MATLAB PROJECT 3

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 # 5

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

1. Corey Wolfe
2. Ekaterina Krysova
3. Dany Rashwan
4. Elton Li
5. Manuel Vera
6. Thomas Pena

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

Part I. Subspaces & Bases

Exercise 1

```
type columnspaces
```

```
function[] = columnspaces(A,B)
m = size (A,1);
n = size (B,1);

if m~=n
    disp('Col A and Col B are subspaces of different spaces');
    return
else
    fprintf('Col A and Col B are subspaces of R^%i\n',m);
end

ColA = rank(A);
ColB = rank(B);

dA = 'The dimension of ColA is %d\n';
dB = 'The dimension of ColB is %d\n';
fprintf(dA, ColA);
fprintf(dB, ColB);

if ColA == m && ColB == m
    fprintf('Col A = Col B = R^%i\n',m);
    return
end

if ColA ~= ColB
    disp('The dimensions of Col A and Col B are different.');
else
     A_red = rref(transpose(A));
     B_red = rref(transpose(B));
     ind = find(abs(sum(A_red,2))==0);
     A_red(ind,:) = [];
     ind = find(abs(sum(B_red,2))==0);
     B_red(ind,:) = [];
     if isequal(A_red, B_red)
        disp('Col A = Col B');
    else
        disp('The dimensions of Col A and Col B are the same but Col A ~= Col B');
    end
end

end
```

```
format
%(a)
A=[2 -4 -2 3;6 -9 -5 8;2 -7 -3 9;4 -2 -2 -1;-6 3 3 4]
```

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

```
B=rref(A)
```

```
B = 5×4
    1.0000         0   -0.3333         0
```

```
         0    1.0000    0.3333         0
         0         0         0    1.0000
         0         0         0         0
         0         0         0         0
```

```
columnspaces(A,B)
```

```
Col A and Col B are subspaces of R^5
The dimension of ColA is 3
The dimension of ColB is 3
The dimensions of Col A and Col B are the same but Col A ~= Col B
```

```
%(b)
A=[2 -4 -2 3;6 -9 -5 8;2 -7 -3 9;4 -2 -2 -1;-6 3 3 4];
B=([rref(A);zeros(5,4)])'
```

```
B = 4×10
    1.0000         0         0         0         0         0         0         0···
         0    1.0000         0         0         0         0         0         0
   -0.3333    0.3333         0         0         0         0         0         0
         0         0    1.0000         0         0         0         0         0
```

```
A=A'
```

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

```
columnspaces(A,B)
```

```
Col A and Col B are subspaces of R^4
The dimension of ColA is 3
The dimension of ColB is 3
Col A = Col 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
```

```
B=ones(5)
```

```
B = 5×5
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

```
columnspaces(A,B)
```

```
Col A and Col B are subspaces of R^5
The dimension of ColA is 5
The dimension of ColB is 1
The dimensions of Col A and Col B are different.
```

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

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

```
B=eye(4)
```

```
B = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
columnspaces(A,B)
```

```
Col A and Col B are subspaces of R^4
The dimension of ColA is 3
The dimension of ColB is 4
The dimensions of Col A and Col B are different.
```

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

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

```
B=[eye(3);zeros(1,3)]
```

```
B = 4×3
     1     0     0
     0     1     0
     0     0     1
     0     0     0
```

```
columnspaces(A,B)
```

```
Col A and Col B are subspaces of R^4
The dimension of ColA is 3
The dimension of ColB is 3
The dimensions of Col A and Col B are the same but Col A ~= Col B
```

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

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

```
B=[hilb(3),eye(3)]
```

```
B = 3×6
    1.0000    0.5000    0.3333    1.0000         0         0
    0.5000    0.3333    0.2500         0    1.0000         0
```

```
    0.3333    0.2500    0.2000         0         0    1.0000
```

```
columnspaces(A,B)
```

```
Col A and Col B are subspaces of R^3
The dimension of ColA is 3
The dimension of ColB is 3
Col A = Col B = R^3
```

```
%{
Elementary row operations preserve the row space of a matrix but not the column space.
So if two matrices are row equivalent their row spaces are equal but their column
spaces are not necessarily equal. This can be seen in part (a), where B is the reduced
row echelon form of A and yet the column space of B is different from the column
space of A.
%}
```

Exercise 2

Part 1

```
type shrink
```

```
function B=shrink(A)
[~,pivot]=rref(A);
B=A(:,pivot);
end
```

```
A=magic(4); A(:,3)=A(:,2)
```

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

```
rref(A)
```

```
ans = 4×4
     1     0     0     0
     0     1     1     0
     0     0     0     1
     0     0     0     0
```

```
[R,pivot]=rref(A)
```

```
R = 4×4
     1     0     0     0
     0     1     1     0
     0     0     0     1
     0     0     0     0
pivot = 1×3
     1     2     4
```

```
B=A(:,pivot)
```

```
B = 4×3
    16     2    13
     5    11     8
     9     7    12
```

```
     4    14     1
```

```
%R is the matrix which is the reduced row echelon form of A and pivot is the
%vector which holds the indexes of the pivot columns of A
%B is the matrix obtained by taking the pivot columns of A
[~,pivot]=rref(A)
```

```
pivot = 1×3
     1     2     4
```

```
%[~,pivot]=rref(A) stores the vector with indexes of the pivot columns of A
%as 'pivot', just like the command [R,pivot]=rref(A)
%However, the command [~,pivot] = rref(A) does not store the reduced row
% echelon form of A in any variable
A=[2 -4 -2 3;6 -9 -5 8;2 -7 -3 9;4 -2 -2 -1;-6 3 3 4]
```

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

```
B=shrink(A)
```

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

```
columnspaces(A,B)
```

```
Col A and Col B are subspaces of R^5
The dimension of ColA is 3
The dimension of ColB is 3
Col A = Col B
```

```
type columnspaces
```

```
function[] = columnspaces(A,B)
m = size (A,1);
n = size (B,1);

if m~=n
    disp('Col A and Col B are subspaces of different spaces');
    return
else
    fprintf('Col A and Col B are subspaces of R^%i\n',m);
end

ColA = rank(A);
ColB = rank(B);

dA = 'The dimension of ColA is %d\n';
dB = 'The dimension of ColB is %d\n';
fprintf(dA, ColA);
fprintf(dB, ColB);

if ColA == m && ColB == m
    fprintf('Col A = Col B = R^%i\n',m);
```

```
    return
end

if ColA ~= ColB
    disp('The dimensions of Col A and Col B are different.');
else
     A_red = rref(transpose(A));
     B_red = rref(transpose(B));
     ind = find(abs(sum(A_red,2))==0);
     A_red(ind,:) = [];
     ind = find(abs(sum(B_red,2))==0);
     B_red(ind,:) = [];
     if isequal(A_red, B_red)
        disp('Col A = Col B');
    else
        disp('The dimensions of Col A and Col B are the same but Col A ~= Col B');
    end
end

end
```

```
%The set of the columns of B forms a basis for the column space of A
%because the columns of B are all the pivot columns of A. So these columns
%are linearly independent and they span the column space of A, so they are
%a basis for the column space of A.
```

Part 2

```
A=[2 -4 -2 3;6 -9 -5 8;2 -7 -3 9;4 -2 -2 -1;-6 3 3 4]
```

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

```
R=rref((A'))
```

```
R = 4×5
     1     0     0     0    -2
     0     1     0     1    -1
     0     0     1    -1     2
     0     0     0     0     0
```

```
M=shrink(R')
```

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

```
B=colspace(sym(A))
```

```
B =
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & -1 \\ -2 & -1 & 2 \end{pmatrix}$$

```
D=double(B)
```

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

```
isequal(D,M)
```

```
ans = logical
   1
```

```
B = double(B);
columnspaces(A,B)
```

```
Col A and Col B are subspaces of R^5
The dimension of ColA is 3
The dimension of ColB is 3
Col A = Col B
```

Bonus 1

```
%The function colspace(sym()) computes the reduced row echelon for of the transpose of A
%and marks it as R. Then it returns the matrix whose columns are the pivot columns of the
%transpose of R.
%The transpose of matrix R is column equivalent to the matrix A. Therefore the column space
%of the transpose of R is equal to the column space of A, so the columns of B, which are the
%pivot columns of the transpose of R form a basis for the column space of the transpose of R
%and hence for the column space of A.
```

Bonus 2

```
type columnspaces_1.m
```

```
function[] = columnspaces_1(A,B)
m = size (A,1);
n = size (B,1);

if m~=n
    disp('Col A and Col B are subspaces of different spaces');
    return
else
    fprintf('Col A and Col B are subspaces of R^%i\n',m);
end

ColA = rank(A);
ColB = rank(B);

dA = 'The dimension of ColA is %d\n';
dB = 'The dimension of ColB is %d\n';
```

```
fprintf(dA, ColA);
fprintf(dB, ColB);

if ColA == m && ColB == m
    fprintf('Col A = Col B = R^%i\n',m);
    return
end

if ColA ~= ColB
    disp('The dimensions of Col A and Col B are different.');
else
    if isequal(double(colspace(sym(A))), double(colspace(sym(B))))
        disp('Col A = Col B');
    else
        disp('The dimensions of Col A and Col B are the same but Col A ~= Col B');
    end
end

end
```

Exercise 3

```
type basis
```

```
function [B,D] = basis(A)
m = size(A,1);
B = shrink(A);

disp('a basis for Col A is the set of columns of')
disp(B);

if rank(B) == m
    fprintf('a basis for R^%i is D=B\n',m)
    D=B;
    return
else
    D = shrink([B eye(m)]);
    if rank(D) == m
        fprintf('a basis for R^%i is\n',m)
        disp(D)
    else
        fprintf('something definitely went wrong!')
    end
end
end
```

```
type shrink
```

```
function B=shrink(A)
[~,pivot]=rref(A);
B=A(:,pivot);
end
```

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

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

```
basis(A);
```

```
a basis for Col A is the set of columns of
     1     0
     0     0
     0     0
     0     1

a basis for R^4 is
     1     0     0     0
     0     0     1     0
     0     0     0     1
     0     1     0     0
```

```
%(b)
A=[0 0;2 0;3 0;0 0]
```

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

```
basis(A);
```

```
a basis for Col A is the set of columns of
     0
     2
     3
     0

a basis for R^4 is
     0     1     0     0
     2     0     1     0
     3     0     0     0
     0     0     0     1
```

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

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

```
basis(A);
```

```
a basis for Col A is the set of columns of
    16     2     3
     5    11    10
     9     7     6
     4    14    15

a basis for R^4 is
    16     2     3     1
     5    11    10     0
     9     7     6     0
     4    14    15     0
```

```
%(d)
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
```

```
basis(A);
```

```
a basis for Col A is the set of columns of
    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

a basis for R^5 is D=B
```

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

```
A = 4×4
     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

```
basis(A);
```

```
a basis for Col A is the set of columns of
     1
     1
     1
     1

a basis for R^4 is
     1     1     0     0
     1     0     1     0
     1     0     0     1
     1     0     0     0
```

Part II. Isomorphism & Change of Basis

Exercise 4

```
type closetozeroroundoff.m
```

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

```
type polyspace
```

```
function P = polyspace(B,Q,r)
%this function creates a matrix of coefficients of the polynomials in
%B and determines if such matrix is a basis for Pn-1. If it is,
%the function finds the B-coordinate vector of polynomial Q, and finally
%the standard-basis polynomial of basis-B vector r.
format;
```

```
u = sym2poly(B(1));
n = length(u);

for i=1:length(B) %creates matrix of coefficients P by iterating through each polynomial
    col = sym2poly(B(i));
    P(:,i) = transpose(col);
end

P=closetozeroroundoff(P,7);

fprintf('matrix of E-coordinate vectors of polynomials in B is\n')

disp(P);

if rank(P) ~= n %determines if the polynomials in B do not form a basis for Pn-1
    sprintf('the polynomials in B do not form a basis for P%d',n-1)
    fprintf('the reduced echelon form of P is\n')
    A = rref(P); %computes and displays the reduced echelon form of P
    disp(A);
    return

else %continues the function if P is a basis for Pn-1
    sprintf('the polynomials in B form a basis for P%d',n-1)

   %part 1 finds the B-coordinate vector of polynomial Q

    cob = sym2poly(Q); %converts the polynomial Q into a row vector
    cob = closetozeroroundoff(cob,7);
    cob = transpose (cob); %converts the row vector into a column vector

    y = P\cob; %performs the change-of-coordinates equation on the vector from Q

    fprintf('the B-coordinate vector of Q is \n')
    disp(y);

    %part 2 finds the standard basis polynomial R of the B-coordinate vector
    %r
    R=0; %initiates the polynomial R

    for i=1:length(P) %loop to multiply each entry of vector r with the polynomials P0..Pn-1
        rowVectorP = transpose(P(:,i));
        R= R + r(i,:)*poly2sym(rowVectorP);
    end

   fprintf('the polynomial whose B-coordinates form the vector r is\n')
   disp(R)
end

end
```

```
syms x

%(a)
B=[x^3+3*x^2,10^(-8)*x^3+x,10^(-8)*x^3+4*x^2+x,x^3+x]
```

B =

$$\begin{pmatrix} x^3+3\,x^2 & \frac{x^3}{100000000}+x & \frac{x^3}{100000000}+4\,x^2+x & x^3+x \end{pmatrix}$$

```
Q=10^(-8)*x^3-2*x^2+x-1
```

Q =

$$\frac{x^3}{100000000} - 2\,x^2 + x - 1$$

```
r=[1;-3;2;4]
```

```
r = 4×1
     1
    -3
     2
     4
```

```
polyspace(B,Q,r);
```

```
matrix of E-coordinate vectors of polynomials in B is
     1     0     0     1
     3     0     4     0
     0     1     1     1
     0     0     0     0
ans =
'the polynomials in B do not form a basis for P3'
the reduced echelon form of P is
    1.0000         0         0    1.0000
         0    1.0000         0    1.7500
         0         0    1.0000   -0.7500
         0         0         0         0
```

```
%(b)
B=[x^3-1,10^(-8)*x^3+2*x^2,10^(-8)*x^3+x,x^3+x]
```

B =

$$\left(x^3 - 1 \quad \frac{x^3}{100000000} + 2\,x^2 \quad \frac{x^3}{100000000} + x \quad x^3 + x\right)$$

```
Q=10^(-8)*x^3-2*x^2+x-1
```

Q =

$$\frac{x^3}{100000000} - 2\,x^2 + x - 1$$

```
r=[1;-3;2;4]
```

```
r = 4×1
     1
    -3
     2
     4
```

```
polyspace(B,Q,r);
```

```
matrix of E-coordinate vectors of polynomials in B is
     1     0     0     1
     0     2     0     0
     0     0     1     1
    -1     0     0     0
ans =
'the polynomials in B form a basis for P3'
the B-coordinate vector of Q is
     1
```

```
    -1
     2
    -1
```

```
the polynomial whose B-coordinates form the vector r is
```

$5\,x^3-6\,x^2+6\,x-1$

```
%(c)
B=[x^4+x^3+x^2+1,10^(-8)*x^4+x^3+x^2+x+1,10^(-8)*x^4+x^2+x+1,10^(-8)*x^4+x+1,10^(-8)*x^4+1]
```

```
B =
```

$$\left(x^4+x^3+x^2+1 \quad \frac{x^4}{100000000}+x^3+x^2+x+1 \quad \frac{x^4}{100000000}+x^2+x+1 \quad \frac{x^4}{100000000}+x+1 \quad \frac{x^4}{100000\cdots}\right.$$

```
Q=x^4-2*x+3
```

Q = $x^4-2\,x+3$

```
M=magic(5);
r=M(:,1)
```

```
r = 5×1
    17
    23
     4
    10
    11
```

```
polyspace(B,Q,r);
```

```
matrix of E-coordinate vectors of polynomials in B is
     1     0     0     0     0
     1     1     0     0     0
     1     1     1     0     0
     0     1     1     1     0
     1     1     1     1     1
ans =
'the polynomials in B form a basis for P4'
the B-coordinate vector of Q is
     1
    -1
     0
    -1
     4
```

```
the polynomial whose B-coordinates form the vector r is
```

$17\,x^4+40\,x^3+44\,x^2+37\,x+65$

Part III. Application to Calculus

Exercise 5

```
type reimsum
```

```
function T =reimsum(fun,a,b,n)
format compact
N = length(n);
```

```
%Riemann sum calculations
for j = 1:N
    h(j) = (b-a)/n(j);

% initialize sums to zero
Il = 0;
Ir = 0;
Im = 0;
    for i = 1:n(j)
        xl = a + h(j)*(i-1);
        xr = a + h(j)*(i);
        xm = a + h(j)*(2*i-1)/2;

        Il = Il + fun(xl)*h(j);
        Ir = Ir + fun(xr)*h(j);
        Im = Im + fun(xm)*h(j);
    end
% populates vectors L,M and R with Riemann sums
L(j,1) = Il;
M(j,1) = Im;
R(j,1) = Ir;
end

A = [n,L,M,R];
%Converts the N-by-4 array A into an N-by-4 table T with the names
% of the variables
T = array2table(A,'VariableNames',{'n','Left','Middle','Right'});

end
```

```
syms x
format long

%(a)
fun=@(x) x.*tan(x) + x + 1
```

```
fun = function_handle with value:
    @(x)x.*tan(x)+x+1
```

```
a=0;b=1;
n=(1:10)';
T = reimsum(fun,a,b,n)
```

T = 10×4 table

	n	Left	Middle	Right
1	1	1.0000	1.7732	3.5574
2	2	1.3866	1.8813	2.6653
3	3	1.5467	1.9062	2.3991
4	4	1.6339	1.9155	2.2733
5	5	1.6888	1.9199	2.2003
6	6	1.7264	1.9224	2.1526
7	7	1.7538	1.9239	2.1192
8	8	1.7747	1.9249	2.0944
9	9	1.7911	1.9255	2.0753

	n	Left	Middle	Right
10	10	1.8044	1.9260	2.0601

```
%The middle points on the subinterval partition seem to be the best
%as they are closer to the approximation which is 1.928...
n=[1;5;10;100;1000;10000];
T=reimsum(fun,a,b,n)
```

T = 6×4 table

	n	Left	Middle	Right
1	1	1.0000	1.7732	3.5574
2	5	1.6888	1.9199	2.2003
3	10	1.8044	1.9260	2.0601
4	100	1.9153	1.9281	1.9409
5	1000	1.9268	1.9281	1.9294
6	10000	1.9280	1.9281	1.9282

```
Int=integral(fun,a,b)
```

```
Int =
   1.928088301365176
```

```
%(b)
fun=@(x) x.^4 - 2*x - 2
```

```
fun = function_handle with value:

    @(x)x.^4-2*x-2
```

```
a=0;b=3;
n=(1:10)';
T=reimsum(fun,a,b,n)
```

T = 10×4 table

	n	Left	Middle	Right
1	1	-6.0000	0.1875	219.0000
2	2	-2.9063	23.9180	109.5938
3	3	5.0000	29.1875	80.0000
4	4	10.5059	31.0964	66.7559
5	5	14.3270	31.9913	59.3270
6	6	17.0938	32.4805	54.5938
7	7	19.1783	32.7764	51.3211
8	8	20.8011	32.9689	48.9261
9	9	22.0988	33.1011	47.0988
10	10	23.1592	33.1957	45.6592

```
%The middle points on the subinterval partition seem to be the best
%as they are closer to the approximation which is 33.600...
n=[1;5;10;100;1000;10000];
T=reimsum(fun,a,b,n)
```

T = 6×4 table

	n	Left	Middle	Right
1	1	-6.0000	0.1875	219.0000
2	5	14.3270	31.9913	59.3270
3	10	23.1592	33.1957	45.6592
4	100	32.4831	33.5960	34.7331
5	1000	33.4876	33.6000	33.7126
6	10000	33.5888	33.6000	33.6113

```
Int=integral(fun,a,b)
```

```
Int =
  33.600000000000001
```

Exercise 6

```
%Prints the function 'polint' into Live Script.
type polint
```

```
%Creates the function 'polint'. Accepts as an input a polynomial P.
function I = polint(P)
% Command 'sym2poly' - Returns numeric coefficients c of input 'P'
c = sym2poly(P);

% Gathers coefficients
c = c(end:-1:1);
A = [];

%'For loop' for integration calculations on 'c' coefficients
    for i = 1:length(c)
        A(i) = c(i)/i;
    end
A = A(end:-1:1);

% Sets arbitrary constant to '3'
A = [A 3];

% Output 'I' is polynomial written in a symbolic form through the
% standard basis.
I = poly2sym(A);
end
```

```
format compact
syms x
%(a)
%Given polynomial assigned to 'P'
P = 6*x^5+5*x^4+4*x^3+3*x^2+2*x+6
```

P = $6\,x^5+5\,x^4+4\,x^3+3\,x^2+2\,x+6$

```
% The output I is polynomial integrated written in a symbolic form through the standard basis
I = polint(P)
```

I = $x^6+x^5+x^4+x^3+x^2+6\,x+3$

```
%Logical command to make sure that output matches the output of a MATLAB built-in function
isequal(I,int(P)+3)
```

```
ans = logical

   1
```

```
%(b)
%Given polynomial assigned to 'P'
P = x^5-2*x^3+3*x+5
```

P = $x^5-2\,x^3+3\,x+5$

```
% The output I is polynomial integrated written in a symbolic form through the standard basis
I = polint(P)
```

I =

$$\frac{x^6}{6}-\frac{x^4}{2}+\frac{3\,x^2}{2}+5\,x+3$$

```
%Logical command to make sure that output matches the output of a MATLAB built-in function
isequal(I,int(P)+3)
```

```
ans = logical

   1
```

Part IV. Application to Markov Chains

Exercise 7

```
type markov
```

```
function q=markov(P,x0)
format
n=size(P,1);

%determine if the matrix P is left stochastic:
columnSum = sum(P,1);
if (~isequal(columnSum,ones(size(columnSum))))
    q=[]
    fprintf("P is not a stochastic matrix");
    return
end

%(1) find the unique steady-state vector q
% basis for probability vector
Q = null(P-eye(n),'r');
% finds the sum of the entries of the columns of Q
sumTotal = sum(Q);
% scale vector Q,since we need entries to add up to 1 for it to be a probability vector.
q = (Q/sumTotal)
fprintf("q is the steady-state vector of the system.");
```

```
%(2)verify that the markov chain converges to q
k = 0; % tracks iterations until achieveing desired accuracy
xk = x0; % initialize xk to x0
while (norm(xk - q) > (10^-7))
    xk = P*xk; %finds next Xk
    k = k+1; %counter increases by 1 each iteration
end
fprintf("\n\nNumber of iterations to verify that\nthe Markov chain converges to q: " + k);
```

```
%(a)
P=[.6 .3;.5 .7]
```

```
P = 2×2
   0.600000000000000   0.300000000000000
   0.500000000000000   0.700000000000000
```

```
x0=[.3;.7]
```

```
x0 = 2×1
   0.300000000000000
   0.700000000000000
```

```
q = markov(P,x0);
```

```
q =

     []

P is not a stochastic matrix
```

```
%(b)
P=[.5 .3;.5 .7]
```

```
P = 2×2
    0.5000    0.3000
    0.5000    0.7000
```

```
%x0 is the same as in part (a)
q = markov(P,x0);
```

```
q = 2×1
    0.3750
    0.6250
q is the steady-state vector of the system.

Number of iterations to verify that
the Markov chain converges to q: 9
```

```
%(c)
P=[.9 .2;.1 .8] %where P is a migration matrix between two regions.
```

```
P = 2×2
    0.9000    0.2000
    0.1000    0.8000
```

```
x0=[.10;.90]
```

```
x0 = 2×1
    0.1000
```

```
    0.9000
```

```
q = markov(P,x0);
```

```
q = 2×1
    0.6667
    0.3333
q is the steady-state vector of the system.

Number of iterations to verify that
the Markov chain converges to q: 45
```

```
%(d)
% Migration matrix P is the same as in (c)
x0=[.81;.19]
```

```
x0 = 2×1
    0.8100
    0.1900
```

```
q = markov(P,x0);
```

```
q = 2×1
    0.6667
    0.3333
q is the steady-state vector of the system.

Number of iterations to verify that
the Markov chain converges to q: 41
```

```
%(e)
P=[.90 .01 .09;.01 .90 .01;.09 .09 .90]
```

```
P = 3×3
    0.9000    0.0100    0.0900
    0.0100    0.9000    0.0100
    0.0900    0.0900    0.9000
```

```
x0=[.5; .3; .2]
```

```
x0 = 3×1
    0.5000
    0.3000
    0.2000
```

```
q = markov(P,x0);
```

```
q = 3×1
    0.4354
    0.0909
    0.4737
q is the steady-state vector of the system.

Number of iterations to verify that
the Markov chain converges to q: 128
```

```
%{
The output vectors q from (c) and (d) are the same. The choice of the
initial vecotr x0 does not have an effect on the steady-state vector
q. Since x0 isnt used to calculate q, q only changes if P changes (Pq=q).
However, the choice of x0 will have an effect on the number of iterations.
```

```
x0 is the inital vector meaning that if P doesnt change, the next
vector(x1,x2,x3..xk) is calculated by xk = P*x(k-1), a different x0 will
generate a different set of products, and will cause the number of iterations to
change.
%}
```