J Viswaksena - AIE21035

21AIE315 AI in Speech Processing

Sheet 1

MATRIX, the foundation for MATLAB

```
C = [-1, 0, 0; 1,1,0; 1,-1,0; 0,0,2]
```

```
C = 4×3
    -1     0     0
     1     1     0
     1    -1     0
     0     0     2
```

```
D = [-1, 0, 0
 1, 1, 0
 1, -1, 0
 0, 0, 2]
```

```
D = 4×3
    -1     0     0
     1     1     0
     1    -1     0
     0     0     2
```

The semicolon here is going to distinguish the row and colomns in the matrix.

```
B = [3,5,6]
```

```
B = 1×3
     3     5     6
```

```
S = [7, 8, B]
```

```
S = 1×5
     7     8     3     5     6
```

another matrix can be instuted under another matrix.

The matrix can be extended/modified by defining new elements.

```
S(2) = 2
```

```
S = 1×5
     7     2     3     5     6
```

```
S(6) = 9
```

```
S = 1×6
     7     2     3     5     6     9
```

```
S(9) = 13
```

```
S = 1×9
     7     2     3     5     6     9     0     0    13
```

```
I = [1,2,4];
S(I) = 42
```

```
S = 1×9
    42    42     3    42     6     9     0     0    13
```

```
S(I+1) = 45
```

```
S = 1×9
    42    45    45    42    45     9     0     0    13
```

Colon Operator

The colon operator is a very powerful operator for creating new matrices.

```
% Create a row vector from 1 to 10
vec = 1:10
```

```
vec = 1×10
     1     2     3     4     5     6     7     8     9    10
```

```

% Create a column vector from 1 to 10
vec_column = (1:10)'
```

```
vec_column = 10×1
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
```

```
% Create a 3x3 matrix
A = [1, 2, 3; 4, 5, 6; 7, 8, 9]
```

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

```

% Select the second row of matrix A
row_2 = A(2, :)
```

```
row_2 = 1×3
```

```
     4     5     6
```

```
% Select the second column of matrix A
col_2 = A(:, 2)
```

```
col_2 = 3×1
     2
     5
     8
```

```
% Select a submatrix
submatrix = A(1:2, 2:3)% Rows 1 to 2, Columns 2 to 3
```

```
submatrix = 2×2
     2     3
     5     6
```

```
% Generate a vector from 0 to 2 in steps of 0.1
range = 0:0.1:2
```

```
range = 1×21
         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.90
```

```
% Generate a vector from 10 to 1 in steps of -1
reverse_range = 10:-1:1
```

```
reverse_range = 1×10
    10     9     8     7     6     5     4     3     2     1
```

```
% Create a 3x4 matrix
B = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12]
```

```
B = 3×4
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

```
% Reshape B into a 2x6 matrix
reshaped_B = reshape(B, 2, 6)
```

```
reshaped_B = 2×6
     1     9     6     3    11     8
     5     2    10     7     4    12
```

```
% Create a 2x3 matrix
C = [1, 2, 3; 4, 5, 6]
```

```
C = 2×3
     1     2     3
     4     5     6
```

```
% Access the second element using linear indexing
second_element = C(2)
```

```
second_element = 4
```

Matrix Operator

```
% Define two matrices
A = [1, 2, 3; 4, 5, 6; 7, 8, 9];
B = [9, 8, 7; 6, 5, 4; 3, 2, 1];

% Matrix addition
C = A + B
```

```
C = 3×3

    10    10    10
    10    10    10
    10    10    10
```

```
% Matrix subtraction
D = A - B
```

```
D = 3×3

    -8    -6    -4
    -2     0     2
     4     6     8
```

```
% Matrix multiplication
E = A * B
```

```
E = 3×3

    30    24    18
    84    69    54
   138   114    90
```

```
% Element-wise multiplication
F = A .* B
```

```
F = 3×3

     9    16    21
    24    25    24
    21    16     9
```

```
% Transpose of A
A_transpose = A'
```

```
A_transpose = 3×3

     1     4     7
     2     5     8
     3     6     9
```

```
% Inverse of A
A_inverse = inv(A)
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =  1.541976
18.
A_inverse = 3×3
```

10^{16} ×

```
   -0.4504    0.9007   -0.4504
    0.9007   -1.8014    0.9007
   -0.4504    0.9007   -0.4504
```

```
% Matrix exponentiation
A_squared = A^2
```

```
A_squared = 3×3
    30    36    42
    66    81    96
   102   126   150
```

```
zeros(6)
```

```
ans = 6×6
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
zeros(3,2)
```

```
ans = 3×2
     0     0
     0     0
     0     0
```

```
ones(6)
```

```
ans = 6×6
     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     1     1     1     1     1
     1     1     1     1     1     1
```

```
ones(3,2)
```

```
ans = 3×2
     1     1
     1     1
     1     1
```

```
C = [1, 2, 3 ; 4,5,6; 7,8,9]
```

```
C = 3×3
```

```
     1     2     3
     4     5     6
     7     8     9
```

```
% prints the diagonal of D
diag(C)
```

```
ans = 3×1

     1
     5
     9
```

Labsheet -1

lab exercises

1

```
help sqrt
```

```
 sqrt   Square root.
    sqrt(X) is the square root of the elements of X. Complex
    results are produced if X is not positive.

    See also sqrtm, realsqrt, hypot.

    Documentation for sqrt
    Other uses of sqrt
```

```
x = sqrt(4)
```

```
x = 2
```

```
y = sqrt(9)
```

```
y = 3
```

```
% we get square roots for the variables given
```

```
 who
```

```
Your variables are:

A            A_inverse    A_squared    A_transpose  B            C            D
```

```
 whos
```

```
  Name             Size            Bytes  Class     Attributes

  A                3x3                72  double
  A_inverse        3x3                72  double
  A_squared        3x3                72  double
  A_transpose      3x3                72  double
  B                3x3                72  double
  C                3x3                72  double
```

```
  D                    3x3                72  double
  E                    3x3                72  double
  F                    3x3                72  double
  I                    1x3                24  double
  S                    1x9                72  double
  ans                  3x1                24  double
  col_2                3x1                24  double
  range                1x21              168  double
  reshaped_B           2x6                96  double
  reverse_range        1x10               80  double
  row_2                1x3                24  double
  second_element       1x1                 8  double
  submatrix            2x2                32  double
  vec                  1x10               80  double
```

```
z = sin(0)
```

```
z = 0
```

```
z1 = sin(pi/2)
```

```
z1 = 1
```

```
whos
```

```
  Name                 Size             Bytes  Class     Attributes

  A                    3x3                72  double
  A_inverse            3x3                72  double
  A_squared            3x3                72  double
  A_transpose          3x3                72  double
  B                    3x3                72  double
  C                    3x3                72  double
  D                    3x3                72  double
  E                    3x3                72  double
  F                    3x3                72  double
  I                    1x3                24  double
  S                    1x9                72  double
  ans                  3x1                24  double
  col_2                3x1                24  double
  range                1x21              168  double
  reshaped_B           2x6                96  double
  reverse_range        1x10               80  double
  row_2                1x3                24  double
  second_element       1x1                 8  double
  submatrix            2x2                32  double
  vec                  1x10               80  double
  vec_column          10x1                80  double
  x                    1x1                 8  double
```

```
clc
who
```

```
Your variables are:

A                A_inverse        A_squared        A_transpose      B                C                D
```

```
clear
who
size(x)
```

```
Unrecognized function or variable 'x'.
```

```
a = [1 2 3]
```

```
a = 1×3
     1     2     3
```

```
b = [3 4 5]
```

```
b = 1×3
     3     4     5
```

```

a.*b
```

```
ans = 1×3
     3     8    15
```

```
C = [a b]
```

```
C = 1×6
     1     2     3     3     4     5
```

```
zeros(4)
```

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

```
zeros(1, 3)
```

```
ans = 1×3
     0     0     0
```

```
ones(1, 2)
```

```
ans = 1×2
     1     1
```

```
P = [zeros(1,3) 1 1 ones(1,2)]
```

```
P = 1×7
     0     0     0     1     1     1     1
```

```
Y = 1 : 0.5 : 5
```

```
Y = 1×9
    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000    4.5000    5.0000
```

```
M = [1 2 3
 4 5 6
 7 8 9]
```

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

```
fprintf('%f\n', M)
```

```
1.000000
4.000000
7.000000
2.000000
5.000000
8.000000
3.000000
6.000000
9.000000
```

```
X = M(1,:)
```

```
X = 1×3
     1     2     3
```

2

a.

```
A = [3, 12, 6, 8; 5, 3 , 9, 11; 1, 2, 14, 7; 10, 5, 3, 6]
```

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

```
A = [3, 12, 6, 8
    5, 3 , 9, 11
    1, 2, 14, 7
    10, 5, 3, 6]
```

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

```
E = [3, 12, 6, 8; 5, 3 , 9, 11]
F = [1, 2, 14, 7; 10, 5, 3, 6]
G = [E; F]
```

```
E = 2×4
     3    12     6     8
     5     3     9    11
```

```
F = 2×4

     1     2    14     7
    10     5     3     6

G = 4×4

     3    12     6     8
     5     3     9    11
     1     2    14     7
    10     5     3     6
```

b.

```
A_transpose = A';
disp('Transpose of A:');
```

```
Transpose of A:
```

```
disp(A_transpose);
```

```
     3     5     1    10
    12     3     2     5
     6     9    14     3
     8    11     7     6
```

```
A_inverse = inv(A);
disp('Inverse of A:');
```

```
Inverse of A:
```

```
disp(A_inverse);
```

```
   -0.0490   -0.0607    0.0296    0.1421
    0.0979   -0.0831    0.0091    0.0113
   -0.0140   -0.0920    0.1286    0.0373
    0.0070    0.2165   -0.1211   -0.0982
```

c.

```
size(A)
```

d.

```
% Extract the second row into vector X
X = A(2, :);

% Extract the third column into vector Y
Y = A(:, 3);

% Display the vectors
disp('Vector X (second row of A):');
```

```
Vector X (second row of A):
```

```
disp(X);
```

```
     5     3     9    11
```

```
disp('Vector Y (third column of A):');
```

```
Vector Y (third column of A):
```

```
disp(Y);
```

```
     6
     9
    14
     3
```

e.

```
% Multiply X and Y (performing the dot product)
result = X * Y;

% Display the result
disp('Result of multiplying X and Y:');
```

```
Result of multiplying X and Y:
```

```
disp(result);
```

```
   216
```

f.

```
% Multiply X and Y element-wise
result_elementwise = X .* Y;

% Display the result
disp('Result of element-wise multiplication of X and Y:');
```

```
Result of element-wise multiplication of X and Y:
```

```
disp(result_elementwise);
```

```
    30    18    54    66
    45    27    81    99
    70    42   126   154
    15     9    27    33
```

g.

```
% Define the new row vector
new_row = [1, 5, 9, 0];

% Replace the last row of A with the new row vector
A(end, :) = new_row;

% Display the updated matrix A
disp('Updated matrix A with the last row replaced:');
```

```
Updated matrix A with the last row replaced:
```

```
disp(A);
```

```
     3    12     6     8
     5     3     9    11
     1     2    14     7
     1     5     9     0
```

h.

```
% Print the elements in one row
disp('Matrix A printed in one row:');
```

```
Matrix A printed in one row:
```

```
disp(A(:)');
```

```
     3     5     1     1    12     3     2     5     6     9    14     9     8    11     7     0
```

```

% Print the elements in one column
disp('Matrix A printed in one column:');
```

```
Matrix A printed in one column:
```

```
disp(A(:));
```

```
     3
     5
     1
     1
    12
     3
     2
     5
     6
     9
    14
     9
     8
    11
     7
```

i.

```
% Calculate the total number of elements in A
total_elements = numel(A);

% Print the elements of A in reverse order
disp('Matrix A printed in reverse order:');
```

```
Matrix A printed in reverse order:
```

```
for i = total_elements:-1:1
    fprintf('%d ', A(i));
```

```
end
```

```
0 7 11 8 9 14 9 6 5 2 3 12 1 1 5 3
```

```
fprintf('\n');
```

j.

```
% Extract the diagonal elements of A into vector D
D = diag(A);

% Display the vector D containing diagonal elements
disp('Vector D containing diagonal elements of matrix A:');
```

```
Vector D containing diagonal elements of matrix A:
```

```
disp(D);
```

```
     3
     3
    14
     0
```

k.

```
% Extract the diagonal elements of A into vector D
D = diag(A);

% Create a diagonal matrix V with the extracted diagonal elements
V = diag(D);

% Display the diagonal matrix V
disp('Diagonal matrix V with the extracted diagonal elements:');
```

```
Diagonal matrix V with the extracted diagonal elements:
```

```
disp(V);
```

```
     3     0     0     0
     0     3     0     0
     0     0    14     0
     0     0     0     0
```

l.

```
% Get the size of the matrix V
[size_rows, size_columns] = size(V);

% Create an identity matrix of the same size as V
identity_matrix = eye(size_rows, size_columns);

% Display the identity matrix
disp('Identity matrix of the same size as V:');
```

```
Identity matrix of the same size as V:
```

```
disp(identity_matrix);
```

```
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

m.

```
% Get the size of the matrix V
[size_rows, size_columns] = size(V);

% Create an identity matrix of the same size as V
identity_matrix = eye(size_rows, size_columns);

% Concatenate V and the identity matrix along columns
new_matrix = [V, identity_matrix];

% Create a new matrix double the size by concatenating new_matrix with itself along rows
new_matrix_double_size = [new_matrix; new_matrix];

% Display the new matrix double the size
disp('New matrix double the size:');
```

```
New matrix double the size:
```

```
disp(new_matrix_double_size);
```

```
     3     0     0     0     1     0     0     0
     0     3     0     0     0     1     0     0
     0     0    14     0     0     0     1     0
     0     0     0     0     0     0     0     1
     3     0     0     0     1     0     0     0
     0     3     0     0     0     1     0     0
     0     0    14     0     0     0     1     0
     0     0     0     0     0     0     0     1
```

3.

```
% Create a vector with the specified elements
vector = 0:0.01:1;

% Find the length of the vector
vector_length = length(vector);

% Display the vector and its length
disp('Vector:');
```

```
Vector:
```

```
disp(vector);
```

```
         0    0.0100    0.0200    0.0300    0.0400    0.0500    0.0600    0.0700    0.0800    0.09
```

```
disp(['Length of the vector: ', num2str(vector_length)]);
```

```
Length of the vector: 101
```

4.

```
% Define the vectors x and y
x = [1, 6, 9, 2];
y = [2, 0, 3, 8];

% Element-wise sum of x and y
sum_xy = x + y;

% Element-wise product of x and y
product_xy = x .* y;

% Display the element-wise sum and product
disp('Element-wise sum of x and y:');
```

```
Element-wise sum of x and y:
```

```
disp(sum_xy);
```

```
     3     6    12    10
```

```

disp('Element-wise product of x and y:');
```

```
Element-wise product of x and y:
```

```
disp(product_xy);
```

```
     2     0    27    16
```

5.

```
% Define some variables
x = 1;
y = [2, 3];
z = 'hello';

% Display variable names using who
disp('Variables in workspace (using who):');
```

```
Variables in workspace (using who):
```

```
who
```

```
Your variables are:

A                       A_inverse               A_transpose             C                       D
```

```

% Display detailed information using whos
```

```
disp('Variables in workspace (using whos):');
```

```
Variables in workspace (using whos):
```

```
whos
```

```
  Name                    Size            Bytes  Class     Attributes

  A                       4x4               128  double
  A_inverse               4x4               128  double
  A_transpose             4x4               128  double
  C                       1x6                48  double
  D                       4x1                32  double
  E                       2x4                64  double
  F                       2x4                64  double
  G                       4x4               128  double
  M                       3x3                72  double
  P                       1x7                56  double
  V                       4x4               128  double
  X                       1x4                32  double
  Y                       4x1                32  double
```

6.

```
% Define the matrix U
U = [4, 5, 6, 7];

% Calculate the sum of all elements
sum_U = sum(U);

% Calculate the mean of all elements
mean_U = mean(U);

% Calculate the median of all elements
median_U = median(U);

% Display the results
disp(['Sum of all elements of U: ', num2str(sum_U)]);
```

```
Sum of all elements of U: 22
```

```
disp(['Mean of all elements of U: ', num2str(mean_U)]);
```

```
Mean of all elements of U: 5.5
```

```
disp(['Median of all elements of U: ', num2str(median_U)]);
```

```
Median of all elements of U: 5.5
```