Lesson 3: Vectors and Matrices

Topics:

- Vectors and Matrices (Class of Arrays)
- Generating, manipulating, and indexing arrays

Learning Outcomes:

1. Define vectors (row and column) in MATLAB
2. Generate vectors such as uniformly spaced points
3. Index vectors and assign values to particular entries
4. Define matrices (including ones, zeros, random)
5. Indexing matrices
6. Scalar array operations

Book Sections: 2.1 - 2.3

1. Creating Vectors

A vector is a numeric list such as $[1, 2, 3, 4]$ or $[2.1, -5, \pi/2, 0, 0, 1, \frac{1}{2}]$. Vectors are fundamental objects in mathematics, statistics, and science.

1.1 Row Vectors

```
% Separate with space or comma
a_vector=[1 2 3 4];
same_vector=[1, 2, 3, 4];

another_vector=[2 1 -5 pi 0 0 1 1/2];
```

1.1.1 Generating (Row) vectors of <u>uniform</u> size

```
% Initialize variables
a=1;b=10;n=10;


% 1. Linear Space Function
```

```
% 1. Linear Space Function
% ------------------------------- %
% n Linearly Equally Spaced Points between a and b
% linspace(a,b,n)
% default: n=100, i.e., linspace(a,b)

% n Uniformly space points on [a,b]
x=linspace(a,b,n)
```

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

```
% 1a. Logspace
% ------------------------------- %
% logspace(a,b,n);  % n equally spaced powers of 10
logspace(0,4,5)
```

```
ans = 1×5
          1          10         100        1000       10000
```

```
% 2. Colon Operator
% ------------------------------- %
% start:increment:end (default increment = 1)

x=1:1:10;            % [1,2,3,4,5,6,7,8,9,10]
y=1:10;              % Same as above

u=0:2:20;            % Even numbers up to 20
v=0:2:21;            % Even numbers up to 20

t=0:0.1:1;           % 0, .1, .2, .3, ...

% Note: Contains a few pathologies
% linspace(0,pi,10) vs. [0:pi/10:pi]
```

1.1.2 Indexing vectors

Vectors are ordered lists. The elements have an address, or **index**, (a.k.a. subsript). It is important to note that in MATLAB, ***indices start at 1*** (unlike java, Fortran, C, and Python).

```
x=linspace(0,1);

% Access the third element
x(3)
```

```
ans = 0.0202
```

```

% Access the 65th element
x(65)
```

```
ans = 0.6465
```

```

% Access the 4th, 6th, and 87th
x([4,6,87])
```

```
ans = 1×3
    0.0303    0.0505    0.8687
```

```

% Access the even elements
x(2:2:100)

% ------------------------------- %

% Assign the 3rd element zero
x(3)=0

% Assign the 1st, 4th, 7th, and 10th element 0
x(1:3:10)=0
```

1.2 Column Vectors

Column vectors are row vectors that span vertically. As with row vectors, we can define them directly, or generate them in various ways. An easy way to create a column vector is to transpose a row vector.

```
% Define column vector using semi-colon between elements
column_vector=[1;2;3;4;5]

% Use transpose operator (single quote ') on a row vector
column_vector=[1, 2, 3, 4, 5]'
column_vector=linspace(0,1)'
```

2. Matrices

2.1 Creating Matrices

Matrices are created similar to vectors. Row entries are separated by commas or spaces, while rows are separated by a semi-colon.

```
% Define a 3(rows) x 4(columns) matrix A
A=[1 2 3 4; 5 6 7 8; 9 10 11 12]

% Define matrix rows using Enter key after each row
A=[1 2 3 4
   5 6 7 8
   9 19 11 12]

% Define using colon operator
A=[1:4; 5:8; 9:12]
```

2.1.1 Common Matrices

Matrices where each entry is a zero or one are very common.

```
% Zero matrices
A=zeros(3,3)        % 3 x 3
A=zeros(3)          % 3 x 3
A=zeros(3,4)        % 3 x 4
A=zeros(5,2)        % 5 x 2

% Ones matrix
```

```
A=ones(3)           % 3 x 3
A=ones(3,3)         % 3 x 3
A=ones(3,4)         % 3 x 4
A=ones(5,2)         % 5 x 2
```

2.1.2 Random Matrices

```
% Random Matrices

A=rand(3)           % 3 x 3 uniformly distributed on (0,1)
A=rand(5,2)         % 5 x 2 uniformly distributed on (0,1)
A=randn(3)          % 3 x 3 normally distributed with mu=0, s=1
A=randi(10,5,2)     % 5 x 2 random integers between 0 and 10
A=randi([3,9],2,3)  % 2 x 3 random integer matrix between 3 and 9
```

2.2 Indexing Matrices

Each entry in a matrix is indexed by its row and column (in that order). The element $a_{2,3}$ refers to the value in the second row third column of a matrix A.

```
% Example
A=rand(5)           % Random 5 x 5 matrix
```

```
A = 5×5
    0.5557    0.7067    0.9879    0.6841    0.1544
    0.1844    0.5578    0.1704    0.4024    0.3813
    0.2120    0.3134    0.2578    0.9828    0.1611
    0.0773    0.1662    0.3968    0.4022    0.7581
    0.9138    0.6225    0.0740    0.6207    0.8711
```

```

A(2,3)              % Element in the 2nd row, 3rd column
```

```
ans = 0.1704
```

```
A(1,5)              % Element in 1st row, 5th column
```

```
ans = 0.1544
```

```
% ----------- %
```

2.2.1 More advanced referencing

Selecting multiple rows (and/or columns) can be accomplished by specifying which rows and columns inside parenthesis. We can also use colon operator to reference entire rows or columns as well as selecting a sequence of rows (or columns).

```
% Generate matrix
A=rand(5)
```

```
A = 5×5
    0.3508    0.5975    0.3596    0.1249    0.9569
    0.6855    0.3353    0.5583    0.0244    0.9357
    0.2941    0.2992    0.7425    0.2902    0.4579
    0.5306    0.4526    0.4243    0.3175    0.2405
    0.8324    0.4226    0.4294    0.6537    0.7639
```

```
% ---------------- %

% Select 1st and 3rd rows and 2nd column
A([1,3],2)
```

```
ans = 2×1
    0.5975
    0.2992
```

```
% Select rows 1 to 3 and columns 2 through 5
A(1:3,2:5)
```

```
ans = 3×4
    0.5975    0.3596    0.1249    0.9569
    0.3353    0.5583    0.0244    0.9357
    0.2992    0.7425    0.2902    0.4579
```

```
% Select row 3
A(3,:)              % Note use of : to indicate ALL columns
```

```
ans = 1×5
    0.2941    0.2992    0.7425    0.2902    0.4579
```

```
% Select column 2
A(:,2)               % Note use of : to indicate ALL rows
```

```
ans = 5×1
    0.5975
    0.3353
    0.2992
    0.4526
    0.4226
```

2.2.1.1 Linear Indexing

A powerful (and fast) indexing method is called **linear indexing**. MATLAB stores matrices in memory columnwise.

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

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

```
% Linear indexing
% A(1) = 1
% A(2) = 6
% A(3) = 11
% A(4) = 2
% A(5) = 7
% .
% .
% .

% We can list all elements by:
A(1:15)              % as a ROW
```

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

```
% or by
A(:)                % as a COLUMN
```

```
ans = 15×1
     1
     6
    11
     2
     7
    12
     3
     8
    13
     4
     ⋮
```

```
% END (last index)
A(1,end)            % returns the last entry in row 1
```

2.3 Dimensions

The dimensions, size, number of elements, and length of arrays frequently arises. Since a vector is a special cases of a matrix, these functions apply to both.

```
A=rand(5,3)
x=rand(1,7)


size(A)             % 5 x 3
numel(A)            % 15 = 3*5

size(x)             % 1 x 7
numel(x)            % 7


% Length works for vectors only.  Applying length to a matrix
% returns the length of a row
length(x)           % 7
```

```
% Often we want to use the number of rows and/or columns
[r,c]=size(A);
```

3. Scalar array operations

You can perform mathematical operations on an entire set of numbers all at once. For example, +, -, scale, and element-wise operations such as multiplication, division, and exponentiation.

```
x=randi(100,5,5)
```

```
x = 5×5
    17    61    46    83    11
    80    27     9    54    97
    32    66    23   100     1
    53    69    92     8    78
    17    75    16    45    82
```

```
y=randi(50,5,5)
```

```
y = 5×5
    44    22     7    43     4
     5    46    44    32    12
    20    10    29    18     7
    13    14    28    26    10
    41     8     8    21    12
```

```
z=randi(5,3,3)
```

```
z = 3×3
     3     5     2
     1     3     5
     5     3     2
```

```
%  Addition
x+y
```

```
ans = 5×5
```

```
    61    83    53   126    15
    85    73    53    86   109
    52    76    52   118     8
    66    83   120    34    88
    58    83    24    66    94
```

```
% Subtraction
x-y
```

```
ans = 5×5
   -27    39    39    40     7
    75   -19   -35    22    85
    12    56    -6    82    -6
    40    55    64   -18    68
   -24    67     8    24    70
```

```
% Scale
3*x
```

```
ans = 5×5
    51   183   138   249    33
   240    81    27   162   291
    96   198    69   300     3
   159   207   276    24   234
    51   225    48   135   246
```

```
% Element-wise scaling
x.*y
```

```
ans = 5×5
         748        1342         322        3569          44
         400        1242         396        1728        1164
         640         660         667        1800           7
         689         966        2576         208         780
         697         600         128         945         984
```

```
% Element-wise division
x./y
```

```
x./y
```

```
ans = 5×5
   0.386363636363636   2.772727272727273   6.571428571428571   1.930232558139535   2.750000000000000
  16.000000000000000   0.586956521739130   0.204545454545455   1.687500000000000   8.083333333333334
   1.600000000000000   6.600000000000000   0.793103448275862   5.555555555555555   0.142857142857143
   4.076923076923077   4.928571428571429   3.285714285714286   0.307692307692308   7.800000000000000
   0.414634146341463   9.375000000000000   2.000000000000000   2.142857142857143   6.833333333333333
```

```
% Element-wise exponentation
z.^2
```

```
ans = 3×3
     9    25     4
     1     9    25
    25     9     4
```

Return to MAT225