

Lab Tools User Guide

Revised August 15, 2014

Table of Contents:

1. Introduction

1.1 Overview of object oriented programming	2
1.2 Downloading the Lab Tools Matlab code.....	2
1.3 Setup and personalization.....	3

2. Using Lab Tools

2.1 Step one: c3d2mat.....	5
2.2 Step two: ReviewEventsGUI.....	5
2.3 Subject file structure and usage.....	7
2.4 Data visualization and plotting functions	9
Appendix.....	11

1. Introduction

The lab tools package was designed specifically for use by students on the Motor Adaptation and Rehabilitation side of the Human Movement Research Lab. After data is collected in the lab, Vicon can export data files in the 'c3d' format. The purpose of the Lab Tools package is to convert these .c3d files into a form that is easier to work with and visualize within MATLAB®. This user guide is intended to orient new students to the functionality of the MATLAB® codes included in the package. A basic understanding of MATLAB® coding and command window use is assumed.

1.1 Overview of Object Oriented Programming

The lab tools code takes advantage of a feature of the MATLAB® language (which is a feature of many coding languages) called 'object oriented programming'. The basic idea behind object oriented programming is that data can be represented in a structured format that is pre-defined and which has pre-defined functions that can be performed on the data. Objects in MATLAB® are very similar to structures in appearance and usage, but offer many advantages over structures such as ability to control the fields of the structure, the values that each field can be set to, and operations that can be performed on the fields. Several examples on how to use the objects that are created by the lab tools code will follow, but if you would like to have a more in-depth understanding of object oriented programming in MATLAB®, refer to the documentation found here:

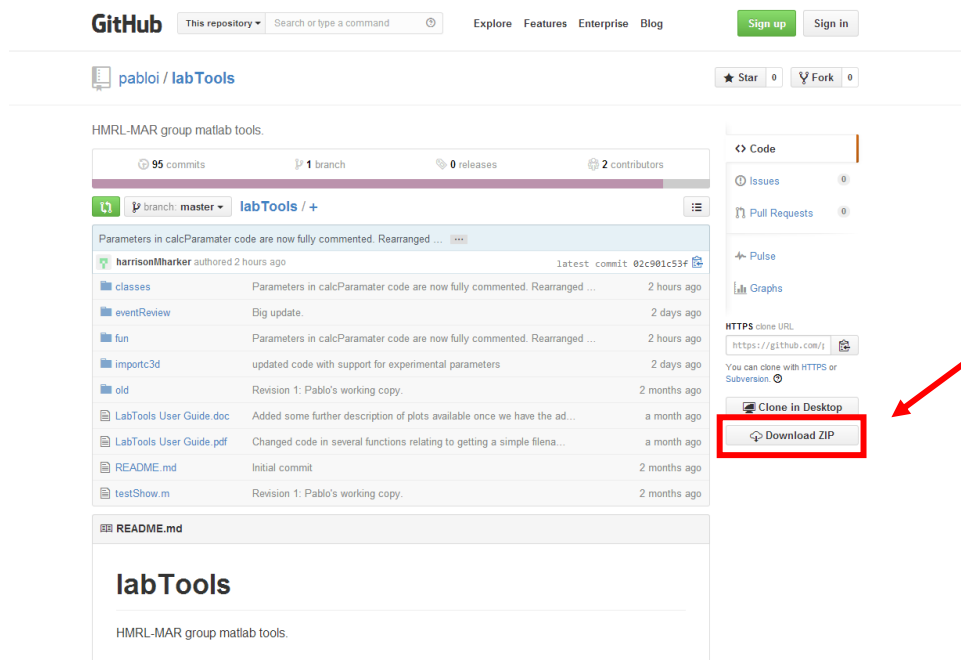
http://www.mathworks.com/help/pdf_doc/matlab/matlab_oop.pdf

1.2 Downloading Lab Tools MATLAB® code

To download the latest version of the Lab Tools package, go to

<https://github.com/pabloi/labTools.git>

and click on the 'download ZIP' button

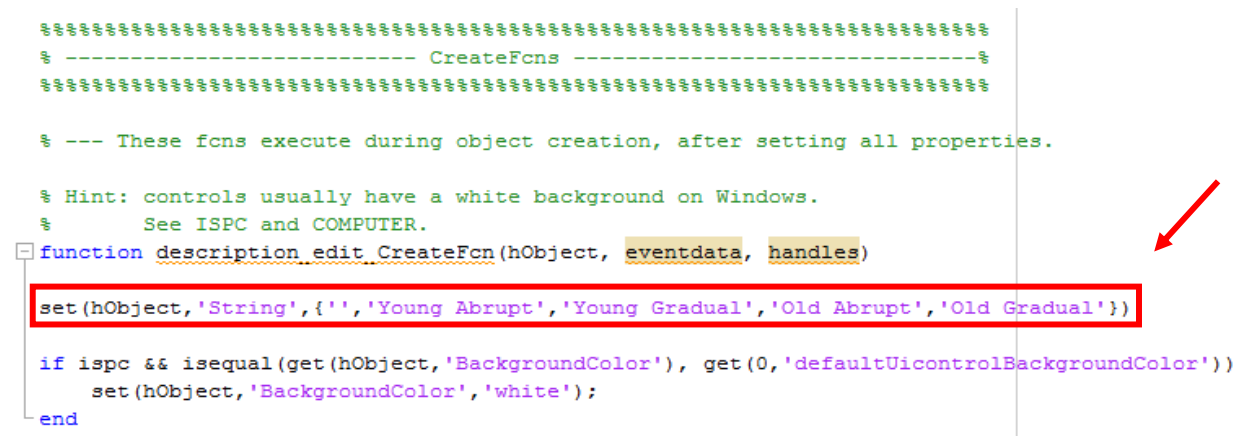


Unzip the file that was downloaded and copy the contents to whichever folder you choose. All of the folders and subfolders that were included in the download should be added to your MATLAB® path (see appendix if unsure how to do this). Additionally, you should download the Biomechanics Toolkit and add those folders/subfolders to your MATLAB® path as well, if you have not already done so (see appendix).

1.3 Lab Tools Set Up and Personalization

Before any data can be processed, there are a few setup steps to perform first:

1. Type 'c3d2mat' into the MATLAB® command window. A GUI should appear which allows you to enter information about the experiment, the subject, and each of the conditions tested. The very first drop-down list is where an experiment description can be selected. Select one of the descriptions and observe how the fields in the 'Trial Info' section of the GUI automatically fill in. If the experiment you are performing is not included on the list, you can add your own experiment types to this list by following steps 2-4. You can exit out of the GUI now (ignore the error message that appears).
2. Type 'edit GetInfoGUI' into the command window. Now use ctrl+f to find 'CreateFcns' or just scroll down to the part of the code that contains the 'CreateFcns'. The first function should be the 'description_edit_CreateFcn'. Here you will want to add the experiment descriptions that you will be using. Simply replace or append the existing descriptions with your experiment descriptions in the set(hObject,'String',...) command. Make sure that the first element in the cell is a blank string (i.e. ''). Make a record of the descriptions you add and then save and close out of GetInfoGUI.m.



```

#####
% ----- CreateFcns -----
#####

% --- These fcns execute during object creation, after setting all properties.

% Hint: controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
function description_edit_CreateFcn(hObject, eventdata, handles)

set(hObject,'String',{'','Young Abrupt','Young Gradual','Old Abrupt','Old Gradual'})

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

3. Now, type 'edit experimentDetails' into the command window. This is where you will define the condition numbers, names, descriptions, and types along with the expected trial numbers that belong to each condition (note that all of these can be edited in the GUI, so don't worry about small variations). First you will need to add a 'case' for each of the experiment descriptions you added in step 2. If two descriptions follow the same paradigm, then include both descriptions for a single case.

```

switch expDescrip
    case {'Old Abrupt','Young Abrupt'}

```

Second, add the condition numbers to the 'for condition =' command and edit the number of conditions in the 'set' command.

ex:

```
%condition numbers
i=1;
for cond = [1 2 4:10]
    eval(['set(handles.condition',num2str(i),','string',' ',num2str(cond),')'])
    i=i+1;
end
set(handles.numofconds,'string','9')
```

Finally, the names, descriptions, trial numbers, and type for each condition can be set. (Note that the numbers in the set commands DO NOT correspond to the condition number. The set command numbers should always be 1,2,3,4... whereas the conditions are allowed to skip a number.) The *condNames* should be simple and (if possible) the same as what other students in the lab are using to name their conditions. The *descriptions* can contain whatever information seems important to someone who is unfamiliar with what you did, such as the number of strides and possibly the speeds the belts were set at. The *trialnums* should be the expected numbers for that condition (multiple numbers can be set as '1:5', '1,2,3' or '4 5' but not '1-5'). The *type* strings indicate things such as over grounds walking, treadmill walking, incline walking, etc. IMPORTANT: The baseline conditions that will later be used to remove subject-specific biases need to be named such that one of the type strings and the string 'base' are both included in the *condName*. Any condition that is of the same type will have that particular baseline tendency removed from the corresponding data. For example, all conditions of the type 'OG' will have the bias from the condition named 'OG base' removed.

4. Type 'c3d2mat' again and select each of your descriptions to make sure the fields populate correctly. Fix any errors that you encounter.
5. The last thing you want to make sure to do before processing any of your data is to check that the marker labels from Vicon will be recognized by the processing functions in labTools. Certain markers must have the correct labels in order for the codes to work, and these are set in the 'findLabel' function. Type 'edit findLabel' into the command window to see which markers these are, and add any labels that aren't included to the appropriate case.

You are now ready to process your first subject!

2. Using the Lab Tools Package

2.1 Step One: c3d2mat

This is the first step in processing any subject. Run c3d2mat and this time fill in all the information in the GUI. **Note:** that if no folder is selected in the '.c3d file location' or the 'Subject file save location', the present working directory (aka the current folder) is assumed.

Once the information is filled in, hit 'OK' and you will be asked if there are any observations for individual trials. These are things like 'left ankle marker fell off around step 100'. If you type 'y' for yes, then you will be able to select each trial for which you have an observation. When complete, simply hit 'Done' at the end of the menu. MATLAB® will do the rest from here, so just pay attention to any of the warnings that pop up until the script is finished running.

When complete, you should see three new files in the folder that was selected as the save location. They should be named (Subject ID)RAW.mat, (Subject ID).mat, and (Subject ID)info.mat where (Subject ID) is the ID that was given to the subject in the GUI. The RAW data file contains only the raw data collected from VICON which are things like marker positions and force plate readings. The regular .mat file contains the raw data as well, but also contains things such as heel strike and toe off events, calculated limb angles, processed EMG signals, and calculated belt speeds. The info file contains the information entered in the GetInfoGUI and is the file that would be selected when using the 'load existing' button within the GUI.

2.2 Step Two: ReviewEventsGUI

When you type 'ReviewEventsGUI' in the command window, a blank version of the figure shown on the next page should pop up. Take a moment to acquaint yourself to all the features of this GUI, and try to load a subject who has already been run through c3d2mat. Choose some of the different fields in the 'Top plot' and 'bottom plot' panels to see what the data looks like. Once you are comfortable using the GUI, you can proceed to review your data to make sure there are no major problems such as drop-outs or mislabeling with the important markers (especially the hips and ankles). An easy way to do this is by checking the angle data since angles are composed of at least two markers and therefore more markers can be reviewed at a time. The 'adaptParams' are what matter in the end, so several of those could also be viewed during the review process. Viewing the max time window also makes this process easier.

If you do find a problematic trial, re-processed that trial in Vicon, replace the old .c3d file with the new, run c3d2mat again (but this time press 'Load Existing' and look for the (SubjectID)info.mat file). The information should all be the same, so you can just hit 'OK' and a new subject file will be generated.

Once you have confirmed that all the marker data is correct, the events themselves should be reviewed to confirm that they correspond to actual heel strike and toe off events.

The ReviewEventsGUI:

The screenshot shows the ReviewEventsGUI interface. At the top, there's a 'Directory' section with a text field containing '/' and a 'Browse' button. Below that is a 'Filename' dropdown menu showing 'OG91'. Underneath is a 'Condition' dropdown menu showing 'OG base'. Below that is a 'Trial' dropdown menu showing '3'. To the right of these is a 'Time Window' section with a vertical slider ranging from 1 to 78, with a checkbox checked at the top. Below the slider are two buttons: '<' and '>'. To the left of the slider is a 'Plot' button. Below the 'Time Window' section is an 'Edit events' section. It contains four buttons: 'Delete Indiv.', 'Delete Range', 'Save', and 'Write to disk'. To the right of these buttons is an 'Add events' section with an 'Add' button and a 'Pop-u...' dropdown menu. At the bottom, there are two plot sections: 'Top Plot' and 'Bottom Plot'. Each has a 'Data type:' dropdown menu and a 'Field:' dropdown menu. The 'Top Plot' 'Data type:' dropdown shows 'angleD...' and the 'Field:' dropdown is empty. The 'Bottom Plot' 'Data type:' dropdown shows 'adaptP...' and the 'Field:' dropdown shows 'good'.

Select the directory where the files of the subject under review are located (if it is not the present working directory)

Select the subject file from the dropdown list. (It should be the plain subject ID file and not the RAW or info files)

The condition can be specified, but will automatically change if using the forward and backward buttons.

The trial within the condition can also be specified, but once again will automatically change when going back and forward.

Choose the number of seconds you want to look at. If you always want to look at the full trial, click the check box near the top arrow button. This will ensure the full trial is visible even when changing which trial is being shown. Note that the box should be unchecked when choosing a smaller time window.

The plot button is actually rarely used, but is sometimes needed to refresh the plots such as when zooming in/out on the graphs.

The back/next buttons can be used to advance frames within a trial, and will automatically advance to the next trial/condition when reaching the end.

Activates a drop-down list which allows you to choose which event you would like to add. Once an event is chosen, a crosshair will appear which allows you to add the event by clicking on the graph

Brings up a crosshair which can be used to select the event(s) that need deleted. Pressing the return (enter) key after clicking on the event(s) will actually delete them.

Brings up a crosshair which allows you to click twice: the first click marks the start and the second click the end of a range of events to be deleted. **All** events in this range are removed.

If you are confident with the events after deleting/adding wherever necessary, you can hit 'save' which activates the 'Write to disk' button. Clicking this button will overwrite the subject file with a new one that has the updated events. A new file will also be generated called (Subject ID)params.mat (explained in section 2.4)

The Top Plot and Bottom Plot drop down lists allow you to choose what data you want to view. Any data that has both an 'R' and 'L' component are plotted simultaneously as well as any adaptParams that have both a 'Fast' and 'Slow' version.

2.3 Subject File Structure and Usage

A good way to get acquainted to the organization of the subject files are by loading one of the files, double-clicking the variable that appears in the 'Workspace' window, and then viewing the variable's contents in the 'Variables' window.

For example:

The screenshot shows the MATLAB Variables window with three panels. The first panel, titled 'expData', shows a 1x1 'experimentData' struct with properties: metaData (1x1 experimentMetaData), subData (1x1 subjectData), data (1x23 cell), isRaw (0), isProcessed (1), isStepped (0), and fastLeg ('R'). The second panel, titled 'expData.metaData', shows a 1x1 'experimentMetaData' struct with properties: ID ('Old Abrupt'), date (1x1 labDate), experimenter ('Carly Sombric'), observations (4x19 char), conditionName (1x10 cell), conditionDescripti... (1x10 cell), trialsInCondition (1x10 cell), and Ntrials (23). The third panel, titled 'expData.metaData.date', shows a 1x1 'labDate' struct with properties: day (25), month (7), and year (2014). Red circles and arrows highlight the path from expData to expData.metaData to expData.metaData.date.

You'll notice that the data is organized almost identically to how values are stored in a MATLAB struct variable. Data values are also access in an identical matter. For example, to save the year that the experiment took place in the example above to the variable 'expYear', you could type

```
expYear=expData.metaData.date.year;
```

Likewise, to save the name of condition 7 to 'cond7name' type:

```
cond7name=expData.metaData.conditionName{7};
```

This method of accessing data can be tedious however, especially when considering accessing the data in the example below:

The screenshot shows the MATLAB Variables window with four panels. The first panel, titled 'expData', shows a 1x1 'experimentData' struct. The second panel, titled 'expData.data', shows a 1x23 'cell' array. The third panel, titled 'expData.data(1,4)', shows a 1x1 'processedTrialData' struct. The fourth panel, titled 'expData.data(1,4).markerData', shows a 1x1 'double orientedLabTimeSeries' struct. The fifth panel, titled 'expData.data(1,4).markerData.labels', shows a 1x54 'cell' array. Red circles and arrows highlight the path from expData to expData.data to expData.data(1,4) to expData.data(1,4).markerData to expData.data(1,4).markerData.labels. The 'markerData.labels' panel shows a table with columns 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54. The table contains numerical data for each column.

And that's where object oriented programming comes in handy. Instead of having to look up what labels correspond to each column of the Data variable and which times correspond to each row, there are pre-defined methods in place to retrieve this information for you. Another method to access the value above is this:

```

Command Window
>> expData.data{4}.getMarkerList

ans =

Columns 1 through 8

    'LASIx'    'LASIy'    'LASIz'    'RASIx'    'RASIy'    'RASIz'    'LPSIx'    'LPSIy'

Columns 9 through 16

    'LPSIz'    'RPSIx'    'RPSIy'    'RPSIz'    'LTHIx'    'LTHIy'    'LTHIz'    'LKNEx'

Columns 17 through 24

    'LKNEy'    'LKNEz'    'LTIBx'    'LTIBy'    'LTIBz'    'LANKx'    'LANKy'    'LANKz'

Columns 25 through 32

    'LHEEx'    'LHEEy'    'LHEEz'    'LTOEx'    'LTOEy'    'LTOEz'    'RTHIx'    'RTHIy'

Columns 33 through 40

    'RTHIz'    'RKNEx'    'RKNEy'    'RKNEz'    'RTIBx'    'RTIBy'    'RTIBz'    'RANKx'

Columns 41 through 48

    'RANKy'    'RANKz'    'RHEEx'    'RHEEy'    'RHEEz'    'RTOEx'    'RTOEy'    'RTOEz'

Columns 49 through 54

    'RHIPx'    'RHIPy'    'RHIPz'    'LHIPx'    'LHIPy'    'LHIPz'

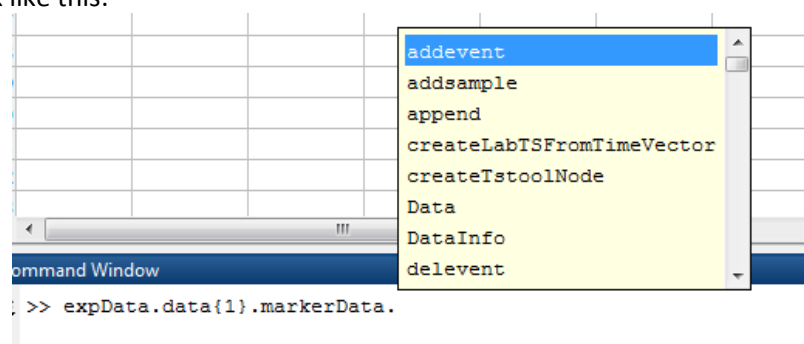
>> LankTS=expData.data{4}.markerData.getDataAsTS('LANKy');
>> LankTS.getSample(60)

ans =

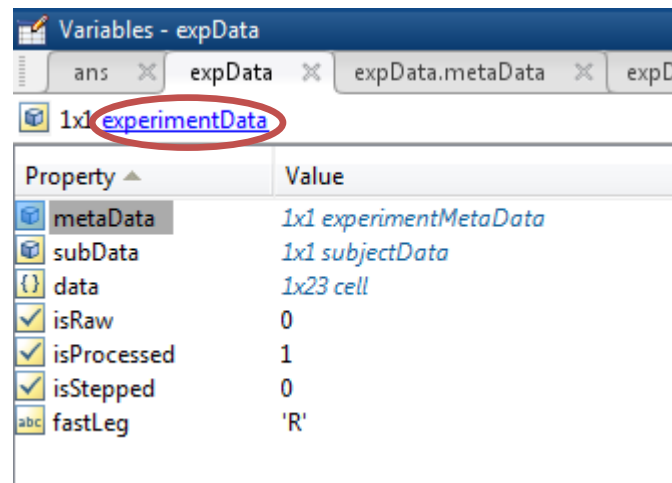
    582.4246

```

This may seem strange at first since, even though it appears you are trying to access a field of the structure, you are in reality calling a function (called a 'method'). While the methods that are available to you are not immediately obvious from inspecting the variable in the 'Variables' window, you can view them by typing the variable name in the command window followed by a period and the tab key. The result should look like this:



Note that the list that appears includes both variable values and methods available. For instance, selecting the option 'Data' would access an array of data values whereas selecting 'createLabTSFromTimeVector' calls a function that requires user input. To view a list that distinguishes between all the properties (fields) and methods (functions) available in any particular class of data, click on the class in the 'Variables' window.



Alternatively, the 'help' function can be used by typing `'help (class name).(field)'` where (class name) is something like 'experimentData' and (field) is one of the properties or methods. In general, methods should be able to be distinguished simply by their name in that most methods begin with something along the lines of 'get', 'make', or 'plot'.

2.4 Data Visualization and Plotting Functions

In section 2.2, the ReviewEventsGUI was introduced as a way to visualize a broad range of data (angles, forces, marker positions, etc.) at the individual trial level. However, most times the important information gained from our experiments is how different gait parameters change throughout the entire experiment. The (SubjectID)params.mat file (which is generated by the ReviewEventsGUI) exists specifically to make working with this information easier. These files are much smaller than the ones that include all of the raw/processed experiment data and therefore are faster to load making group analysis quicker.

The param.mat file stores a variable of the 'adaptaitonData' class which includes several methods for visualizing data on the experiment level. To view data for an individual subject, load the param.mat file so that a variable called adaptData appears in the workspace. To see which parameters are available to be plotted, type something along the lines of `'labels=adaptData.data.labels'`. Now, if you type `'adaptData.plotParamTimeCourse(specify a parameter or a cell of parameters here)'` A colorful plot of the parameters entered will appear. If you would like to see the data divided up by trial instead of by condition, use `.plotParamTrialTimeCourse` instead.

If you are interested in plotting the average values of different parameters early and/or late into each condition to compare them, you can use the function `.plotParamByCondition`, with the same

syntax as before. This will result in a bar plot of the average values of the parameters in the first 3/5 strides of each condition, and the last 20 strides of each condition.

You might be also interested in comparing several subjects, or even several populations in this way. You can do this by calling the static function `adaptationData.plotGroupedSubjects`. First you need to have the files that contain the `adaptData` variables for each subject (`param.mat`) available in your current directory. Once that is done, you can call:

```
adaptationData.plotGroupedSubjects({'Sub1file.mat','Sub2file.mat','Sub3file.mat',...},{'parameter1','parameter2',...})
```

Note that the first cell array is an array of strings, each one pointing to the name of one of the files containing adaptation data for a subject. The second cell array is an array of strings with the names of the parameters you wish to plot.

If you wish to compare populations (two or more groups of several subjects each), you can use the same function substituting the first argument. In this case, the first argument needs to be a cell array of cell arrays of strings, for example:

```
{{'Group1Sub1file','Group1Sub2file',...},{'Group2Sub1file','Group2Sub2file',...},{'Group3Sub1file',...},...}
```

Appendix

Adding folders to the MATLAB® path:

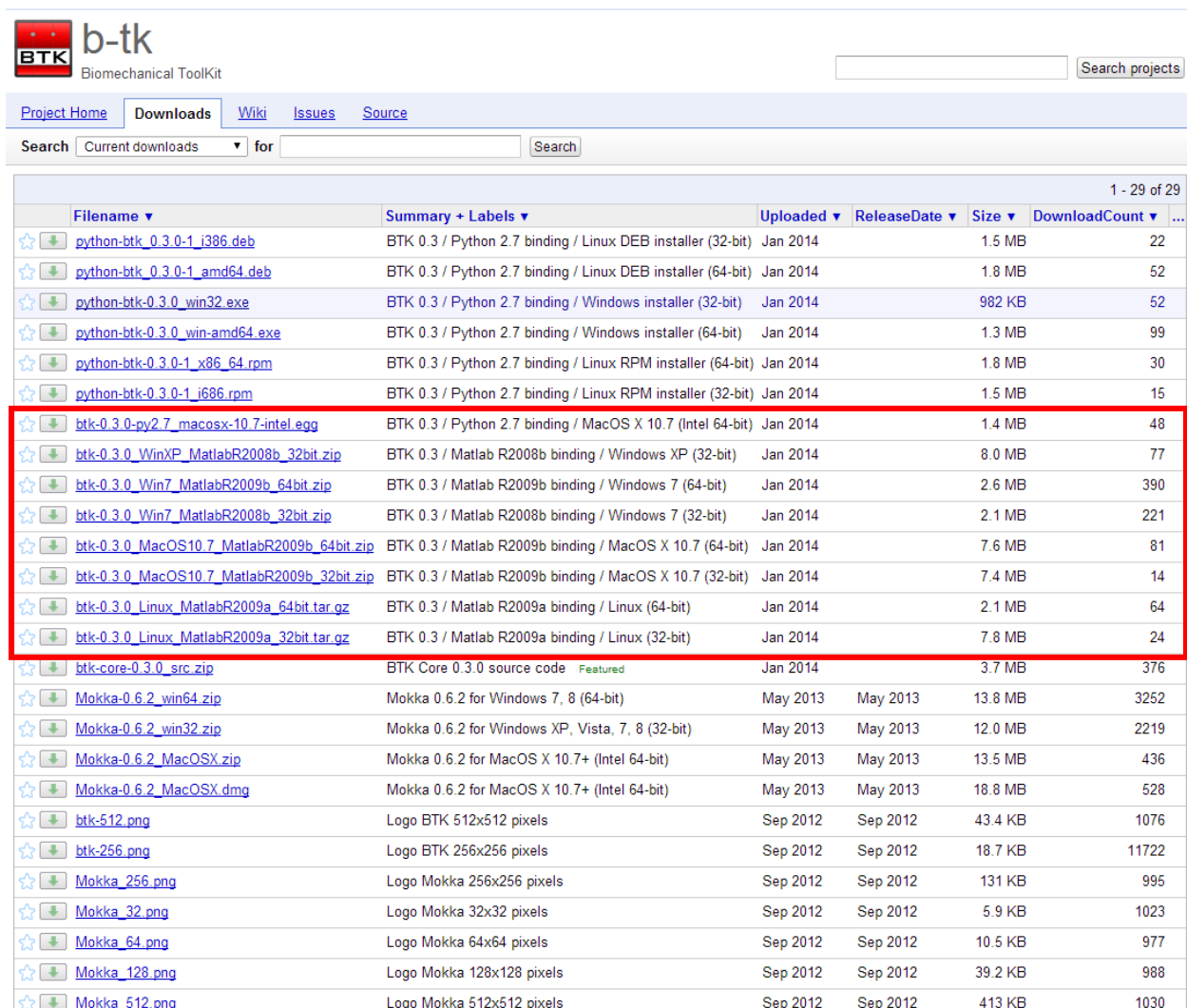
Go here for a how-to: http://www.mathworks.com/help/matlab/matlab_env/add-remove-or-reorder-folders-on-the-search-path.html

Make sure to choose 'add with subfolders...' as opposed to just 'add folder...' and to save changes before closing.

Downloading the Biomechanics Toolkit

Go to the btk website: <https://code.google.com/p/b-tk/downloads/list>

and then download one of the following downloads that corresponds to your operating system:



Filename	Summary + Labels	Uploaded	ReleaseDate	Size	DownloadCount	...
python-btk-0.3.0-1_i386.deb	BTK 0.3 / Python 2.7 binding / Linux DEB installer (32-bit)	Jan 2014		1.5 MB	22	
python-btk-0.3.0-1_amd64.deb	BTK 0.3 / Python 2.7 binding / Linux DEB installer (64-bit)	Jan 2014		1.8 MB	52	
python-btk-0.3.0_win32.exe	BTK 0.3 / Python 2.7 binding / Windows installer (32-bit)	Jan 2014		982 KB	52	
python-btk-0.3.0_win-amd64.exe	BTK 0.3 / Python 2.7 binding / Windows installer (64-bit)	Jan 2014		1.3 MB	99	
python-btk-0.3.0-1_x86_64.rpm	BTK 0.3 / Python 2.7 binding / Linux RPM installer (64-bit)	Jan 2014		1.8 MB	30	
python-btk-0.3.0-1_i686.rpm	BTK 0.3 / Python 2.7 binding / Linux RPM installer (32-bit)	Jan 2014		1.5 MB	15	
btk-0.3.0-py2.7_macosx-10.7-intel.egg	BTK 0.3 / Python 2.7 binding / MacOS X 10.7 (Intel 64-bit)	Jan 2014		1.4 MB	48	
btk-0.3.0_WinXP_MatlabR2008b_32bit.zip	BTK 0.3 / Matlab R2008b binding / Windows XP (32-bit)	Jan 2014		8.0 MB	77	
btk-0.3.0_Win7_MatlabR2009b_64bit.zip	BTK 0.3 / Matlab R2009b binding / Windows 7 (64-bit)	Jan 2014		2.6 MB	390	
btk-0.3.0_Win7_MatlabR2008b_32bit.zip	BTK 0.3 / Matlab R2008b binding / Windows 7 (32-bit)	Jan 2014		2.1 MB	221	
btk-0.3.0_MacOS10.7_MatlabR2009b_64bit.zip	BTK 0.3 / Matlab R2009b binding / MacOS X 10.7 (64-bit)	Jan 2014		7.6 MB	81	
btk-0.3.0_MacOS10.7_MatlabR2009b_32bit.zip	BTK 0.3 / Matlab R2009b binding / MacOS X 10.7 (32-bit)	Jan 2014		7.4 MB	14	
btk-0.3.0_Linux_MatlabR2009a_64bit.tar.gz	BTK 0.3 / Matlab R2009a binding / Linux (64-bit)	Jan 2014		2.1 MB	64	
btk-0.3.0_Linux_MatlabR2009a_32bit.tar.gz	BTK 0.3 / Matlab R2009a binding / Linux (32-bit)	Jan 2014		7.8 MB	24	
btk-core-0.3.0_src.zip	BTK Core 0.3.0 source code Featured	Jan 2014		3.7 MB	376	
Mokka-0.6.2_win64.zip	Mokka 0.6.2 for Windows 7, 8 (64-bit)	May 2013	May 2013	13.8 MB	3252	
Mokka-0.6.2_win32.zip	Mokka 0.6.2 for Windows XP, Vista, 7, 8 (32-bit)	May 2013	May 2013	12.0 MB	2219	
Mokka-0.6.2_MacOSX.zip	Mokka 0.6.2 for MacOS X 10.7+ (Intel 64-bit)	May 2013	May 2013	13.5 MB	436	
Mokka-0.6.2_MacOSX.dmg	Mokka 0.6.2 for MacOS X 10.7+ (Intel 64-bit)	May 2013	May 2013	18.8 MB	528	
btk-512.png	Logo BTK 512x512 pixels	Sep 2012	Sep 2012	43.4 KB	1076	
btk-256.png	Logo BTK 256x256 pixels	Sep 2012	Sep 2012	18.7 KB	11722	
Mokka_256.png	Logo Mokka 256x256 pixels	Sep 2012	Sep 2012	131 KB	995	
Mokka_32.png	Logo Mokka 32x32 pixels	Sep 2012	Sep 2012	5.9 KB	1023	
Mokka_64.png	Logo Mokka 64x64 pixels	Sep 2012	Sep 2012	10.5 KB	977	
Mokka_128.png	Logo Mokka 128x128 pixels	Sep 2012	Sep 2012	39.2 KB	988	
Mokka_512.png	Logo Mokka 512x512 pixels	Sep 2012	Sep 2012	413 KB	1030	

Unzip the folder that was downloaded and then add the folder (with subfolders) to your matlab path.