Arrays

- The fundamental unit of data in MATLAB
- Scalars are also treated as arrays by MATLAB (1 row and 1 column).
- Row and column indices of an array start from 1.
- Arrays can be classified as vectors and matrices.

Variable assignment:

Variable_name = expression;

Basic Mathematics

- \( a + b \) → \( a + b \)
- \( a - b \) → \( a - b \)
- \( a \times b \) → \( a \times b \)
- \( a / b \) → \( a / b \)
- \( a^b \) → \( a^b \)

Array:

A collection of data values organized into rows and columns, and known by a single name.

Variables and Arrays

Array: A collection of data values organized into rows and columns, and known by a single name.

Array Dimensions

\[
\begin{bmatrix}
1 & 2 \\
3 & 4 \\
5 & 6
\end{bmatrix}
\]

3x2 matrix → 6 elements

\[
\begin{bmatrix}
1 & & \\
2 & & \\
3 & & \\
4 & &
\end{bmatrix}
\]

1x4 array → 4 elements, row vector

\[
\begin{bmatrix}
1 & \\
3 & \\
5 &
\end{bmatrix}
\]

3x1 array → 3 elements, column vector

\[
\begin{array}{c}
\text{Row} \\
1 \\
2 \\
3 \\
4
\end{array}
\]

\[
\begin{array}{c}
\text{Col} \\
1 \\
2 \\
3 \\
4 \\
5
\end{array}
\]

Array:

A collection of data values organized into rows and columns, and known by a single name.
Variables

- A region of memory containing an array, which is known by a **user-specified name**.
- Contents can be used or modified at any time.
- Variable names must begin with a letter, followed by any combination of letters, numbers and the underscore (_) character. Only the first 31 characters are significant.
- The MATLAB language is Case Sensitive. NAME, name and Name are all different variables.

Variable Types

double: 64-bit double-precision floating-point numbers

```matlab
>> var = f + i;
```

- char: 16-bit values, each representing a single character

The char arrays are used to hold character strings.

```matlab
>> comment = 'This is a character string';
```

The type of data assigned to a variable determines the type of variable that is created.

Initializing Variables

An assignment statement has the general form

```
var = expression
```

**Examples:**

- What is the result of each?

  ```matlab
  >> var = 40 * i;  >> a2 = [0 1+8];
  >> var2 = var / 5; >> b2 = [a2(2) 7 a];
  >> array = [1 2 3 4]; >> c2(2,3) = 5;
  >> x = 1; y = 2; >> d2 = [1 2];
  >> a = [3.4]; >> d2(4) = 4;
  >> b = [1.0 2.0 3.0 4.0];
  >> c = [1.0; 2.0; 3.0];
  >> d = [1, 2, 3; 4, 5, 6]
  >> e = [1, 2, 3 automatic echoing of values
  4, 5, 6];
  >> a2 = [0 1+8];
  >> b2 = [a2(2) 7 a];
  >> c2(2,3) = 5;
  >> d2 = [1 2];
  >> d2(4) = 4
  ```

Initializing with Shortcuts

**first: increment: last**

- **Colon operator:** a shortcut notation used to initialize arrays with thousands of elements

  ```matlab
  >> x = 1 : 2 : 10;
  >> angles = (0.01 : 0.01 : 1) * pi;
  ```

- **Transpose operator:** (') swaps the rows and columns of an array

  ```matlab
  >> f = [1:4];
  >> g = 1:4;
  >> h = [g' g'];
  ```

Arrays are constructed using brackets and semicolons.

- All of the elements of an array are listed in row order.
- The values in each row are listed from left to right and they are separated by blank spaces or commas.
- The rows are separated by semicolons or new lines.
- The number of elements in every row of an array must be the same.

- The expressions used to initialize arrays can include algebraic operations and all or portions of previously defined arrays.
Initializing with Built-in Functions

- `zeros(n)` >> `a = zeros(2);`
- `zeros(n,m)` >> `b = zeros(2, 3);`
- `zeros(size(arr))` >> `c = [1, 2, 3, 4];`
- `ones(n)`
- `ones(n,m)`
- `ones(size(arr))`
- `eye(n)`
- `eye(n,m)`
- `length(arr)`
- `size(arr)`

Initializing with Keyboard Input

- The `input` function displays a prompt string in the Command Window and then waits for the user to respond.

  ```matlab
  my_val = input('Enter an input value: ');
  in1 = input('Enter data: ');
  in2 = input('Enter data: ', 's');
  ```

Multidimensional Arrays

- A two-dimensional array with `m` rows and `n` columns will occupy `m`x`n` successive locations in the computer's memory.

  ```matlab
  a = [1 2 3; 4 5 6; 7 8 9; 10 11 12];
  ```

  ![Two-dimensional array example]

- For two-dimensional arrays, a colon can be used in a subscript to select all of the values of that subscript.

Subarrays

- The `end` function: When used in an array subscript, it returns the highest value taken on by that subscript.

  ```matlab
  arr3 = [1 2 3 4 5 6 7 8]
  arr3(5:end) is the array [___ ___ ___]
  ```

- Using subarrays on the left hand-side of an assignment statement:

  ```matlab
  arr4(1:2, [1 4]) = [20 21; 22 23]
  >> (1,1), (1,4), (2,1) and (2,4) are updated.
  arr4 = [20 21; 22 23]
  ```

  ![Subarray example]

Subarrays

- The `end` function: When used in an array subscript, it returns the highest value taken on by that subscript.

  ```matlab
  arr3 = [1 2 3 4 5 6 7 8]
  arr3(5:end) is the array [___ ___ ___]
  ```

- Using subarrays on the left hand-side of an assignment statement:

  ```matlab
  arr4(1:2, [1 4]) = [20 21; 22 23]
  >> (1,1), (1,4), (2,1) and (2,4) are updated.
  arr4 = [20 21; 22 23]
  ```

Subarrays

- Assigning a Scalar to a Subarray: A scalar value on the right-hand side of an assignment statement is copied into every element specified on the left-hand side.

  ```matlab
  >> arr4 = [1 2 3 4; 5 6 7 8; 9 10 11 12];
  >> arr4(1:2, 1:2) = 1
  arr4 =
  ```

  ![Subarray example]
Displaying Output

The `disp(array)` function

```matlab
>> disp('Hello')
Hello
>> disp(5)
5
>> disp(['Chocolate ' 'cake'])
Chocolate cake
>> name = 'Judy';
>> disp(['Hello ' name])
Hello Judy
```

for-loop & if-else

• Syntax is similar to Python

```matlab
for i = 1:5
    disp('Hello');
end

if i > 3
    disp('Hello World!')
else
    disp('Greetings All')
end
```

Using Functions

```matlab
function [out1, out2, ...] = funname(in1, in2)
    % defines function funname that accepts inputs in1, in2, etc. and returns outputs out1, out2, etc.
    % Put the following into an '.m' file and call it from the workspace

    function [mean, stdev] = stat(x)
        n = length(x);
        mean = sum(x)/n;
        stdev = sqrt(sum((x-mean).^2/n));
    end
```

Using the Debugger

• Place 'stop signs' by clicking in the left margin of the editor
• Use the 'Debug' menu and/or learn the key strokes to
  • Step
  • Step over / into
  • Run to cursor
• Check values in the workspace window
  • K >> prompt while the debugger is running

Matlab Recap and New Things

• Programming environment
  • Command area, editor
• Programming structures
  • if-else, for loop
  • functions: name via '.m' file; input; OUTPUT
• main() function needed?
  • '.m' files called script files
  • Typically, each function is in its OWN script file
  • Typically, no 'main()' function is used
• Accessing data arrays
  • Matlab: array(row, col)
  • Python: pict(x, y)
  • Where 'row' maps to 'y', and 'col' to 'x'
• New
  • Help files
    • In workspace vs. help window and internet
  • Plotting
2D Plotting

• We will make simple 2D plots of data, and fancier 3D plots

```matlab
>> t = (1:10)
>> out = 10*rand(1,10)
>> plot(t,out,'b.-')
>> out = 10*rand(1,10).*ones(1,10)
>> stem(t,out,'ro','linewidth',2)
```

Plotting with Subplots

• Multiple plots in one figure window

```matlab
>> x = -2*pi : 0.01 : 2*pi;
>> subplot(3, 1, 1)
>> plot(x, sin(x))
>> grid on
>> title('Sine, Cosine and Tangent')
>> ylabel('Amplitude') % Label each sub plot
>> axis([-6 6 -1 1]) % Set axes for each sub plot
>> subplot(3, 1, 2)
>> plot(x, cos(x))
>> subplot(3, 1, 3)
>> plot(x, tan(x))
```

3D Plotting

• Three dimensional plots

- Plot 3, Surface, Mesh

```matlab
>> t = 0:pi/50:10*pi;
>> plot3(sin(t),cos(t),t)
>> grid on
>> axis square

>> [X,Y,Z] = peaks(30);
>> surf(X,Y,Z)
>> colormap hsv
>> axis([-3 3 -3 3 -10 5])
```

```matlab
>> fh = @(x,y) x.*exp(-x.^2-y.^2);
>> ezmesh(fh,40);
>> colormap([0 0 1])
```
Types of Errors...

- Syntax errors
  - Check spelling and punctuation
- Run-time errors
  - Check input data
  - Can remove ';' or add "disp" statements
- Logical errors
  - Use shorter statements
  - Check typos
  - Check units
  - Ask your friends, assistants, instructor, …

Special Values

- \( \pi \): \( \pi \) value up to 15 significant digits
- \( i, j \): \( \sqrt{-1} \)
- \( \text{Inf} \): infinity (such as division by 0)
- \( \text{NaN} \): Not-a-Number (division of zero by zero)
- \( \text{clock} \): current date and time in the form of a 6-element row vector containing the year, month, day, hour, minute, and second
- \( \text{date} \): current date as a string such as 16-Feb-2004
- \( \text{eps} \): epsilon is the smallest difference between two numbers
- \( \text{ans} \): stores the result of an expression

Built-in Matlab Functions

- `result = function_name( input );`
  - `abs, sign`
  - `log, log10, log2`
  - `exp`
  - `sqrt`
  - `sin, cos, tan`
  - `asin, acos, atan`
  - `max, min`
  - `round, floor, ceil, fix`
  - `mod, rem`
  - `help elfun → help for elementary math functions`

Formatted Output

- `fprintf( format, data )`
  - `%d` integer
  - `%f` floating point format
  - `%e` exponential format
  - `%g` either floating point or exponential format, whichever is shorter
  - `%s` string
  - `
` new line character
  - `	` tab character

Formatted Output Examples:

- `>> fprintf( 'Result is %d', 3 )
  Result is 3`
- `>> fprintf( 'Area of a circle with radius %d is %f', 3, pi*3^2 )
  Area of a circle with radius 3 is 28.274334`
- `>> x = 5;
  >> fprintf( 'x = %3d', x )
  x = 5`
- `>> x = pi;
  >> fprintf( 'x = %0.2f', x )
  x = 3.14`
- `>> fprintf( 'x = %6.2f', x )
  x = 3.14`
- `>> fprintf( 'x = %d\ny = %d', 3, 13 )
  x = 3
  y = 13`

Commands Summary

- `help command → Online help`
- `lookfor keyword → Lists related commands`
- `clear → Clears the workspace`
- `clc → Clears the command window`
- `clf → Clears the active figure window`
- `diary filename → Sends output to file`
- `diary on/off → Turns diary on/off`
- `who, whos → Lists content of the workspace`
- `more on/off → Enables/disables paged output`
- `Ctrl+c → Aborts operation`
- `... → Continuation`
- `% → Comments`