

A. Data Format

MATLAB can display output using many different formats. Here are the choices of format, as obtained by typing the command `help format`. The default format is `short`.

Command	Description
<code>format SHORT</code>	Scaled fixed point format with 5 digits.
<code>format LONG</code>	Scaled fixed point format with 15 digits for double and 7 digits for single.
<code>format SHORTE</code>	Floating point format with 5 digits.
<code>format LONGE</code>	Floating point format with 15 digits for double and 7 digits for single.
<code>format SHORTG</code>	Best of fixed or floating point format with 5 digits.
<code>format LONGG</code>	Best of fixed or floating point format with 15 digits for double and 7 digits for single.
<code>format SHORTENG</code>	Engineering format that has at least 5 digits and a power that is a multiple of three
<code>format LONGENG</code>	Engineering format that has exactly 16 significant digits and a power that is a multiple of three.
<code>format HEX</code>	Hexadecimal format.
<code>format +</code>	The symbols +, - and blank are printed for positive, negative and zero elements. Imaginary parts are ignored.
<code>format BANK</code>	Fixed format for dollars and cents.
<code>format RAT</code>	Approximation by ratio of small integers. Numbers with a large numerator or large denominator are replaced by *.
<code>format COMPACT</code>	Suppresses extra line-feeds.
<code>format LOOSE</code>	Puts the extra line-feeds back in.

B. One-dimensional arrays (vectors)

The basic MATLAB data type is the *array*. Try these commands:

```
>> p1 = [2 3 5 7 11]
>> p2 = [2, 3, 5, 7, 11]
>> q1 = [2; 3; 5; 7; 11]
>> q2 = [2
3
5
7
11]
```

If you have a long array, it is not efficient to type the entries individually. Try the following commands to see other ways to generate arrays and work with array contents.

```
>> a = -1 : 3 : 14
>> b = 10 : -1 : 0
>> c = linspace(0, 1, 11) % Generates an array of 11 numbers evenly spaced between 0 and 1
>> d = c' % Turns a row vector into a column vector
>> c(3) % Gives the 3rd entry in vector c
>> c(end) % Gives the last entry in c. This is useful if you do not know the length of c.
>> c([1 3 5]) % Gives the 1st, 3rd, and 5th entries in c
>> c(3) = -5 % Sets the value of the 3rd entry in c to -5
```

C. Two-dimensional arrays (matrices)

Try these commands:

```
>> A = [1, 2; 3, 4; 5, 6]
>> B = A' % B is the matrix whose rows are the columns of A
>> A(2, 1) % Gives the entry in the 2nd row and 1st column of A
```

D. Special matrices

Try these commands:

```
>> C = zeros(2, 3)
>> D = ones(3, 4)
>> E = eye(3)
```

E. Combining and indexing arrays

Try these commands:

```
>> B = [3*ones(3, 3) eye(3)]
>> C = [3*ones(3, 3); eye(3)]
>> D = [C C]
>> B(:, 4) % Gives the entries in the 4th column of B
>> B(:, 4:5) % Gives the entries in the 4th and 5th columns of B
>> B(2, :) % Gives the entries in the 2nd row of B
>> B(2:3, 2:4) % Gives the entries in rows 2 and 3 between columns 2 and 4 of B
>> B([1 3], [2 4]) % Gives the entries in rows 1 and 3, columns 2 and 4 of B
```

F. Deleting elements

```
>> p1(3) = [] % Deletes the 3rd element of vector p1
>> B(:, 6) = [] % Deletes the 6th column of matrix B
>> B(1:2, :) = [] % Deletes the 1st and 2nd rows of matrix B
```

G. Useful functions

```
>> length(p1)
>> size(B)
>> C = reshape(B, 2, 9) % Creates a matrix with 2 rows and 9 columns using the elements of B
>> D = diag(p1) % Creates a diagonal matrix with elements of p1 on the diagonal
>> v = diag(A) % Creates a vector from the diagonal elements of A
```

H. Strings

MATLAB can handle characters (letters, numerals, special characters, spaces) as well as numerical data. Try these commands:

```
>> a = 'Today is Monday.'
>> length(a)
>> a(7)
```

Practice Problems (from Gilat, *MATLAB: An Introduction with Applications*)

1. Create a column vector with the following elements:
 $\frac{32}{3 \cdot 2^2}$, $\sin^2(35^\circ)$, 6.1, $\ln(29^2)$, 0.00552, $\ln^2(29)$, and 133
2. Create a row vector with 9 equally spaced elements in which the first element is 81 and the last element is 12. **Do not type in all 9 elements. Use the linspace command.**
3. Create a vector named `vecA` that has 14 elements of which the first is 49, the increment is -3, and the last element is 10. Then, **using the colon symbol**, create a new vector named `vecB` that has 8 elements. The first 4 elements are the first 4 elements of `vecA`, and the last 4 are the last 4 elements of `vecA`.
4. Create the following matrix B .

$$\begin{bmatrix} 18 & 17 & 16 & 15 & 14 & 13 \\ 12 & 11 & 10 & 9 & 8 & 7 \\ 6 & 5 & 4 & 3 & 2 & 1 \end{bmatrix}$$

Use the matrix B to

- (a) Create a six-element column vector named `va` that contains the elements of the second and fifth columns of B .
 - (b) Create a seven-element column vector named `vb` that contains elements 3 through 6 of the third row of B and the elements of the second column of B .
 - (c) Create a nine-element column vector named `vc` that contains the elements of the second, fourth, and sixth columns of B .
5. Use the zeros, ones, and eye commands to create the following arrays.

(a) $\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$

(b) $\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

(c) $\begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$