Data Handling v7

Dr Amir-Homayoun Javadi
a.h.javadi@gmail.com
www.javadilab.com

Contents

Data Handling in MATLAB

General notes

To list the files, you can use `dir` function:

```
List = dir('*.mat')     % lists all the files with .mat extension
List = dir('Data_*.*')  % lists all the files beginning with 'Data_'
```

`List` will be a structure-array with multiple fields. For example:

```
List(2).name      % refers to the name of the 2nd file/folder
List(3).isdir     % indicates whether the 3rd file/folder is a
                  % file (zero) or a folder (one)
List(4).folder    % indicates the address of the 4th file/folder
```

To refer to files in different directories, you can use `fullfile` function. For example the line below lists all the files beginning with 'Data_' and file extension '.mat' in the folder 'Data, Sample'.

```
List = dir(fullfile('Data, Sample', 'Data_*.mat'))
```

Of course, if you want to access the files listed in array `List`, you need to include the folder address as well.

```
load(fullfile('Data, Sample', List(1).name))    % loads file List(1).name from
                                                % folder 'Data, Sample'
```

Native MATLAB data

To save and load MATLAB data you can easily use `save` and `load` functions:

```
save Data   % saves all the variables in the memory to the current folder
load Data   % loads all the variables in file Data.mat
```

If you want to specify specific variables to save or load, you can use the following:

```
save Data var1 var2 …
load Data var1 var2 …
```

A better way of saving and loading files is to use standard method of calling functions using parentheses:

```
save('Data', 'var1', 'var2')
load('Data', 'var1', 'var2')
```

Excel data

To save and load Excel files, you can use `xlsread` and `xlswrite` as follows:

```
xlswrite('filename.xlsx', 'sheet')                    % sheet is optional
[num, text, raw] = xlsread('filename.xlsx', 'sheet')  % sheet is optional
```

`num`, `text` and `raw` refer to numerical values only, text values only and all content, respectively. If you want to read the text content without numerical values you can use ~ sign as follows:

```
[~, text] = xlsread('filename.xlsx')
```

Using the same commands you can save and load .xls files. .xls files are for 97-2003 excel file formats and .xlsx files are for 2007 and later file formats.

One note, when you read and write files, make sure that they are not open elsewhere.

Data Handling in Python

General notes

To list the files, you can use `listdir` function from `os` library:

```
import os
List = os.listdir()     # lists all the files in this folder
```

Pay attention that `*` to indicate 'all' cannot be used for `os.listdir`. If you need, you can use the following

```
import glob
List = glob.glob('*.mat')
```

To refer to files in different directories, you can use `join` function from `os.path` library. For example, the line below lists all the files in folder 'Data, Sample', which is in folder 'Projects'.

```
List = os.listdir(os.path.join('Projects', 'Data, Sample')
```

Of course, if you want to access the files listed in array `List`, you need to include the folder address as well.

```
Data = sio.loadmat(os.path.join('Projects', 'Data, Sample', List[1]))
      # loads file List[1] from folder 'Data, Sample' in folder 'Projects'
```

If you want to include the address direction, you need to use `\\` for Windows and `/` for Mac operating systems.

```
Data = sio.loadmat('Data, Sample\\Data-011.mat')      # in Windows
Data = sio.loadmat('Data, Sample/Data-011.mat') # in Mac
```

You can add `r` (raw) prior to `'Data, Sample\\Data-011.mat'` to write `\` instead of `\\`

```
Data = sio.loadmat(r'Data, Sample\Data-011.mat')      # in Windows
```

For further information regarding `OS` functions, you can refer to this link.

No native Python data

Python does not have any native file format as MATLAB does. Therefore, you need to use different toolboxes to save data. There are quite a lot of options. The one that I would suggest you to use is saving and loading MATLAB data files. It has multiple advantages such as compatibility with MATLAB, benefitting from MATLAB well defined structures and data types. For this purpose, you need to use `scipy.io` library. The code below is to load a MATLAB file and access different variables.

```
import scipy.io as sio
Data = sio.loadmat('filename.mat')
var1 = Data['var1']
var2 = Data['var2']
```

To save data to a MATLAB file, you can use the following command:

```
sio.savemat('filename.mat', {'var1':var1, 'var2':var2})
```

One note, Data in the above example is a `dictionary` which contains all the variables in data file `filename.mat`. To access individual variables, you need to use the keys as shown above. To access all the keys, you can look at `Data.keys()`

Excel data

To read and write Excel data files you need to use two different libraries. To read an Excel file you need to use `xlrd` toolbox as below. First you need to make a link to the file, and then to the sheet to finally extract values from the file.

```
import xlrd
WB = xlrd.open_workbook('filename.xlsx')  # access to the file
WB.sheet_names()                    # lists all the sheets in the Excel file.
S = WB.sheet_by_name('sheet')       # access to the actual sheet
                                    # using the name of the sheet
S = WB.sheet_by_index(number)       # access to the actual sheet using the
                                    # index of the sheet. This index is based
                                    # on what is shown in WB.sheet_names()
                                    # number zero refers to the first sheet
```

So far, we have opened the Excel file and accessed the sheet. To actually get the data out you need to use the following:

```
S.row_values(i,col_b,col_e)   # values in row i, between columns col_b & col_e
S.row_values(i)               # returns all values in row i
S.col_values(j,row_b,row_e)   # values in column j, between rows row_b & row_e
S.col_values(j)               # returns all values in column j
S.cell_value(i, j)      # content of row i and column j
                        # pay attention that i and j begin from zero
S.nrows                 # number of rows
S.ncols                 # number of columns
```

To write to Excel files you need to use `xlsxwriter` toolbox:

```
import xlsxwriter as xlsw
WB = xlsw.Workbook('filename.xlsx')
                        # attention that Workbook is with capital W
                        # attention that the file extension must be .xlsx
S = WB.add_worksheet('sheet')
```

```
S.write(i, j, value)    # stores value in row i and column j.
WB.close()              # make sure that you close your Excel file at the end.
```

There are many alternatives to xlrd and xlsxwriter. See below for a collection:

- openpyxl toolbox, which has recently developed (link).
- Excel in Pandas toolbox; Pandas is a data analysis toolbox, which also supports Excel files (link, see also this link).

Text files

You can use numpy to read text files using loadtxt (link). See examples below:

```
Data = np.loadtxt('Sample-Text-Numerical.txt')
Data = np.loadtxt('Sample-Text-String.txt', dtype='str')
```

'Sample-Text-Numerical.txt' contains numerical values, and 'Sample-Text-String.txt' contains strings. Pay attention that you need to indicate the type of the content, in particular if it is string.

You can also use numpy to save text files using savetxt (link). See examples below:

```
x = np.arange(0, 5, 0.2)      # values between 0 and 5 (exclusive of 5)
                              # with steps of 0.2
np.savetxt('Test-Output.txt', x, delimiter=',', fmt='%.1f')
```