"Data import and export"

General remarks:

- Supported data types: See list from Matlab help for supported data types, for which specific functions are available. Other data types most probably have to be read and written with low-level I/O functions.
- The Matlab Import Wizard: If you need to import some complicated kind of data (in particular if you have to do it only once or few times), it is probably a good idea to try to use the graphical tool provided by the Matlab environment, the Matlab Import Wizard. Maybe you want to store the data in Matlab file format afterwards…
    - To open the Matlab Import Wizard select "import data" from the "file" menu in the matlab window.
    - Depending on the file format Matlab will ask you to specify e.g. delimitors in text files, header formats etc, if the output should be a structure or several variables…
- Specialized functions: If you want to stick with the file formats provided by other programs it is probably a better idea to use specialized functions (either provided by Matlab or written by yourself) for data import and export.
- Some nice commands for data handling:
    - *exist('filename.mat','file')* Returns 2 (!) if file exists in current directory(or in directory specified by path in name) and 0 if not
    - *whos –file filename.mat* displays variables contained in file in command window
    - *w=whos(' –file','filename.mat')* returns struct array with fields name, size, bytes, class of variables contained in file
    - *delete filename.ext* deletes file on disk
    - *copyfile(oldname, dirname)* copy file oldname to directory
    - *copyfile(oldname, newname)* copy file oldname to newname
    - *cd directory* changes the current working directory
    - *p=path* writes the current Matlab search path to string variable p
    - *path(path,'newpath')* adds the directory 'newpath' to the Matlab search path

Matlab data:

- .mat files: The regular way to save data with Matlab is to write the workspace content (or parts of it) to a .mat file.
    - Format: Matlab binary format, cannot be opened by other programs
    - Content: Variables from the workspace are saved with their names and content.
    - Platform compatibility: mat-files are not platform-independent, but are cross-platform compatible. Variables saved on one platform can be opened on other Matlab platforms without any special treatment.
    - Version compatibility: The binary format used for saving data changed from Matlab version 6 to version 7. Data saved with version 6 can be loaded into version 7 without additional treatment. If you want to save

variables with Matlab 7 and load them into Matlab 6 you have to use the *–v6* option for the save command (e.g *save filename variable –v6*).

- Saving and loading with graphical interface: When using the graphical interface to save data Matlab will open a save dialog box to choose the location where to save the file. The .mat extension is automatically added to the file name you specify.
    - Save the entire workspace:
        - File menu in Matlab desktop -> save as
        - Save button in workspace browser
    - Save selected variables:
        - Select variables in Workspace browser. To select multiple variables, use shift+click or ctrl+click. Then open the context menu with right click and choose save as.
    - Load:
        - Double click on a .mat or ASCII file in the current directory window
        - Use the "import data" icon in the Workspace browser
        - File menu in Matlab desktop -> import data
- Save command: The command to save workspace variables is save. By default, data is saved as .mat file, but save can also write ASCII files. save can be used in various ways:
    - *save* — saves workspace to matlab.mat
    - *save filename* — saves workspace to filename.mat (extension .mat is added by Matlab if not specified, filename can contain path information)
    - *save filename var1 var2* — saves specified variables to filename.mat
    - *save filename options* — saves workspace to filename.mat with options. For a list of options (including use of regular expressions, saving structure fields as separate variables, Matlab version information…), please refer to the help page.
    - *save filename var1 var2 options* — saves specified variables with options
    - *save('filename','var1',var2',…)* — more convenient way to save if you generate filenames or variabe namens as strings. E.g. *fn='myfile'; save(fn,'M')*
- Load command: The *load* command is used to load Matlab variables or ASCII files into the workspace. Data contained in ASCII files will be stored in a variable with the same name as the file. If variables with the same names as the newly loaded already exist in the workspace they will be overwritten. *load* also comes in several flavors:
    - *load* — loads all variables saved in matlab.mat into workspace
    - *load filename* — loads all saved variables saved in filename.mat into workspace (extension .mat is first assumed, if the file has a different extension, Matlab will try to open it as ASCII file).
    - *load –ascii filename* — forces load to treat the file as ascii,

regardless of the extension. If the file is not numeric text, load will return an error.

- *load –mat filename* forces load to treat the file as mat-file, regardless of the extension. If the file is not in matlab binary format, load will return an error.
- *load filename var1 var2* loads specified variables into workspace
- *load('filename','var1',var2')* more convenient way to load with automatically generated file or variable names. E.g.

```
for i=1:N
        fn=sprintf('data%d',i);
        load(fn)
end
```

- *s=load('filename','var1',var2')* loads variables var1 and var2 into a structure variable s with fields var1 and var2. Good way to prevent overwriting of workspace variables.

Formatted text:

- Load: The normal *load* command can read ASCII data into Matlab, as long as the file is organized as a rectangular table of numbers, with each number in a row separated by a blank, comma, or tab character, and with an equal number of elements in each row. In this case, the numbers will be assigned to a variable with the name of the file name (if the filename starts with a digit, matlab will add a preceding X, all nonalpabetic characters will be replaced by underscores). You can use the option *–ascii* to force Matlab to read a file as ASCII, but Matlab will try to do so anyway if the file name does not have the ending .mat.
- Save: Matlab will save the content of a variable in an ASCII-file if the save command is used with the option *–ascii*.
- Specialized functions to read formatted text:
  - *dlmread* reads numeric data from the ASCII-delimited file filename, using the specified delimiter. Use *\t* to specify a tab delimiter.
    M = dlmread(filename, delimiter)
  - *textread* reads formatted text from file into several output variables. For format options see help
    [A,B,C,...] = textread('filename','format')
  - *textscan* reads formatted text from file into cell array after *fopen.* For format options see help
    C = textscan(fid, 'format')
  - *fscanf* reads data from file after *fopen* (see below)
  - *fgetl* reads text line-by-line after *fopen*, including terminator symbol (see below)
  - *fgets* reads text line-by-line after *fopen*, without terminator symbol (see below)

Graphics:

- Matlab figures: Matlab saves figures by default as Matlab figures with the ending .fig. Unfortunately, many programs use the .fig postfix for different

formats and (as far as I know) no other program can directly use Matlab figures in a reasonable way. So you have to choose a different format if you want to use figures for different contexts. However, it makes sense to save .fig files of figures if you want to continue your work on a figure later. All properties of the figure will be stored and if only parts of a data set are plotted the full data set will still be available after saving and loading as .fig files.

- Exporting graphics from Matlab figures: Possible either by selecting "export" on the "file" menu of the figure window or by using the function *saveas*. Will be covered in the graphics session
- Importing graphics to a matrix**:** function *imread* can read multiple graphics formats (color or grayscale), including jpeg, tiff, gif, bmp, png, hdf, xwd. Standard: *A = imread(filename, fmt)* with text string *fmt* specifying the file format with its standard file extension (e.g. *'gif'*) returns array *A*. If the image is a gray scale picture, A is a MxN array, if it is a true color picture, A is a MxNx3 array (for rgb) , if it is a tiff picture, A is a MxNx4 array (CMYK). Refer to help page for more information and options.
- Exporting graphics from a matrix: function *imwrite* can write multiple graphics formats. See help for usage, formats and format specific parameters. Standard: *imwrite(A,filename,fmt)* with text string *fmt* specifying the file format with its standard file extension (e.g. 'gif')
- Information about a graphics file: *info = imfinfo(filename)* returns a structure, *info*, whose fields contain information about an image in a graphics file. See help for information contained in the structure.

Sound:

- Microsoft WAVE .wav files:
  - wavread: Reads Microsoft WAVE .wav sound files. Standard: *[y,Fs,nbits] =wavread('filename.wav')* returns y: sampled data, Fs: sample rate in Hz, nbits: number of bits per sample used to encode the data in the file. For more options please refer to the help page
  - wavwrite: Writes Microsoft Wave .wav files. Standard. *wavwrite(y,FS,nbits,'filename.wav')* with y: sampled data, Fs: sample rate in Hz, nbits: number of bits used for encoding (standard is 16, values 8, 24, 32 also possible). Amplitude values outside the range [-1,+1] are clipped prior to writing.

Video:

- Audio/Video Interleaved .avi file:
  - aviread: *mov = aviread(filename)* reads the AVI movie filename into the MATLAB movie structure *mov*. If filename does not include an extension, then .avi is used. Use the movie function to view the movie *mov*. The movie structure has two fields, *cdata* and *colormap*.
  - avifile: Creates AVI file (see help for usage)
  - movie2avi: Creates AVI file from Matlab movie (see help for usage)
  - movie(M): Play recorded movie frames (see help for usage and options)
  - im2frame: *f=im2frame(x,map)* converts image x and colormap map into a movie frame
  - frame2im: *[x,map]=frame2im(F)* converts movie frame to image

Compatibility with Microsoft programs:

- Excel spreadsheet files:
  - Read Excel spreadsheet files: use the function *xlsread*. Standard usage *num=xlsread('filename.xls')* returns numeric data in double array *num*. Spreadsheet cells filled with text will be replaced by *NaN*. For more options please refer to the help page.
  - Write Excel spreadsheet files: use the function *xlswrite*. Standard usage *xlswrite('filename.xls',M)* will write the content of Matrix or cell array M to the first worksheet in the xls file, starting at cell A1. For more options please refer to the help page.
  - Test if a file is an Excel file: typ=xlsinfo('filename') returns the string 'Micorsoft Excel Spreadsheet' if the file specified by filename is an XLS file that can be read by the MATLAB xlsread function. Otherwise, typ is the empty string, (' '). For more options please refer to the help page.
- Data files generated with word: Save the data as text file with .txt and try to read as ASCII, but in general: better use some other program…
- Windows paintbrush pcx files: can be read in with imread

Low-level data input and output:

- Steps of low-level data input and output:
  1. Open the file with fopen
  2. Read or write
     a. binary data with *fread* or with *fwrite*
     b. formatted ascii data with *fscanf* or with *fprintf*
     c. read line-by-line from a text file with *fgets* or *fgetl*
  3. Close the file with fclose
- Opening and closing:
  - *fid=fopen('filename','permission')* Opens file. Values for permission: 'r' read only, 'w' write only, 'a' append only, 'r+' read and write. For binary files a 'b' must be appended, e.g. 'rb'. If successful, *fopen* returns the file identifier to the variable *fid.* If it fails, *fid* is -1 (you should test for this!)
  - *status=fclose(fid)* Closes file
- Reading and writing:
  - *A=fread(fid)* reads a binary file and stores it in a matrix. Each byte of input is used as next element of the matrix. (For options to read only part of the file or to specify data types with different encodings see help)
  - *count=fwrite(fid,M)* writes values in Matrix M to opened file in column order. The number of elements written successfully is returned to *count*. For more options see help.
  - *A = fscanf(fid, format)* reads data from the file specified by fid, converts it according to the specified format string, and returns it in matrix A.

- *count = fprintf(fid, format, A, ...)* formats the data in the real part of matrix A (and in any additional matrix arguments) under control of the specified format string, and writes it to the file associated with file identifier fid. fprintf returns a count of the number of bytes written.
- *tline = fgets(fid)* returns the next line of the file including line terminator symbol. If *fgets* encounters the end-of-file indicator, it returns -1.
- *tline = fgetl(fid)* returns the next line of the file without line terminator symbol. If *fgetl* encounters the end-of-file indicator, it returns -1

- Formats used for fscanf, fprintf: The format string for *fscanf* consists of the initial character %, an optional field width and a conversion character, e.g. %12e. To skip reading, the field width is *. For fprintf, the format string is e.g. %-12.5e with initial character %, an optional flag for positioning (-, +, 0 or whitespace, see help), optional field width and optional positions after decimal point, and a conversion character. Conversion characters for both functions are:
  - *%c* Sequence of characters; number specified by field width
  - *%d* Base 10 integers
  - *%e, %f, %g* Floating-point numbers
  - *%i* Defaults to base 10 integers. Data starting with 0 is read as base 8. Data starting with 0x or 0X is read as base 16.
  - *%o* Signed octal integer
  - *%s* A series of non-white-space characters
  - *%u* Signed decimal integer
  - *%x* Signed hexadecimal integer
- Controlling position in a file:
  - Controlling position in a file: Once a file is opened, Matlab will maintain a file position indicator that specifies a particular location within a file. MATLAB uses the file position indicator to determine where in the file the next read or write operation will begin.
  - *message=ferror(fid)* Tests file I/O status and returns error string message
  - *tf=feof(fid)* Tests for end of file
  - *status=fseek(fid,offset,origin)* Sets file position indicator. *offset* is positive or negative offset value in bits. *origin* can be: ‘bof’ beginning of file, ‘eof’ end of file, ‘cof’ current position in file. Returns –1 if operation fails.
  - *position=ftell(fid)* Gets file position indicator
  - *frewind(fid)* Rewinds file

Homework:

- If you have not already done so: Try to read in the data relevant to you and structure it in a convenient way.