition number has a direct relationship with the norm of the differences. This being said, it seems that part(e) happened to compute the result more accurately than part(f), in my opinion, this is due to chance. The most important thing to remember is that the norm of the differences is bound below the condition number.

```
A=hilb(7);
b=ones(7,1);
x=A\b;
b1=b+0.01;
y=A\b1;
disp('Exploring sensitivity of pertubations of a badly condition matrix hilb(7)')
```

```
Exploring sensitivity of pertubations of a badly condition matrix hilb(7)
```

```
norm(x-y)
```

```
ans =
     5.242180560287886e+02
```

```
c3=rcond(A)
```

```
c3 =
     1.015027595488996e-09
```

```

A=magic(7);
b=ones(7,1);
x=A\b;
b1=b+0.01;
y=A\b1;
disp('Exploring sensitivity of pertubations of a badly condition matrix magic(7)')
```

```
Exploring sensitivity of pertubations of a badly condition matrix magic(7)
```

```
norm(x-y)
```

```
ans =
     1.511857892036916e-04
```

```
c3=rcond(A)
```

```
c3 =
   0.116711903838697
```

% The difference of the norms between the **hilb** matrix computations is much greater than that of the **magic** matrix, this supports the earlier statements made. This point is emphasised by the **rcond** number returned which shows the **magic** matrices condition number is far closer to 1 than **hilb's** matrix condition number (which is closer to 0) - implying **magic** is more well-conditioned.

Part 4. Area, Volume and Graphics in MATLAB

Exercise 4

```
format
type areavol
```

```
function D=areavol(A)
%First determining it's rank
rank(A);
%Determines whether the vector is within R^2 or R^3 determining if it's a parallelogram or not
if (size(A,2) == 2)
    isParallelogram = 1;
else
    isParallelogram = 0;
end
%condition checking whether it will be built or not
if (size(A,1) > rank(A))
    if (isParallelogram == 1)
        %from the earlier condition determing if the matrix is in R^2
        disp('parallelogram cannot be built.')
    else
        disp('parallelipiped cannot be built.')
    end
    %Displays an empty output
    D = 0
    return;
else
    %does formula for the volume of parallelipiped or area of parallelogram
    D = abs(det(A));
    %finally displays it
    if (isParallelogram == 1)
        disp(['The area of the parallelogram is ', num2str(D)])
    else
        disp(['The volume of the parallelipiped is ', num2str(D)])
    end
end
end
```

(a)

```
A = randi(10,2)
```

```
A = 2×2
     7     2
     7     2
```

```
areavol(A);
```

```
parallelogram cannot be built.
D = 0
```

(b)

```
A=fix(10*rand(3))
```

```
A = 3×3
     4     5     2
     9     2     5
     3     7     6
```

```
areavol(A);
```

```
The volume of the parallelipiped is 173
```

(c)

```
A = magic(3)
```

```
A = 3×3
     8     1     6
     3     5     7
     4     9     2
```

```
areavol(A);
```

```
The volume of the parallelipiped is 360
```

(d)

```
B = randi([-10,10],2,1);
A = [B,3*B]
```

```
A = 2×2
     8    24
    10    30
```

```
areavol(A);
```

```
parallelogram cannot be built.
D = 0
```

(e)

```
X = randi([-10,10],3,1);
Y=randi([-10,10],3,1);
A=[X,Y,X-Y]
```

```
A = 3×3
     1    -5     6
```

```
    -8     7   -15
    -7    -5    -2
```

```
areavol(A);
```

```
parallelipiped cannot be built.
D = 0
```

Exercise 5

```
R1 = [1,0;0,-1]
```

```
R1 = 2×2
     1     0
     0    -1
```

```
R2 = [-1,0;0,1]
```

```
R2 = 2×2
    -1     0
     0     1
```

```
VS = [1,0;2,1]
```

```
VS = 2×2
     1     0
     2     1
```

```
type transf
```

```
function C=transf(A,E)
C=A*E;
x=C(1,:);
y=C(2,:);
plot(x,y)
v=[-5 5 -5 5];
axis(v)
end
```

Function Analysis

Line 2 using matrix multiplication to create matrix c of the matrices passsed in. **Line 3** creates x as a vector equal to the first column of c, and y as the second column of c. **Line 4** plots x and y on a graph. **Line 5** cretaes a vector of 5s used to change the scaling. **Line 6** uses axis to alter the scaling of the graph by v

```
E=[0 1 1 0 0;0 0 1 1 0];
A=eye(2);
hold on
grid on
E=transf(A,E)
```

```
E = 2×5
     0     1     1     0     0
     0     0     1     1     0
```

```
A=VS;
E=transf(A,E)
```

```
E = 2×5
     0     1     1     0     0
     0     2     3     1     0
```

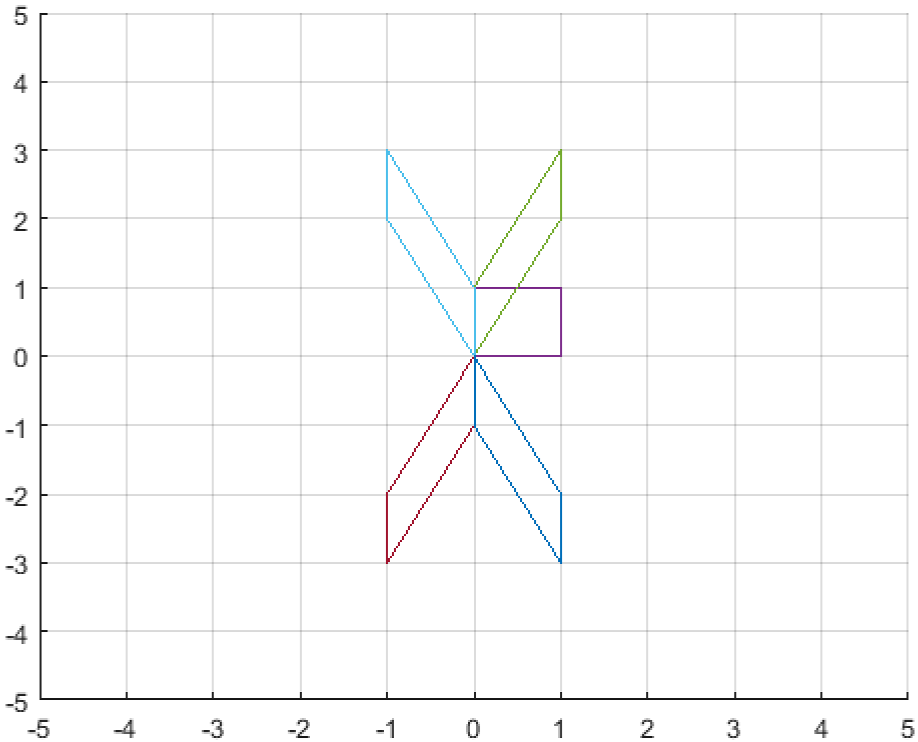
```
A=R2;
E=transf(A,E)
```

```
E = 2×5
     0    -1    -1     0     0
     0     2     3     1     0
```

```
A=R1;
E=transf(A,E)
```

```
E = 2×5
     0    -1    -1     0     0
     0    -2    -3    -1     0
```

```
A=R2;
E=transf(A,E)
```




```
E = 2×5
     0     1     1     0     0
     0    -2    -3    -1     0
```

Part 5. Cofactors, Determinant and Inverse Matrices

Exercise 6

Part 1

```
type closetozeroroundoff
```

```
function B = closetozeroroundoff(A,p)
A(abs(A)<10^-p)=0;
B=A;
end
```

```
type cofactor
```

```
function C=cofactor(a)
format compact
n = size(a,1);
C = zeros(n);
for i=1:n
    for j=1:n
        b = a;
        b(i,:) = [];
        b(:,j) = [];
        C(i, j) = (-1)^(i+j)*det(b);
    end
end
disp('the matrix of cofactors is')
end
```

Part 2

```
type determine
```

```
function D=determine(a,C)
D=[];
n=size(a,1);
if (rank(a)~=n)
    disp('the determinant of the matrix is')
    D=0;
    return
end
E=zeros(n,2);
for i=1:n
    r = 0;
    c = 0;
    for j=1:n
        r = r+a(i, j)*C(i,j);
        c = c+a(j, i)*C(j,i);
    end
    E(i, 1) = r;
    E(i, 2) = c;
end
for i=1:n
    for j=1:2
        if (closetozeroroundoff(E(1,1) - E(i,j), 7)~=0)
            disp('Something went wrong!')
            return
        end
    end
end
d=det(a);
if (closetozeroroundoff(d-E(1,1), 7)==0)
    disp('the determinant is')
    D = E(1,1);
else
    disp('Check the code!')
```

```
end
end
```

Part 3

```
type inverse
```

```
function B=inverse(a,C,D)
B=[];
if (isempty(D) || D==0)
    disp('a is not invertible')
    return
end
disp('a is invertible')
B = transpose(C)/D;F=inv(a);
if (closetozeroroundoff(B-F, 7)==0)
    disp('the inverse is calculated correctly and it is the matrix')
else
    disp('Something went wrong!')
end
end
```

Part 4

(a)

```
a=diag([1,2,3,4])
```

```
a = 4×4
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
```

```
C=cofactor(a)
```

```
the matrix of cofactors is
C = 4×4
    24     0     0     0
     0    12     0     0
     0     0     8     0
     0     0     0     6
```

```
D=determine(a,C)
```

```
the determinant is
D = 24
```

```
B=inverse(a,C,D)
```

```
a is invertible
the inverse is calculated correctly and it is the matrix
B = 4×4
    1.0000         0         0         0
         0    0.5000         0         0
         0         0    0.3333         0
         0         0         0    0.2500
```

(b)

```
a=ones(4)
```

```
a = 4×4
     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

```
C=cofactor(a)
```

```
the matrix of cofactors is
C = 4×4
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

```
D=determine(a,C)
```

```
the determinant of the matrix is
D = 0
```

```
B=inverse(a,C,D)
```

```
a is not invertible
B =
     []
```

(c)

```
a=magic(5)
```

```
a = 5×5
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

```
C=cofactor(a)
```

```
the matrix of cofactors is
C = 5×5
   -0.2503    2.1873   -1.5340    0.2373    0.1397
    2.5935   -1.8915    0.1560   -0.3315    0.2535
   -1.7940   -0.2340    0.1560    0.5460    2.1060
    0.0585    0.6435    0.1560    2.2035   -2.2815
    0.1723    0.0747    1.8460   -1.8753    0.5622
```

```
D=determine(a,C)
```

```
the determinant is
D = 5.0700e+06
```

```
B=inverse(a,C,D)
```

```
a is invertible
the inverse is calculated correctly and it is the matrix
B = 5×5
   -0.0049    0.0512   -0.0354    0.0012    0.0034
    0.0431   -0.0373   -0.0046    0.0127    0.0015
   -0.0303    0.0031    0.0031    0.0031    0.0364
    0.0047   -0.0065    0.0108    0.0435   -0.0370
```

```
    0.0028    0.0050    0.0415   -0.0450    0.0111
```

(d)

```
a=magic(6)
```

```
a = 6×6
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
```

```
C=cofactor(a)
```

```
the matrix of cofactors is
C = 6×6
    2.5894    2.5894   -1.2947   -2.5894   -2.5894    1.2947
   -0.0000    0.0000   -0.0000         0   -0.0000   -0.0000
   -2.5894   -2.5894    1.2947    2.5894    2.5894   -1.2947
   -2.5894   -2.5894    1.2947    2.5894    2.5894   -1.2947
    0.0000    0.0000   -0.0000    0.0000   -0.0000   -0.0000
    2.5894    2.5894   -1.2947   -2.5894   -2.5894    1.2947
```

```
D=determine(a,C)
```

```
the determinant of the matrix is
D = 0
```

```
B=inverse(a,C,D)
```

```
a is not invertible
B =
     []
```

(e)

```
a=hilb(4)
```

```
a = 4×4
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
```

```
C=cofactor(a)
```

```
the matrix of cofactors is
C = 4×4
    0.0000   -0.0000    0.0000   -0.0000
   -0.0000    0.0002   -0.0004    0.0003
    0.0000   -0.0004    0.0011   -0.0007
   -0.0000    0.0003   -0.0007    0.0005
```

```
D=determine(a,C)
```

```
the determinant is
D = 1.6534e-07
```

```
B=inverse(a,C,D)
```

```
a is invertible
the inverse is calculated correctly and it is the matrix
B = 4×4
    0.0160   -0.1200    0.2400   -0.1400
   -0.1200    1.2000   -2.7000    1.6800
    0.2400   -2.7000    6.4800   -4.2000
   -0.1400    1.6800   -4.2000    2.8000
```