GETTING STARTED WITH DATA: MATLAB

WiO Workshop
01.28.2016
Presenter: Kelsey Miller
MATLAB

- Multi-paradigm, numerical computing environment
- 4\textsuperscript{th} gen programming language
- Allows matrix manipulation, function and data plotting, algorithm implementation, user interface creation, and interfacing with other languages including 3\textsuperscript{rd} gen languages: C, C++, Java, Fortran and Python
In layman’s terms:

- Powerful, user-friendly tool for data analysis that plays well with other languages
- Loading data into MATLAB
- Organizing data in your workspace
- Using for loops to deal with large data sets
- Exporting your data
- load
- importdata
- xlsread
- imread
- fitsread

Loading Data into MATLAB
- **Function**: Loads variables from a file into the workspace (output in double)

- **Syntax**: 
  
  ```matlab
  load filename
  A = load('filename')
  A = load('filename', variables)
  A = load('filename', '-format')
  A = load('filename', '-format', variables)
  ```

- **Example**: 
  ```matlab```
  exampleload.m
  ```
**Function:** Loads variables from a file into the workspace. Can be used to load image files (output in uint8) and can read header information in ASCII file.

**Syntax:**
(paste from clipboard)

\[ A = \text{importdata}('\text{-pastespecial}') \]

(import from folder)

\[ A = \text{importdata}('filename') \]

\[ A = \text{importdata}('filename', 'delimiterIn') \]

\[ A = \text{importdata}('filename', 'delimiterIn', 'headerlinesIn') \]

**Example:**
exampleimportdata.m
- **Function:** Reads Microsoft Excel spreadsheet file (output in double).

- **Syntax:**
  
  \[
  A = \text{xlsread}('filename')
  \]
  
  \[
  A = \text{xlsread}('filename', \text{sheet})
  \]
  
  \[
  A = \text{xlsread}('filename', \text{range})
  \]
  
  \[
  A = \text{xlsread}('filename', \text{sheet}, \text{range})
  \]
  
  \[
  [\text{data, txt, raw}] = \text{xlsread}('filename', \text{sheet}, \text{range})
  \]

- **Example:**

  examplexlsread.m
- **Function:** Read an image from a graphics file (Formats supported: BMP, CUR, GIF, HDF4, ICO, JPEG, JPEG 2000, PBM, PCX, PGM, PNG, PPM, RAS, TIFF, XWD.) (Output type depends on file bit depth.)

- **Syntax:**

  \[
  A = \texttt{imread('filename')} \\
  A = \texttt{imread('filename', 'format')} \\
  \texttt{[A, map] = imread('filename')} \\
  \]

  (For information about graphics file)

  \[
  \texttt{info = imfinfo('filename')} \\
  \]

- **Example:**

  exampleimread.m
- **Function**: Reads data from a FITS file (Flexible Image Transport System)*

- **Syntax**:
  
  \[
  \text{data} = \text{fitsread('filename')}
  \]
  
  \[
  \text{info} = \text{fitsinfo('filename')}
  \]

- **Example**:
  
  examplefitsread.m

- Navigating the command window
- Calling/partitioning selected data
- Cell arrays
- Data cubes

Organizing Data
- Print working directory
  ```
  pwd
  ```
- Change directory
  ```
  cd directoryname
  cd ..
  ```
- Create new directory
  ```
  mkdir directoryname
  ```
- Remove directory
  ```
  rmdir directoryname
  ```
- Copy files
  \textit{copyfile} \textit{filename-to-be-copied} \textit{foldername-to-copy-to}

- Move files
  \textit{movefile} \textit{filename-to-be-moved} \textit{foldername-to-move-to}

- List files
  \textit{ls} OR \textit{dir} \textit{directoryname}

- Search specific file type
  \textit{ls} \textit{*.filetype}

- Delete files
  \textit{delete} \textit{filename}
- Calling specific column from matrix A
  \[ x = A( : , \text{column number}) \]

- Calling specific row from matrix A
  \[ x = A( \text{row number}, : ) \]

- Call specific value from matrix A
  \[ x = A( \text{row number}, \text{column number}) \]
Assigning data to new matrix B

- Create matrix B
  
  \[ B = \text{zeros (number of rows, number of columns)} \]

- Fill matrix B with data from matrix A
  
  \[ B(\ : , \ column \ number) = A(\ : , \ column \ number) \]

Example:

example_SelectingData.m
• Create a cell array to hold multiple matrices, images, etc. all of various sizes

\[ A = \text{cell}( \text{number of rows, number of columns} ) \]

Can create multi-dimensional arrays:

\[ A = \text{cell}( m, n, p, q, \ldots ) \]

• Fill cell array

Assign data to first cell in simple 2D array

\[ A\{ 1, 1 \} = \text{data} \]

• **Example:**

  example_CellArray.m
Create a ‘cube’ matrix to contain multiple data sets of the same size. NOTE: Not limited to 3 dimensions.

Create empty 3D data cube

\[ A = \text{zeros}(y, x, z) \]

Fill data cube

Assign data to first position in data cube

\[ A(:, :, 1) = \text{data} \]
Useful for group statistics and single image stats on large numbers of images when used in a for loop (covered in the following chapter)

Example Problem: Finding the mean value and standard deviation of a single pixel across multiple images

\[ M = \text{mean}(\text{DataCube}(\text{row num, column num, : })) \]
\[ S = \text{std}(\text{DataCube}(\text{row num, column num, : })) \]

Example:
```
example_DataCube.m
```
- Using *for* loops to load multiple files
- Using *for* loops in data analysis
Useful to use *for* loops when loading large number of files into your workspace.

Think ahead when coming up with your data naming scheme:

- Good example:
  
  `data_1.fits`
  
  `data_2.fits`
  
  ...

- Allows for easy implementation of a *for* loop in both loading and organizing your data files in your workspace.
Sample syntax:
Loading data into a Data Cube

\[ N = \text{number of files to be loaded} \]

\[ \text{DataCube} = \text{zeros(num rows, num cols, } N) \]

\[ \text{for } k = 1 : N \]

\[ \text{filename} = \text{horzcat('file',num2str(k),'.txt')} \]

\[ \text{DataCube}( :, :, k ) = \text{load(filename)} \]

end
Sample syntax: Loading data into a Cell Array

\[ N = \text{number of files to be loaded} \]
\[ \text{CellArray} = \text{cell}(1,N) \]

\[ \text{for } k = 1 : N \]
\[ \quad \text{filename} = \text{horzcat( 'file_', num2str(k), '.txt') } \]
\[ \quad \text{CellArray}\{k\} = \text{load( filename) } \]
\[ \quad \text{end} \]

Example:
example_LoadingMultipleFiles.m
- Using *for* loops to iterate through data cubes

- **Example Problem:** Finding the mean of every image stored in a data cube

\[
N = \text{number of files in data cube}
\]

\[
M_{\text{vector}} = \text{zeros(1,}\ N)\]

\[
\text{for } k = 1 : N
\]

\[
M_{\text{vector}}(k) = \text{mean2(DataCube(:, :, k))}
\]

\[
\text{end}
\]

- **Example:**

  example_DataCube.m
Exporting Data

Communicating with Zemax

- save
- fitswrite
- xlswrite
- saveas
- **Function**: Saves workspace variables to .mat file or ASCII

- **Syntax**:
  
  ```
  save filename
  save filename variables
  save('filename', 'variables', '-format')
  save('filename', 'variables', '-append')
  ```

- **Example**: examplesave.m
Function: Saves figure to specific file format. Can be saved as .m, FIG, JPEG, PNG, PDF, TIF, BMP, etc. (Saves figure to a .fig file if the file extension is unspecified.)

Syntax:
(current figure)

saveas(gcf, filename)

saveas(fig, filename)

saveas(fig, filename, formattype)

Example:
examplesaveas.m
**Function:** Writes data to Microsoft Excel spreadsheet file

**Syntax:**

```matlab
xlswrite('filename', variable)
xlswrite('filename', variable, sheet number, range)
xlswrite('filename', variable, sheet number)
xlswrite('filename', variable, range)
```

**Example:**

`examplexlswrite.m`
- **Function:** Writes an image to a FITS file

- **Syntax:**
  ```matlab
  fitswrite(imagedata, 'filename')
  ```

- **Example:**
  ```matlab
  examplefitswrite.m
  ```
- **Function:** Uses Zemax’s built in DDE (Dynamic Data Exchange) server to communicate with MATLAB as a client application.

- Can talk both ways between MATLAB and Zemax
  - Push lens data from MATLAB into the Zemax LDE (Lens Data Editor)
  - Pull data/plots from Zemax into the MATLAB workspace

- **Example:**
  ZemaxMatlabCom.m
- **Functionality:**
  - Atmosphere, Sun and Astronomy
  - Radiometry, Photometry, Spectral Filtering, Signal Chain and Processing
  - Transfer Functions: OTF, MTF, Spatial Filtering, etc.
  - Image Processing and Exploitation
  - Input and Output
  - Optical Modelling, Analysis and Testing
  - Surveillance
  - Internal Zemax Commands
  - Plotting/Miscellaneous
- Reference:

Examples URL:
wp.optics.arizona.edu/womeninoptics/
Workshops in MATLAB & Python

Offered by SOCk
Mid-to-end of February

- Covering MATLAB toolboxes and publication workflow (e.g. function handles, memory allocation, interfacing with LaTeX)
- Replacing both Mathematica and MATLAB with Python

But Wait..... There’s MORE!