Linear Algebra HW2

Shih-Yu Lai

Department of Psychology, National Cheng Kung University, Tainan, Taiwan
D84099084@gs.ncku.edu.tw

1 [1.4] ex. 11.

(a) $\boldsymbol{A = C + C^T}$

A is symmetric because $(C + C^T)^T = C^T + (C^T)^T = C^T + C = C + C^T = A$.

(b) $\boldsymbol{B = C - C^T}$

B is skew-symmetric because $(C-C^T)^T = C^T-(C^T)^T = C^T-C = -(C-C^T) = -B$.

(c) $\boldsymbol{D = C^T C}$

D could be non-symmetric because the transpose of D is $(C^T C)^T = C^T(C^T)^T$ which does not equal $C^T C$ unless C is symmetric, which it is not by definition.

(d) $\boldsymbol{E = C^T C - CC^T}$

E could be non-symmetric for the same reason as D, as $(C^T C)^T \neq CC^T$ and $(CC^T)^T \neq C^T C$ in general. Hence, E does not necessarily equal its transpose.

(e) $\boldsymbol{F = (I + C)(I + C^T)}$

F could be non-symmetric. The transpose of F is $(I + C)^T(I + C^T)^T = (I + C^T)(I + C)$ which is the same as F, but that does not mean F is symmetric since the multiplication of two symmetric matrices is not necessarily symmetric.

(f) $\boldsymbol{G = (I + C)(I - C^T)}$

G could be non-symmetric. The transpose of G is $(I - C^T)^T(I + C)^T = (I - C)(I + C^T)$, which is not necessarily equal to G since C is not symmetric.

Thus, only A is necessarily symmetric and B is necessarily skew-symmetric, while $D, E, F,$ and G could be non-symmetric.

2 [1.5] ex. 14.

U has the form:

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

R has the form:

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & r_{nn} \end{bmatrix}$$

The product $T = UR$ will result in a matrix where the element at position (i, j) is computed as:

$$t_{ij} = \sum_{k=1}^{n} u_{ik} r_{kj}$$

For T to be upper triangular, $t_{ij} = 0$ for all $i > j$.

1. When $i > j$, since U is upper triangular, for any $k < i$, $u_{ik} = 0$. Also, since R is upper triangular, for any $k > j$, $r_{kj} = 0$. Thus, all terms in the sum $\sum_{k=1}^{n} u_{ik} r_{kj}$ will be zero since either u_{ik} or r_{kj} will be zero, or both. Hence, $t_{ij} = 0$ for all $i > j$, confirming T is upper triangular.
2. For the diagonal elements t_{jj}, the sum simplifies to just one term:

$$t_{jj} = \sum_{k=1}^{n} u_{jk} r_{kj} = u_{jj} r_{jj}$$

since for all $k < j$, $r_{kj} = 0$, and for all $k > j$, $u_{jk} = 0$. Therefore, the diagonal of T will be the product of the diagonals of U and R, i.e., $t_{jj} = u_{jj} r_{jj}$.

Hence, T is upper triangular and the diagonal elements of T are the products of the corresponding diagonal elements of U and R.

3 [1.5] ex. 24.

(a) If A is row equivalent to B, and B is row equivalent to C, then A is row equivalent to C.

Proof: Being row equivalent means that one matrix can be transformed into another by a sequence of elementary row operations. These operations can be represented by multiplication by elementary matrices. If A is row equivalent to B, there exists a finite sequence of elementary matrices $E_1, E_2, \ldots, E_k$ such that:

$$B = E_k E_{k-1} \ldots E_1 A$$

Similarly, if B is row equivalent to C, there exists a finite sequence of elementary matrices $F_1, F_2, \ldots, F_m$ such that:

$$C = F_m F_{m-1} \ldots F_1 B$$

Combining these two, we get:

$$C = F_m F_{m-1} \ldots F_1 E_k E_{k-1} \ldots E_1 A$$

Since the product of elementary matrices is also an elementary matrix (as the set of elementary matrices is closed under multiplication), this shows that A can be transformed into C by a sequence of elementary row operations. Therefore, A is row equivalent to C.

(b) Any two nonsingular $n \times n$ matrices are row equivalent.
Proof: A nonsingular matrix is one that is invertible. If we have two nonsingular matrices A and B, each can be transformed into the identity matrix I using a series of elementary row operations (since they are invertible):

$$I = E_A \cdot A$$
$$I = E_B \cdot B$$

Here, E_A and E_B are products of elementary matrices corresponding to the series of row operations that transform A and B into I, respectively. Since both A and B can be transformed into the identity matrix, they can be transformed into each other by the sequence of operations that first transforms A into I and then I into B:

$$B = E_B \cdot I = E_B \cdot E_A^{-1} \cdot A$$

$E_B \cdot E_A^{-1}$ is again a product of elementary matrices, which proves that A and B are row equivalent.

4 [1.6] ex. 11.

For the matrix

$$A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$$

where A_{11} and A_{22} are nonsingular $n \times n$ matrices, the inverse of A, if it exists, can be found using the formula for the inverse of a block matrix. The inverse of A is given by

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} & -A_{11}^{-1}A_{12}A_{22}^{-1} \\ 0 & A_{22}^{-1} \end{bmatrix}$$

(a) The matrix A is nonsingular because both A_{11} and A_{22} are nonsingular. Therefore, A^{-1} exists and has the block structure shown above.

To determine the block C which corresponds to the upper-right block of A^{-1}, we can multiply A by A^{-1} and set the result equal to the identity matrix:

$$AA^{-1} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \begin{bmatrix} A_{11}^{-1} & C \\ 0 & A_{22}^{-1} \end{bmatrix} = I$$

Multiplying out the matrices gives:

$$\begin{bmatrix} A_{11}A_{11}^{-1} & A_{11}C + A_{12}A_{22}^{-1} \\ 0 & A_{22}A_{22}^{-1} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

(b) For the resulting product to be the identity matrix, the upper-right block must be zero. Therefore, we get the equation:

$$A_{11}C + A_{12}A_{22}^{-1} = 0$$

Solving for C gives:

$$C = -A_{11}^{-1} A_{12} A_{22}^{-1}$$

Thus, the complete inverse of A is:

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} & -A_{11}^{-1} A_{12} A_{22}^{-1} \\ 0 & A_{22}^{-1} \end{bmatrix}$$

5 [Matlab ex] ex. 5.

Part (a):
Generate a 6×6 matrix A with entries as the floor of 10 times random numbers between 0 and 1. Generate a vector b with entries as the floor of 20 times random numbers between 0 and 1, subtracted by 10. Solve for x in $Ax = b$.

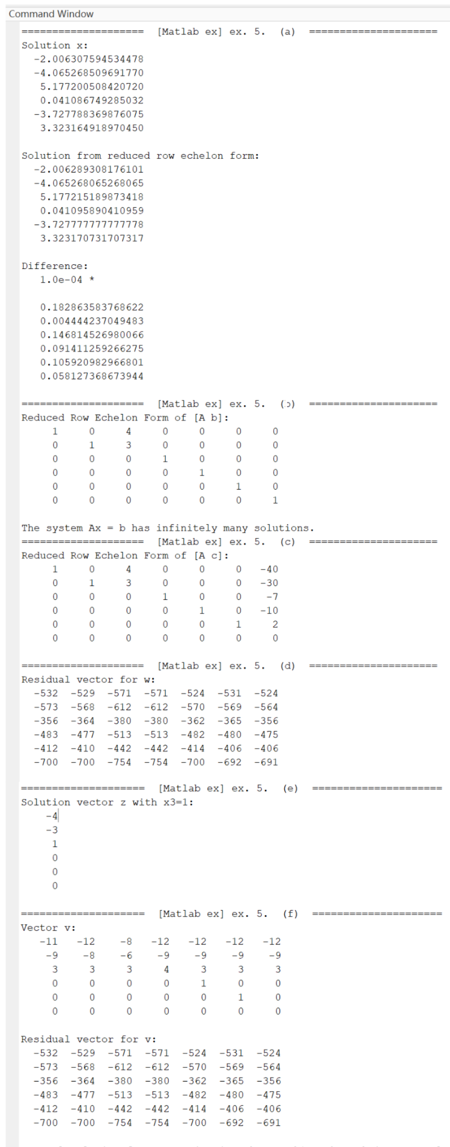
Part (b):
Make A singular by setting its third column to be a linear combination of the first two columns. Compute the RREF of the augmented matrix $[A \quad b]$ and discuss the number of solutions.

Part (c):
Generate a new vector c as A times a new random vector. Find the RREF of $[A \quad c]$ to explore the solutions of $Ax = c$.

Part (d):
Identify the solution associated with the free variable $x_3 = 0$ from the RREF found in part (c). Compute the residual vector $c - Ax$ to check the solution.

Part (e):
Compute the null space of A. Determine a vector z in the null space with the third entry equal to 1.

Part (f):
Form a new solution v as $w + 3z$. Explain that v is a solution to $Ax = c$ and use MATLAB to compute the residual $c - Av$. Discuss how the general solution of $Ax = c$ can be expressed in terms of w and z.



```
Command Window
====================  [Matlab ex] ex. 5.  (a)  =====================
Solution x:
  -2.006307594534478
  -4.065268509691770
   5.177200508420720
   0.041086749285032
  -3.727788369876075
   3.323164918970450

Solution from reduced row echelon form:
  -2.006289308176101
  -4.065268065268065
   5.177215189873418
   0.041095890410959
  -3.727777777777778
   3.323170731707317

Difference:
   1.0e-04 *

   0.182863583768622
   0.004444237049483
   0.146814526980066
   0.091411259266275
   0.105920982966801
   0.058127368673944

====================  [Matlab ex] ex. 5.  (b)  =====================
Reduced Row Echelon Form of [A b]:
     1     0     4     0     0     0     0
     0     1     3     0     0     0     0
     0     0     0     1     0     0     0
     0     0     0     0     1     0     0
     0     0     0     0     0     1     0
     0     0     0     0     0     0     1

The system Ax = b has infinitely many solutions.
====================  [Matlab ex] ex. 5.  (c)  =====================
Reduced Row Echelon Form of [A c]:
     1     0     4     0     0     0   -40
     0     1     3     0     0     0   -30
     0     0     0     1     0     0    -7
     0     0     0     0     1     0   -10
     0     0     0     0     0     1     2
     0     0     0     0     0     0     0

====================  [Matlab ex] ex. 5.  (d)  =====================
Residual vector for w:
  -532  -529  -571  -571  -524  -531  -524
  -573  -568  -612  -612  -570  -569  -564
  -356  -364  -380  -380  -362  -365  -356
  -483  -477  -513  -513  -482  -480  -475
  -412  -410  -442  -442  -414  -406  -406
  -700  -700  -754  -754  -700  -692  -691

====================  [Matlab ex] ex. 5.  (e)  =====================
Solution vector z with x3=1:
    -4
    -3
     1
     0
     0
     0

====================  [Matlab ex] ex. 5.  (f)  =====================
Vector v:
   -11   -12    -8   -12   -12   -12   -12
    -9    -8    -6    -9    -9    -9    -9
     3     3     3     4     3     3     3
     0     0     0     0     1     0     0
     0     0     0     0     0     1     0
     0     0     0     0     0     0     0

Residual vector for v:
  -532  -529  -571  -571  -524  -531  -524
  -573  -568  -612  -612  -570  -569  -564
  -356  -364  -380  -380  -362  -365  -356
  -483  -477  -513  -513  -482  -480  -475
  -412  -410  -442  -442  -414  -406  -406
  -700  -700  -754  -754  -700  -692  -691

General solution for Ax = c is given by w + k*z, where k is any scalar.
```

Fig. 1. Results of [Matlab ex] ex. 5.

```matlab
%% Main Program
clear; clc;

%% Part (a)

% Generate matrix A and vector b
A = floor(10 * rand(6));
b = floor(20 * rand(6, 1)) - 10;

% Solve the system Ax = b for x
x = A\b;

% Compute the reduced row echelon form of [A b]
U = rref([A b]);

% Extract the solution from the last column of U
U_solution = U(:, end);

% Compute the difference
difference = U(:, 7) - x;

% Display results
format long
disp('====================  [Matlab ex] ex. 5.  (a)  ====================');
disp('Solution x:');
disp(x);
disp('Solution from reduced row echelon form:');
disp(U_solution);
disp('Difference:');
disp(difference);

%% Part (b)

% Modify A to make it singular
A(:,3) = A(:,1:2) * [4; 3];

% Compute the reduced row echelon form of the augmented matrix [A b]
rref_Ab = rref([A b]);

% Display the rref of [A b]
disp('====================  [Matlab ex] ex. 5.  (b)  ====================');
disp('Reduced Row Echelon Form of [A b]:');
disp(rref_Ab);

% Determine the number of solutions
% If the last row of rref_Ab is all zeros except for the last element,
% then there are no solutions. Otherwise, there are infinitely many.
if any(rref_Ab(end,1:end-1)) && rref_Ab(end,end) ~= 0
    disp('The system Ax = b has no solutions.');
elseif rank(A) < size(A, 1)
    disp('The system Ax = b has infinitely many solutions.');
else
    disp('The system Ax = b has a unique solution.');
end

%% Part (c)
% Generate a new vector c
c = A * (floor(20 * rand(6,1)) - 10);

% Compute the reduced row echelon form of [A c]
U_Ac = rref([A c]);

% Display the reduced row echelon form
disp('====================  [Matlab ex] ex. 5.  (c)  ====================');
disp('Reduced Row Echelon Form of [A c]:');
disp(U_Ac);

%% Part (d)
% Assuming U is from part (c)
% The solution vector for x3 = 0
w = U_Ac;
w(:, end) = 0; % Set the last column to zero assuming x3 = 0

% Check the solution
residual_w = c - A * w;

% Display the residual vector
disp('====================  [Matlab ex] ex. 5.  (d)  ====================');
disp('Residual vector for w:');
disp(residual_w);

%% Part (e)

% Assuming U_Ac is the reduced row echelon form from part (c)
U = U_Ac;
U(:, end) = zeros(6, 1); % Set the last column to zero

% Find the null space of A
nullA = null(A, 'rational');

% If there is no null space (i.e., A is full rank), we cannot find a non-trivial z
if isempty(nullA)
    error('Matrix A is full rank, no non-trivial solution to Ax=0 exists.');
else
    % Set x3 to 1 in the solution to Ax=0, if possible
    % Adjust this depending on the structure of your null space vectors
    z = nullA(:, end); % Take the last vector of the basis of null space
    % If this doesn't directly give us x3=1, we might need to scale the vector
    % or combine vectors from the null space basis, depending on their structure.
end

% Display the solution vector z
disp('====================  [Matlab ex] ex. 5.  (e)  ====================');
disp('Solution vector z with x3=1:');
disp(z);

%% Part (f)

% Assuming w is the solution vector from part (d) when x3 = 0
% and z is the particular solution from part (e) when x3 = 1

% Define vector v as w + 3 * z
v = w + 3 * z;

% Verify that v is a solution by checking that Av = c
residual_v = c - A * v;

% Display the results
disp('====================  [Matlab ex] ex. 5.  (f)  ====================');
disp('Vector v:');
disp(v);
disp('Residual vector for v:');
disp(residual_v);

% Explain how all solutions of the system Ax = c can be described
% Any solution can be expressed as a linear combination of a particular
% solution (w) and a homogeneous solution (z) scaled by a free variable
% (in this case, the scalar multiple of z).
disp('General solution for Ax = c is given by w + k*z, where k is any scalar.');
```

Fig. 2. Codes of [Matlab ex] ex. 5.

6 [Matlab ex] ex. 6(a) to 6(d).

Part (a):

This part involves determining the adjacency matrix A for the given graph. An adjacency matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.

Part (b):

Once we have the adjacency matrix A, we compute A^2. In graph theory, the entry in the i-th row and j-th column of A^2 represents the number of walks of length 2 from vertex V_i to vertex V_j. We specifically look at the walks from V_1 to V_7, V_4 to V_5, to V_6, and V_8 to V_3.

Part (c):

Similarly, computing A^4, A^6, and A^8 gives us the number of walks of lengths 4, 6, and 8, respectively, between any two vertices i and j. The script focuses on the number of such walks from V_1 to V_7.

Part (d):

This part is similar to part (c), but for walks of odd lengths 3, 5, and 7. Again, the entry in the i-th row and j-th column of A^n represents the number of walks of length n from vertex V_i to vertex V_j, with the specific walks from V_1 to V_7 being of interest.

The conjectures made at the end of the script seem to predict whether there are an even or odd number of walks of certain lengths between the vertices V_1 and V_7. The MATLAB code will print the number of walks of these lengths, and the conjectures are based on whether the sum of the number of walks of certain lengths is even or odd.

This explanation corresponds to the parts of the MATLAB code provided and assumes that the graph structure does not change. If the graph structure changes, the adjacency matrix A must be updated accordingly to reflect the correct connections between the vertices.

```
Command Window
====================  [Matlab ex] ex. 6.  (a)  ====================
Adjacency matrix A:
     0     1     0     1     0     0     0     0
     1     0     1     0     1     0     0     0
     0     1     0     0     0     1     0     0
     1     0     0     0     0     0     1     0
     0     1     0     0     0     0     0     1
     0     0     1     0     0     0     1     0
     0     0     0     1     0     1     0     1
     0     0     0     0     1     0     1     0

====================  [Matlab ex] ex. 6.  (b)  ====================
Number of walks of length 2 for each pair of vertices:
     1
     1
     0
     0

====================  [Matlab ex] ex. 6.  (c)  ====================
A^4:
     7     0     6     0     6     0     7     0
     0    12     0     7     0     7     0     7
     6     0     7     0     6     0     7     0
     0     7     0     7     0     6     0     6
     6     0     6     0     7     0     7     0
     0     7     0     6     0     7     0     6
     7     0     7     0     7     0    12     0
     0     7     0     6     0     6     0     7

A^6:
    33     0    32     0    32     0    40     0
     0    57     0    40     0    40     0    40
    32     0    33     0    32     0    40     0
     0    40     0    33     0    32     0    32
    32     0    32     0    33     0    40     0
     0    40     0    32     0    33     0    32
    40     0    40     0    40     0    57     0
     0    40     0    32     0    32     0    33

A^8:
   170     0   169     0   169     0   217     0
     0   291     0   217     0   217     0   217
   169     0   170     0   169     0   217     0
     0   217     0   170     0   169     0   169
   169     0   169     0   170     0   217     0
     0   217     0   169     0   170     0   169
   217     0   217     0   217     0   291     0
     0   217     0   169     0   169     0   170
====================  [Matlab ex] ex. 6.  (d)  ====================
A^3:
     0     4     0     3     0     2     0     2
     4     0     4     0     4     0     3     0
     0     4     0     2     0     3     0     2
     3     0     2     0     2     0     4     0
     0     4     0     2     0     2     0     3
     2     0     3     0     2     0     4     0
     0     3     0     4     0     4     0     4
     2     0     2     0     3     0     4     0

A^5:
     0    19     0    14     0    13     0    13
    19     0    19     0    19     0    21     0
     0    19     0    13     0    14     0    13
    14     0    13     0    13     0    19     0
     0    19     0    13     0    13     0    14
    13     0    14     0    13     0    19     0
     0    21     0    19     0    19     0    19
    13     0    13     0    14     0    19     0

A^7:
     0    97     0    73     0    72     0    72
    97     0    97     0    97     0   120     0
     0    97     0    72     0    73     0    72
    73     0    72     0    72     0    97     0
     0    97     0    72     0    72     0    73
    72     0    73     0    72     0    97     0
     0   120     0    97     0    97     0    97
    72     0    72     0    73     0    97     0

Number of walks of length 2:
     1
     1
     0
     0

Number of walks of length 4:
     7

Number of walks of length 6:
    40

Number of walks of length 8:
   217

Number of walks of length 3:
     0

Number of walks of length 5:
     0

Number of walks of length 7:
     0

Conjecture for walks of even length from V1 to V7:
   0
   0
   1
   1

Conjecture for walks of odd length from V1 to V7:
   0
```

Fig. 3. Results of [Matlab ex] ex. 6(a) to 6(d).

Editor - D:\Linear Algebra\HW2\HW2_1_Matlab_ex_6.m

HW2_1_Matlab_ex_5.m HW2_1_Matlab_ex_6.m

```matlab
% Main Script

% Clear the workspace and command window
clear; clc;

%% (a) Adjacency matrix for the graph
A = [
    0 1 0 1 0 0 0 0;
    1 0 1 0 1 0 0 0;
    0 1 0 0 0 1 0 0;
    1 0 0 0 0 0 1 0;
    0 1 0 0 0 0 0 1;
    0 0 1 0 0 0 1 0;
    0 0 0 1 0 1 0 1;
    0 0 0 0 1 0 1 0
];

% Display the adjacency matrix
disp('====================  [Matlab ex] ex. 6.  (a)  ====================');

disp('Adjacency matrix A:');
disp(A);

%% (b) Compute A^2 and determine the number of walks of length 2
A2 = A^2;
walks_2 = [
    A2(1,7); % Number of walks of length 2 from V1 to V7
    A2(4,8); % Number of walks of length 2 from V4 to V8
    A2(5,6); % Number of walks of length 2 from V5 to V6
    A2(8,3)  % Number of walks of length 2 from V8 to V3
];

% Display the results for part (b)
disp('====================  [Matlab ex] ex. 6.  (b)  ====================');

disp('Number of walks of length 2 for each pair of vertices:');
disp(walks_2);

%% (c) Compute A^4, A^6, and A^8
A4 = A^4;
A6 = A^6;
A8 = A^8;

% Determine the number of walks of length 4, 6, and 8
walks_4 = A4(1,7); % Number of walks of length 4 from V1 to V7
walks_6 = A6(1,7); % Number of walks of length 6 from V1 to V7
walks_8 = A8(1,7); % Number of walks of length 8 from V1 to V7

% Display the results for part (c)
disp('====================  [Matlab ex] ex. 6.  (c)  ====================');

disp('A^4:');
disp(A4);
disp('A^6:');
disp(A6);
disp('A^8:');
disp(A8);

%% (d) Compute A^3, A^5, and A^7
A3 = A^3;
A5 = A^5;
A7 = A^7;

% Display the results for part (d)
disp('====================  [Matlab ex] ex. 6.  (d)  ====================');

disp('A^3:');
disp(A3);
disp('A^5:');
disp(A5);
disp('A^7:');
disp(A7);

% Determine the number of walks of length 3, 5, and 7
walks_3 = A3(1,7); % Number of walks of length 3 from V1 to V7
walks_5 = A5(1,7); % Number of walks of length 5 from V1 to V7
walks_7 = A7(1,7); % Number of walks of length 7 from V1 to V7


% Display the results
disp('Number of walks of length 2:');
disp(walks_2);

disp('Number of walks of length 4:');
disp(walks_4);

disp('Number of walks of length 6:');
disp(walks_6);

disp('Number of walks of length 8:');
disp(walks_8);

disp('Number of walks of length 3:');
disp(walks_3);

disp('Number of walks of length 5:');
disp(walks_5);

disp('Number of walks of length 7:');
disp(walks_7);

% Conjecture
% Assuming the starting point for the conjecture is vertex V1 and the ending point is V7
conjecture_even = mod(walks_2 + walks_4 + walks_6 + walks_8, 2) == 0;
conjecture_odd = mod(walks_3 + walks_5 + walks_7, 2) == 1;

disp('Conjecture for walks of even length from V1 to V7:');
disp(conjecture_even);

disp('Conjecture for walks of odd length from V1 to V7:');
disp(conjecture_odd);
```

Fig. 4. Codes of [Matlab ex] ex. 6(a) to 6(d).