Lecture 2: Matlab Fundamentals: computer representation, data classes, image display

Learning Objectives:
- Create matrices and perform basic operations in Matlab
- Complex number representation in Matlab
- Basic image display in Matlab

Assignment:
1. Read pages 1-9 in the Matlab Primer by Kermit Sigmon.

I. Defining matrices and basic operations in Matlab
a. Manual definition

There are several methods for data input into Matlab. The most direct is to manually define the elements of a matrix, A:

\[ A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \]

The output of the input on the line above is an M X N matrix where M = N = 3:

\[
A = \begin{bmatrix} 
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 
\end{bmatrix}
\]

Note the semicolon used to demarcate rows of the matrix A.

Note that Matlab uses the term “vector” to describe a 1 X n or n X 1 array of numbers.

One important advantage of Matlab is its ability to readily represent complex data. Define a new matrix A that has complex elements where we use multiplication by the imaginary number, i = sqrt(-1):

\[ A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + i \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \]

The output of the input on the line above is then:

\[
A = \begin{bmatrix} 
1.0000 + 5.0000i & 2.0000 + 6.0000i \\
3.0000 + 7.0000i & 4.0000 + 8.0000i 
\end{bmatrix}
\]

This example also illustrates matrix multiplication with the `*` operator. One can type “help `*`” at the Matlab command line to obtain specific definitions of fundamental math operators. Note that element by element multiplication is `.*` rather than `*`. For example, if we wanted to square the elements of matrix A, we would use:

\[ A.*A = \]

\[
\begin{bmatrix} 
-24.0000 & +10.0000i & -32.0000 & +24.0000i \\
-40.0000 & +42.0000i & -48.0000 & +64.0000i 
\end{bmatrix}
\]
Lecture 2: Matlab Fundamentals: computer representation, data classes, image display

Note that this is equivalent to:

\[ A.^2 = \]

\[
\begin{bmatrix}
-24.0000 +10.0000i & -32.0000 +24.0000i \\
-40.0000 +42.0000i & -48.0000 +64.0000i \\
\end{bmatrix}
\]

If we wanted to calculate the square of the matrix \( A \), we would use:

\[ A*A = \]

\[
\begin{bmatrix}
-60.0000 +42.0000i & -68.0000 +56.0000i \\
-76.0000 +74.0000i & -84.0000 +96.0000i \\
\end{bmatrix}
\]

So please be aware of this fundamental difference in meaning of the two operators.

Below is a partial output after using the ‘help’ command on the ‘**’ operator:

```
help '**'
Operators and special characters.

Arithmetic operators.
   plus    - Plus                           +
   uplus   - Unary plus                     +
   minus   - Minus                         -
   uminus  - Unary minus                   -
   mtimes  - Matrix multiply              *
   times   - Array multiply                .*
   mpower  - Matrix power                  ^
   power   - Array power                   .^ 
   mldivide - Backslash or left matrix divide    \
   mrdivide - Slash or right matrix divide   /
   ldivide  - Left array divide           .\ 
   rdivide  - Right array divide          ./
   kron     - Kronecker tensor product     kron
```

Since we are primarily interested in image acquisition and analysis in this class, consider the result of calling the graphic function image():

```
image(A)
??? Error using ==> image
Error using ==> image
Image CData can not be complex.
```

```
title('image(A)') %Figure 1a
figure;image(abs(A))
axis('image')
colorbar %Figure 1b
colormap('gray') %Figure 1c
brighten(0.5) %Figure 1d
```
help brighten

BRIGHTEN(BETA) brightens or darkens color map.

BRIGHTEN(BETA) replaces the current color map with a brighter
or darker map involving essentially the same colors. The map is
brighter if 0 < BETA <= 1 and darker if -1 <= BETA < 0.

BRIGHTEN(BETA), followed by BRIGHTEN(-BETA) restores the
original map.

MAP = BRIGHTEN(BETA) returns a brighter or darker version of the
color map currently being used without changing the display.

NEWMAP = BRIGHTEN(MAP,BETA) returns a brighter or darker version
of the specified color map without changing the display.

BRIGHTEN(FIG,BETA) brightens all the objects in the figure FIG.

% Note that brighten is changing the level, but not the window.
% Consider that Matlab’s image display functions are cumbersome and non-
% intuitive, especially for 3D. We have created a “Volume-Viewer” that integrates
these tools to make image display and visualization more intuitive. See website
to download.