MATLAB PROJECT 1

Please include this page in your Group file, as a front page. Type in the group number and the names of all members WHO PARTICIPATED in this project.

GROUP #   26

FIRST & LAST NAMES (UFID numbers are NOT required):

_1 Anthony Khmarin
_2 Ryan Houston
_3. Hao Lin
_4 Sean Madsen
_5 William Liu
_6 James Luberisse

By including your names above, each of you had confirmed that you did the work and agree with the work submitted.

Exercise 1

(1)

```
J = 3*eye(6)+diag(ones(5,1),1)
```

```
J = 6×6
     3     1     0     0     0     0
     0     3     1     0     0     0
     0     0     3     1     0     0
     0     0     0     3     1     0
     0     0     0     0     3     1
     0     0     0     0     0     3
```

(2)

```
A = toeplitz(mod(1:5,2))
```

```
A = 5×5
     1     0     1     0     1
     0     1     0     1     0
     1     0     1     0     1
     0     1     0     1     0
     1     0     1     0     1
```

(3)

```
A = randi([1, 10], [4,4])
```

```
A = 4×4
     4     2     6     3
     2     6     3     9
     5     3     3    10
     5     4     7     8
```

```
B = [A;mean(A)]
```

```
B = 5×4
    4.0000    2.0000    6.0000    3.0000
    2.0000    6.0000    3.0000    9.0000
    5.0000    3.0000    3.0000   10.0000
    5.0000    4.0000    7.0000    8.0000
    4.0000    3.7500    4.7500    7.5000
```

```
C = [A;sum(A)]
```

```
C = 5×4
     4     2     6     3
     2     6     3     9
     5     3     3    10
     5     4     7     8
    16    15    19    30
```

```
D = [A sum(A,2)]
```

```
D = 4×5
     4     2     6     3    15
     2     6     3     9    20
     5     3     3    10    21
     5     4     7     8    24
```

(4)

```
tril(triu(rand(6),-1),1)
```

```
ans = 6×6
    0.3439    0.2607         0         0         0         0
    0.5841    0.5944    0.4229         0         0         0
         0    0.0225    0.0942    0.0336         0         0
         0         0    0.5985    0.0688    0.7184         0
         0         0         0    0.3196    0.9686    0.4235
         0         0         0         0    0.5313    0.0908
```

Exercise 2

```
format
type format
```

```
'format' is a built-in function.
```

```
type jordan
```

```
function [V,J] = jordan(A)
%JORDAN Jordan Canonical Form.
%   JORDAN(A) computes the Jordan Canonical/Normal Form of the matrix A.
%   The matrix must be known exactly, so its elements must be integers
%   or ratios of small integers.  Any errors in the input matrix may
%   completely change its JCF.
%
%   [V,J] = JORDAN(A) also computes the similarity transformation, V, so
%   that V\A*V = J.  The columns of V are the generalized eigenvectors.
%
%   Example:
%      A = gallery(5);
%      [V,J] = jordan(A)
%
%   See also CHARPOLY, SYM/EIG, EIG, POLY

%   Copyright 1993-2014 The MathWorks, Inc.

oldDigits = digits(16);
cleanupObj = onCleanup(@() digits(oldDigits));

if nargout < 2
   V = cast(jordan(sym(A)),'like',A);
else
   [V,J] = jordan(sym(A));
   V = cast(V,'like',A);
   J = cast(J,'like',A);
end
```

```
%(a)
n=0; r=2;
J=jord(n,r);
```

```
n=0 is not valid input and Jordan Block cannot be built
```

```
%(b)
n=-2; r=3;
```

```
J=jord(n,r);
```

```
n=-2 is not valid input and Jordan Block cannot be built
```

```
%(c)
n=3.5; r=rand(1)
```

```
r = 0.2665
```

```
J=jord(n,r);
```

```
n=3.500000e+00 is not valid input and Jordan Block cannot be built
```

```
%(d)
n=2; r=4;
J=jord(n,r);
```

```
Jordan matrix of the size 2 by 2 is
J = 2×2
     4     1
     0     4
```

```
%(e)
n=4;r=randi(10,1)
```

```
r = 2
```

```
J=jord(n,r);
```

```
Jordan matrix of the size 4 by 4 is
J = 4×4
     2     1     0     0
     0     2     1     0
     0     0     2     1
     0     0     0     2
```

```
%(f)
n=1;r=rand(1)
```

```
r = 0.2810
```

```
J=jord(n,r);
```

```
n=1 is not valid input and Jordan Block cannot be built
```

Exercise 3

```
type span
```

```
function c=span(A,b)
format
[m,n]=size(A);
fprintf('matrix A has %i rows\n',m)
c=[];
if length(b)~=m
    fprintf('the dimensions mismatch: vector b is not in R^%i',m)
    return;
end
if length(b)==m
    fprintf('the dimensions match: vector b is in R^%i\n',m)
```

```
end
if rank(A)~=rank([A b])
    fprintf('the vector b is not in the span of columns of A ');
    return;
end
if rank(A)==rank([A b])
    if rank(A)==m
        fprintf('the columns of A span the whole R^%i\n',m)
    else
        fprintf('the columns of A do not span the whole R^m, but vector b is in the Span of the columns of A')
    end
end
c=A\b;
if closetozeroroundoff(A*c-b,7)~=0
    fprintf('check the code!')
    return;
end
if closetozeroroundoff(A*c-b,7)==0
    if m==n
        disp('the unique vector of weights is')
    else
        disp('one of the vectors of weights is')
    end
end
c=closetozeroroundoff(c,7)
end
```

```
type closetozeroroundoff
```

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

```
%(a)
A=magic(5), b=ones(5,1)
```

```
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
b = 5×1
     1
     1
     1
     1
     1
```

```
c=span(A,b);
```

```
matrix A has 5 rows
the dimensions match: vector b is in R^5
the columns of A span the whole R^5
the unique vector of weights is
c = 5×1
    0.0154
    0.0154
    0.0154
    0.0154
    0.0154
```

```
%(b)
A=magic(4), b=ones(5,1)
```

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

```
c=span(A,b);
```

```
matrix A has 4 rows
the dimensions mismatch: vector b is not in R^4
```

```
%(c)
A=randi(10,5,4), b=ones(5,1)
```

```
A = 5×4
     5    10     3     3
     6     7     7     7
     5    10     7     9
     9     3     1     4
     6     7     3     8
b = 5×1
     1
     1
     1
     1
     1
```

```
c=span(A,b);
```

```
matrix A has 5 rows
the dimensions match: vector b is in R^5
the vector b is not in the span of columns of A
```

```
%(d)
A=randi(10,5,4), b=A( : , end)
```

```
A = 5×4
     7     1     4     8
     1     5     8     5
     7     5     5     2
     4     5     1     4
    10     8     2     7
b = 5×1
     8
     5
     2
     4
     7
```

```
c=span(A,b);
```

```
matrix A has 5 rows
the dimensions match: vector b is in R^5
```

```
the columns of A do not span the whole R^m, but vector b is in the Span of the columns of Aone of the vectors of wei
c = 4×1
     0
     0
     0
     1
```

```
%(e)
A=randi(10,5,6), b=A( : , end)
```

```
A = 5×6
     2     8     7     7     3     7
     8     2     6     7     2     8
     3     3     5    10     7     4
    10     1     7     3     5     7
     3     6     7     8     5     5
b = 5×1
     7
     8
     4
     7
     5
```

```
c=span(A,b);
```

```
matrix A has 5 rows
the dimensions match: vector b is in R^5
the columns of A span the whole R^5
one of the vectors of weights is
c = 6×1
     0
     0
     0
     0
     0
     1
```

```
%(f)
A=randi(10,5,5), b=zeros(5,1)
```

```
A = 5×5
     9     6    10     7     3
     9     9     7     6     2
     3     3     5     8     2
     7     4     7     6     1
     6     2     6    10     5
b = 5×1
     0
     0
     0
     0
     0
```

```
c=span(A,b);
```

```
matrix A has 5 rows
the dimensions match: vector b is in R^5
the columns of A span the whole R^5
the unique vector of weights is
c = 5×1
     0
     0
     0
```

```
     0
     0
```

```
%(g)
A=randi(10,4,5), b=zeros(4,1)
```

```
A = 4×5
     5     8     2     6     7
     4    10     7     9     8
     8    10     1     5     6
     7     2     6     4     4
b = 4×1
     0
     0
     0
     0
```

```
c=span(A,b);
```

```
matrix A has 4 rows
the dimensions match: vector b is in R^4
the columns of A span the whole R^4
one of the vectors of weights is
c = 5×1
     0
     0
     0
     0
     0
```

Exercise 4

```
type crossdot
```

```
function [] = crossdot(a,b,c)

NormCrossAB = norm(cross(a,b));   %Check to see if AB Parallel
NormCrossAC = norm(cross(a,c));   %Check to see if AC Parallel

if(NormCrossAB ~= 0)
    disp('we build a parallelogram on vectors a and b')
    %1 Area of Parallelogram
    AreaAB = norm((cross(a,b)))
    %2 Height of Parallelogram
    HeightAB = norm(AreaAB)/norm(b)
    %3 Angle between A and B
    angleAB = acosd( abs(dot(a,b)) / (norm(a).*norm(b)))
    %4 Orthogonal Projection of vector b onto vector a
    orthogonalProjectionAB = (dot(b,a) / dot(a,a)) * a
elseif(NormCrossAC ~= 0)
    disp('we build a parallelogram on vectors a and c')
    %1 Area of Parallelogram
    AreaAC = norm((cross(a,c)))
    %2 Height of Parallelogram
    HeightAC = norm(AreaAC)/norm(c)
    %3 Angle between A and B
    angleAC = acosd( abs(dot(a,c)) / (norm(a).*norm(c)))
    %4 Orthogonal Projection of vector b onto vector a
    orthogonalProjectionAC = (dot(c,a) / dot(a,a)) * a
else
    disp('a parallelogram cannot be built using vectors a,b,c')
    return
end
```

```
%coplanar if ScalarTripleProduct == 0
ScalarTripleProduct = dot(c,cross(a,b));

if(ScalarTripleProduct ~= 0)
    %5 Volume
    VolumeOfParallelepiped = abs(ScalarTripleProduct)
    %6 Distance from c to the plane spanned by a and b
    DistanceFromCtoPlaneAB = VolumeOfParallelepiped/norm(cross(a,b))
else
    disp('a,b,c are coplanar and parallelepiped cannot be built on them')

end


end
```

```
% (a)
a = [1;2;3], b = a, c = 2*a
```

```
a = 3×1
     1
     2
     3
b = 3×1
     1
     2
     3
c = 3×1
     2
     4
     6
```

```
crossdot(a,b,c)
```

```
a parallelogram cannot be built using vectors a,b,c
```

```
%(b)
a = [1;2;3], b = a, c = randi(10,3,1)
```

```
a = 3×1
     1
     2
     3
b = 3×1
     1
     2
     3
c = 3×1
     2
     6
     3
```

```
crossdot(a,b,c)
```

```
we build a parallelogram on vectors a and c
AreaAC = 12.5300
HeightAC = 1.7900
angleAC = 28.5807
orthogonalProjectionAC = 3×1
    1.6429
    3.2857
```

```
    4.9286
a,b,c are coplanar and parallelepiped cannot be built on them
```

```
%(c)
a=[1;2;0], b = [2;0;1], c = -b
```

```
a = 3×1
     1
     2
     0
b = 3×1
     2
     0
     1
c = 3×1
    -2
     0
    -1
```

```
crossdot(a,b,c)
```

```
we build a parallelogram on vectors a and b
AreaAB = 4.5826
HeightAB = 2.0494
angleAB = 66.4218
orthogonalProjectionAB = 3×1
    0.4000
    0.8000
         0
a,b,c are coplanar and parallelepiped cannot be built on them
```

```
%(d)
E=eye(3); a=E(:,1), b = 2*E(:,2), c = 3*E(:,3)
```

```
a = 3×1
     1
     0
     0
b = 3×1
     0
     2
     0
c = 3×1
     0
     0
     3
```

```
crossdot(a,b,c)
```

```
we build a parallelogram on vectors a and b
AreaAB = 2
HeightAB = 1
angleAB = 90
orthogonalProjectionAB = 3×1
     0
     0
     0
VolumeOfParallelepiped = 6
DistanceFromCtoPlaneAB = 3
```

```
%(e)
a = [1;0;-3], b = [-2;1;-1], c=ones(3,1)
```

```
a = 3×1
     1
     0
    -3
b = 3×1
    -2
     1
    -1
c = 3×1
     1
     1
     1
```

```
crossdot(a,b,c)
```

```
we build a parallelogram on vectors a and b
AreaAB = 7.6811
HeightAB = 3.1358
angleAB = 82.5824
orthogonalProjectionAB = 3×1
    0.1000
         0
   -0.3000
VolumeOfParallelepiped = 11
DistanceFromCtoPlaneAB = 1.4321
```

```
%(f)
a=randi(10,3,1), b=randi(10,3,1), c = randi(10,3,1)
```

```
a = 3×1
     1
     8
     3
b = 3×1
     5
     7
     4
c = 3×1
     8
     4
     7
```

```
crossdot(a,b,c)
```

```
we build a parallelogram on vectors a and b
AreaAB = 36.4829
HeightAB = 3.8456
angleAB = 26.5543
orthogonalProjectionAB = 3×1
    0.9865
    7.8919
    2.9595
VolumeOfParallelepiped = 99
DistanceFromCtoPlaneAB = 2.7136
```

Exercise 5

```
type closetozeroroundoff
```

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

```
B=A;
end
```

```
type homobasis
```

```
function C = homobasis(A)
format
[m,n]=size(A);
red_ech_form = rats(rref(A));
C=[];
if rank(A) == n
    disp('the homogeneous system has only the trivial solution')
    C = zeros(n,1);
    return;
else
    disp('the homogeneous system has non-trivial solutions')
    [~,pivot_c] = rref(A);
    S=1:n;
    nonpivot_c = setdiff(S,pivot_c);
    q=numel(nonpivot_c);
    j=1:q;
    fprintf('a free variable is x%i\n',nonpivot_c(j))
    C=zeros(n,q);
    Aref = rref(A);
    colOfZero = 0;
    for k = 1:n
        if Aref(:,k) == zeros(m,1)
            colOfZero = 1 + colOfZero;
        else
            break;
        end
    end
    C(1+colOfZero:m+colOfZero,j)= -Aref(1:m,nonpivot_c(j));
    for i = 1:q
     C(nonpivot_c(i),i) = 1;
    end
    if isequal(rank(C),q) && isequal(closetozeroroundoff(A*C,5),zeros(m,q))
        disp('columns of C form a basis for solution set of homogeneous system')
        C
    else
        disp('Alert, bug detected')
        C = [];
    end
end
```

```
%(a)
A=[1 -1 -1 2;-2 5 4 4]
```

```
A = 2×4
     1    -1    -1     2
    -2     5     4     4
```

```
C=homobasis(A);
```

```
the homogeneous system has non-trivial solutions
a free variable is x3
a free variable is x4
columns of C form a basis for solution set of homogeneous system
C = 4×2
    0.3333   -4.6667
   -0.6667   -2.6667
    1.0000         0
         0    1.0000
```

```
%(b)
A=[1 2 -3]
```

```
A = 1×3
     1     2    -3
```

```
C=homobasis(A);
```

```
the homogeneous system has non-trivial solutions
a free variable is x2
a free variable is x3
columns of C form a basis for solution set of homogeneous system
C = 3×2
    -2     3
     1     0
     0     1
```

```
%(c)
A=magic(3)
```

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

```
C=homobasis(A);
```

```
the homogeneous system has only the trivial solution
```

```
%(d)
A=[magic(3), ones(3,1)]
```

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

```
C=homobasis(A);
```

```
the homogeneous system has non-trivial solutions
a free variable is x4
columns of C form a basis for solution set of homogeneous system
C = 4×1
   -0.0667
   -0.0667
   -0.0667
    1.0000
```

```
%(e)
A=magic(4)
```

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

```
C=homobasis(A);
```

```
the homogeneous system has non-trivial solutions
a free variable is x4
```

```
columns of C form a basis for solution set of homogeneous system
C = 4×1
    -1
    -3
     3
     1
```

```
%(f)
A=[0 1 2 3;0 2 4 6]
```

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

```
C=homobasis(A);
```

```
the homogeneous system has non-trivial solutions
a free variable is x1
a free variable is x3
a free variable is x4
columns of C form a basis for solution set of homogeneous system
C = 4×3
     1     0     0
     0    -2    -3
     0     1     0
     0     0     1
```

```
%(g)
A=[0 1 0 2 0 3; 0 2 0 4 0 6; 0 4 0 8 0 6]
```

```
A = 3×6
     0     1     0     2     0     3
     0     2     0     4     0     6
     0     4     0     8     0     6
```

```
C=homobasis(A);
```

```
the homogeneous system has non-trivial solutions
a free variable is x1
a free variable is x3
a free variable is x4
a free variable is x5
columns of C form a basis for solution set of homogeneous system
C = 6×4
     1     0     0     0
     0     0    -2     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
     0     0     0     0
```

```
%(h)
A=[1 0 2 0 3;2 0 5 0 6]
```

```
A = 2×5
     1     0     2     0     3
     2     0     5     0     6
```

```
C=homobasis(A);
```

```
the homogeneous system has non-trivial solutions
a free variable is x2
```

```
a free variable is x4
a free variable is x5
columns of C form a basis for solution set of homogeneous system
C = 5×3
     0     0    -3
     1     0     0
     0     0     0
     0     1     0
     0     0     1
```

```
%(k)
A=[1 0 0 2 3;2 0 0 4 6]
```

```
A = 2×5
     1     0     0     2     3
     2     0     0     4     6
```

```
C=homobasis(A);
```

```
the homogeneous system has non-trivial solutions
a free variable is x2
a free variable is x3
a free variable is x4
a free variable is x5
columns of C form a basis for solution set of homogeneous system
C = 5×4
     0     0    -2    -3
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
%(l)
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=homobasis(A);
```

```
the homogeneous system has only the trivial solution
```

```
%}
```

Exercise 6

```
type stochastic.m
```

```
function [S1, S2, L, R] = stochastic(A)
L = [];
R = [];
%**First, your function has to check whether A contains both a zero column and a zero row. If
% yes, output a message “A is neither left nor right stochastic and cannot be scaled to either of
% them”. The empty outputs assigned previously to L and R will stay.
columns = sum(A);
nOfCol = numel(columns);

rows = sum(A, 2);
nOfRow = numel(rows);
```

```
zeroInCol = false;
zeroInRow = false;
Continue = true;

% for loop checks if zero is in column or row.
for c = 1:nOfCol
    if columns(c) == 0
        zeroInCol = true;
    end
end

for c = 1:nOfCol
    if rows(c) == 0
        zeroInRow = true;
    end
end

if zeroInRow && zeroInCol
    fprintf('A is neither left nor right stochastic and cannot be scaled to either of them');
    Continue = false;
end

if Continue == true
    fprintf('the vector of sums down each column is\n');
    S1 = sum(A)

    fprintf('the vector of sums across each row is\n');
    S2 = sum(A, 2)


    Neither = false;
    rightStoch = true;
    leftStoch = true;

    sumCol = sum(A, 1);
    sumRow = sum(A, 2);
    for c = 1:nOfCol
        if sumCol(c) ~= 1

            leftStoch = false;
        end
    end
    for c = 1:nOfCol
        if sumRow(c) ~= 1
            rightStoch = false;
        end
    end

    if leftStoch == true & rightStoch == true
        L = A;
        R = A;
        disp('A is doubly stochastic')
    elseif rightStoch == true & leftStoch == false
        R = A;
        disp('A is only right stochastic')

    elseif rightStoch == false & leftStoch == true
        L = A;
        disp('A is only left stochastic')
    else
        Neither = true;
    end

    %logic for neither
```

```
if Neither == true
    zeroInCol = false;
    zeroInRow = false;
    for c = 1:nOfCol
        if S2(c) == 0
            zeroInCol = true;
        end
    end

    for c = 1:nOfRow
        if S1(c) == 0
            zeroInRow = true;
        end
    end

    if zeroInRow == false && zeroInCol == false
        L = A;
        R = A;
        if closetozeroroundoff(L, 7) == closetozeroroundoff(R, 7)
            %scale L
            for n = 1:size(S1, 2)
                S1(n) = 1 / S1(n);
            end
            S1 = repmat(S1, size(A, 2), 1);
            L = A .* S1;

            disp('A has been scaled to a doubly stochastic matrix:');

            disp(L);
        else
            %scale L
            disp('A is scaled to a left stochastic matrix:')

            for n = 1:size(S1, 2)
                S1(n) = 1 / S1(n);
            end
            S1 = repmat(S1, size(A, 2), 1);
            L = A .* S1;
            disp(L);

            %scale R
            disp('and A is scaled to a right stochastic matrix:')
            for n = 1:size(S2, 1)
                S2(n) = 1 / S2(n);
            end
            S2 = reshape(S2',size(A, 2), []);
            S2 = repmat(S2' ,size(A,1), 1);

            R = A .* S2;
            disp(R);
        end
    end
    if zeroInRow == false && zeroInCol == true
        for n = 1:size(S1, 2)
            S1(n) = 1 / S1(n);
        end
        S1 = repmat(S1, size(A, 2), 1);
        L = A .* S1;
        disp('A is not stochastic but can be scaled to left stochastic only:');
        disp(L);
    end
    if zeroInRow == true && zeroInCol == false

        for n = 1:size(S2, 1)
            S2(n) = 1 / S2(n);
```

```
            end
            S2 = reshape(S2',size(A, 2), []);
            S2 = repmat(S2' ,size(A,1), 1);

            R = A .* S2;
            disp('A is not stochastic but can be scaled to right stochastic only:')
            disp(R);
        end

    end
end
end
```

```
type closetozeroroundoff.m
```

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

```
type jord.m
```

```
function J = jord(n,r)
if mod(n,1)==0 && n>1
    oneVector=r*ones(n,1);
    J=diag(oneVector);
    for i=[1:n-1]
    J(i,i+1)=1;
    end
fprintf('Jordan matrix of the size %i by %i is\n',n,n)
J
else
J=[];
fprintf('n=%i is not valid input and Jordan Block cannot be built\n',n)
end
end
```

```
%(a)
A=[0.5, 0, 0.5; 0, 0, 1; 0.5, 0, 0.5]
```

```
A = 3×3
    0.5000         0    0.5000
         0         0    1.0000
    0.5000         0    0.5000
```

```
stochastic(A);
```

```
the vector of sums down each column is
S1 = 1×3
     1     0     2
the vector of sums across each row is
S2 = 3×1
     1
     1
     1
A is only right stochastic
```

```
%(b)
A = transpose(A)
```

```
A = 3×3
```

```
    0.5000         0    0.5000
         0         0         0
    0.5000    1.0000    0.5000
```

```
stochastic(A);
```

```
the vector of sums down each column is
S1 = 1×3
     1     1     1
the vector of sums across each row is
S2 = 3×1
     1
     0
     2
A is only left stochastic
```

```
%(c)
A=[0.5, 0, 0.5; 0, 0, 1; 0, 0, 0.5]
```

```
A = 3×3
    0.5000         0    0.5000
         0         0    1.0000
         0         0    0.5000
```

```
stochastic(A);
```

```
the vector of sums down each column is
S1 = 1×3
    0.5000         0    2.0000
the vector of sums across each row is
S2 = 3×1
    1.0000
    1.0000
    0.5000
A is not stochastic but can be scaled to right stochastic only:
    0.5000         0    1.0000
         0         0    2.0000
         0         0    1.0000
```

```
%(d)
A=transpose(A)
```

```
A = 3×3
    0.5000         0         0
         0         0         0
    0.5000    1.0000    0.5000
```

```
stochastic(A);
```

```
the vector of sums down each column is
S1 = 1×3
    1.0000    1.0000    0.5000
the vector of sums across each row is
S2 = 3×1
    0.5000
         0
    2.0000
A is not stochastic but can be scaled to left stochastic only:
    0.5000         0         0
         0         0         0
```

```
    0.5000    1.0000    1.0000
```

```
%(e)
A=[0.5, 0, 0.5; 0, 0.5, 0.5; 0.5, 0.5, 0]
```

```
A = 3×3
    0.5000         0    0.5000
         0    0.5000    0.5000
    0.5000    0.5000         0
```

```
stochastic(A);
```

```
the vector of sums down each column is
S1 = 1×3
     1     1     1
the vector of sums across each row is
S2 = 3×1
     1
     1
     1
A is doubly stochastic
```

```
%(f)
A=magic(4)
```

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

```
stochastic(A);
```

```
the vector of sums down each column is
S1 = 1×4
    34    34    34    34
the vector of sums across each row is
S2 = 4×1
    34
    34
    34
    34
A has been scaled to a doubly stochastic matrix:
    0.4706    0.0588    0.0882    0.3824
    0.1471    0.3235    0.2941    0.2353
    0.2647    0.2059    0.1765    0.3529
    0.1176    0.4118    0.4412    0.0294
```

```
%(g)
B=[1 2;3 4;5 6]; A=B*B'
```

```
A = 3×3
     5    11    17
    11    25    39
    17    39    61
```

```
stochastic(A);
```

```
the vector of sums down each column is
```

```
S1 = 1×3
    33    75   117
the vector of sums across each row is
S2 = 3×1
    33
    75
   117
A has been scaled to a doubly stochastic matrix:
    0.1515    0.1467    0.1453
    0.3333    0.3333    0.3333
    0.5152    0.5200    0.5214
```

```
%(h)
A=jord(4,3)
```

```
Jordan matrix of the size 4 by 4 is
J = 4×4
     3     1     0     0
     0     3     1     0
     0     0     3     1
     0     0     0     3
A = 4×4
     3     1     0     0
     0     3     1     0
     0     0     3     1
     0     0     0     3
```

```
stochastic(A);
```

```
the vector of sums down each column is
S1 = 1×4
     3     4     4     4
the vector of sums across each row is
S2 = 4×1
     4
     4
     4
     3
A has been scaled to a doubly stochastic matrix:
    1.0000    0.2500         0         0
         0    0.7500    0.2500         0
         0         0    0.7500    0.2500
         0         0         0    0.7500
```

```
%(k)
A=randi(10,4);A(:,1)=0;A(1,:)=0
```

```
A = 4×4
     0     0     0     0
     0     3     9     9
     0     2     4     8
     0     9     8     4
```

```
stochastic(A);
```

```
A is neither left nor right stochastic and cannot be scaled to either of them
```

Exercise 7

```
type economy
```

```
function [] = economy(n)
format
format compact

%left stochastic scaling
A = randi(10, n);
S1 = sum(A);
for n = 1:size(S1, 2)
    S1(n) = 1 / S1(n);
end

S1 = repmat(S1, size(A, 2), 1);
L = A .* S1;
Sector = L;

B = [L ones(size(L, 1), 1)];
T = array2table(Sector)
sizeOfColOfSect = size(B, 2);

% either add column of ones or no

%constructs B matrix
for c = 1:sizeOfColOfSect
    r = c;
    if r == sizeOfColOfSect
        r = r - 1;
    end
    B(r, c) = B(r, c) - B(r, sizeOfColOfSect);
    B(r, sizeOfColOfSect) = B(r, sizeOfColOfSect) - B(r, sizeOfColOfSect);
end

C = homobasis(B);

syms p
fprintf('vector of equilibrium prices with parameter p=x%i is\n', n)
x=p*C

end
```

```
type homobasis
```

```
function C = homobasis(A)
format
[m,n]=size(A);
red_ech_form = rats(rref(A));
C=[];
if rank(A) == n
    disp('the homogeneous system has only the trivial solution')
    C = zeros(n,1);
    return;
else
    disp('the homogeneous system has non-trivial solutions')
    [~,pivot_c] = rref(A);
    S=1:n;
    nonpivot_c = setdiff(S,pivot_c);
    q=numel(nonpivot_c);
    j=1:q;
    fprintf('a free variable is x%i\n',nonpivot_c(j))
    C=zeros(n,q);
    Aref = rref(A);
    colOfZero = 0;
```

```
    for k = 1:n
        if Aref(:,k) == zeros(m,1)
            colOfZero = 1 + colOfZero;
        else
            break;
        end
    end
    C(1+colOfZero:m+colOfZero,j)= -Aref(1:m,nonpivot_c(j));
    for i = 1:q
     C(nonpivot_c(i),i) = 1;
    end
    if isequal(rank(C),q) && isequal(closetozeroroundoff(A*C,5),zeros(m,q))
        disp('columns of C form a basis for solution set of homogeneous system')
        C
    else
        disp('Alert, bug detected')
        C = [];
    end
end
```

```
economy(4)
```

T = *4×4 table*

Sector1	Sector2	Sector3	Sector4
0.12	0.24138	0.074074	0.47368
0.32	0.17241	0.33333	0.31579
0.4	0.31034	0.37037	0.10526
0.16	0.27586	0.22222	0.10526

```
the homogeneous system has non-trivial solutions
a free variable is x4
a free variable is x5
columns of C form a basis for solution set of homogeneous system
C = 5×2
    1.0517         0
    1.4033         0
    1.5270         0
    1.0000         0
         0    1.0000
vector of equilibrium prices with parameter p=x4 is
x =
```

$$\begin{pmatrix} \frac{99675\,p}{94772} & 0 \\ \frac{2293\,p}{1634} & 0 \\ \frac{36180\,p}{23693} & 0 \\ p & 0 \\ 0 & p \end{pmatrix}$$

```
economy(6)
```

T = *6×6 table*

Sector1	Sector2	Sector3	Sector4	Sector5	Sector6
0.125	0.052632	0.15152	0.16667	0.19565	0.073171
0.2	0.10526	0.060606	0.14286	0.15217	0.12195
0.225	0.10526	0.030303	0.21429	0.21739	0.2439
0.2	0.36842	0.27273	0.21429	0.13043	0.14634
0.1	0.26316	0.18182	0.2381	0.1087	0.21951

```
      0.15      0.10526     0.30303    0.02381    0.19565     0.19512
the homogeneous system has non-trivial solutions
a free variable is x6
a free variable is x7
columns of C form a basis for solution set of homogeneous system
C = 7×2
    0.8418         0
    0.8138         0
    1.0966         0
    1.3565         0
    1.1671         0
    1.0000         0
         0    1.0000
vector of equilibrium prices with parameter p=x6 is
x =
```

$$\begin{pmatrix} \frac{7582324785573003\,p}{9007199254740992} & 0 \\ \frac{3665141855670163\,p}{4503599627370496} & 0 \\ \frac{4938640749108329\,p}{4503599627370496} & 0 \\ \frac{3054598387043795\,p}{2251799813685248} & 0 \\ \frac{5256035884557797\,p}{4503599627370496} & 0 \\ p & 0 \\ 0 & p \end{pmatrix}$$

```
economy(8)
```

T = *8×8 table*

Sector1	Sector2	Sector3	Sector4	Sector5	Sector6	Sector7	Sector8
0.125	0.19231	0.11905	0.055556	0.17073	0.10638	0.090909	0.017544
0.0625	0.15385	0.21429	0.27778	0.073171	0.19149	0.30303	0.17544
0.14583	0.11538	0.071429	0.11111	0.14634	0.19149	0.030303	0.14035
0.020833	0.19231	0.14286	0.11111	0.17073	0.06383	0.060606	0.14035
0.14583	0.11538	0.16667	0.16667	0.12195	0.06383	0.060606	0.017544
0.14583	0.019231	0.02381	0.11111	0.14634	0.12766	0.060606	0.15789
0.16667	0.038462	0.16667	0.11111	0.12195	0.14894	0.21212	0.17544
0.1875	0.17308	0.095238	0.055556	0.04878	0.10638	0.18182	0.17544

```
the homogeneous system has non-trivial solutions
a free variable is x8
a free variable is x9
columns of C form a basis for solution set of homogeneous system
C = 9×2
    0.8396         0
    1.3702         0
    0.8600         0
    0.8840         0
    0.7917         0
    0.6918         0
    1.0271         0
    1.0000         0
         0    1.0000
vector of equilibrium prices with parameter p=x8 is
x =
```

$$\begin{pmatrix} \frac{945343611119559\,p}{1125899906842624} & 0 \\ \frac{1542716199842227\,p}{1125899906842624} & 0 \\ \frac{484143116202143\,p}{562949953421312} & 0 \\ \frac{3981198990178315\,p}{4503599627370496} & 0 \\ \frac{7131159954297463\,p}{9007199254740992} & 0 \\ \frac{3115665224282957\,p}{4503599627370496} & 0 \\ \frac{2312803030551481\,p}{2251799813685248} & 0 \\ p & 0 \\ 0 & p \end{pmatrix}$$