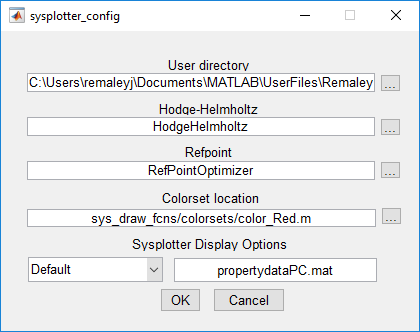
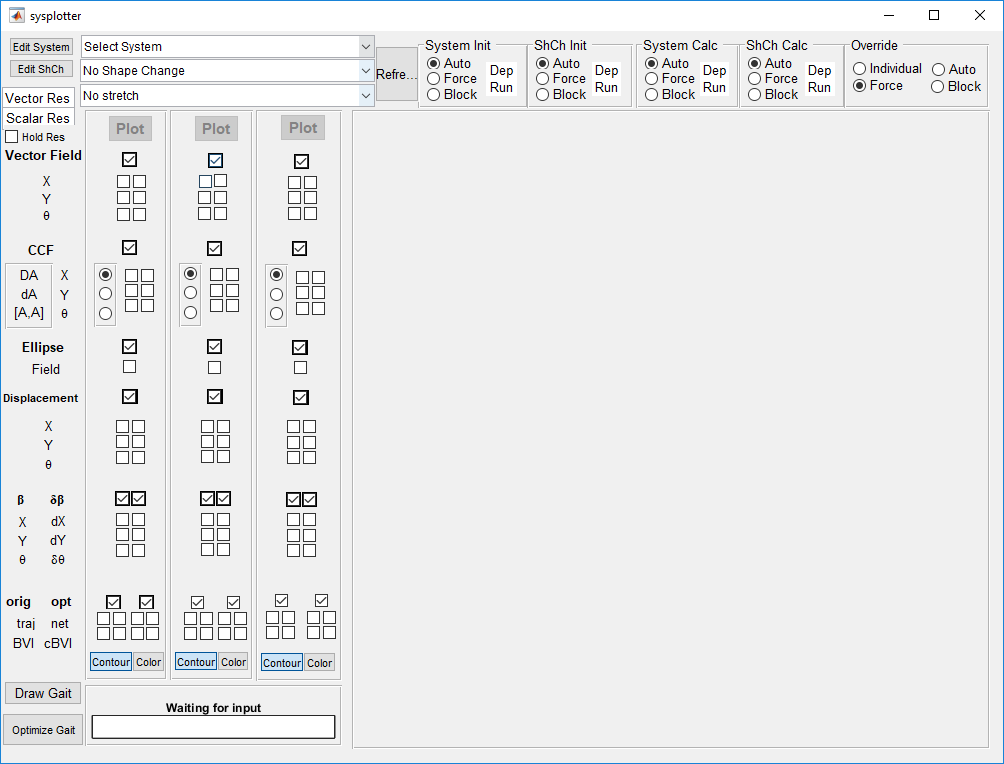
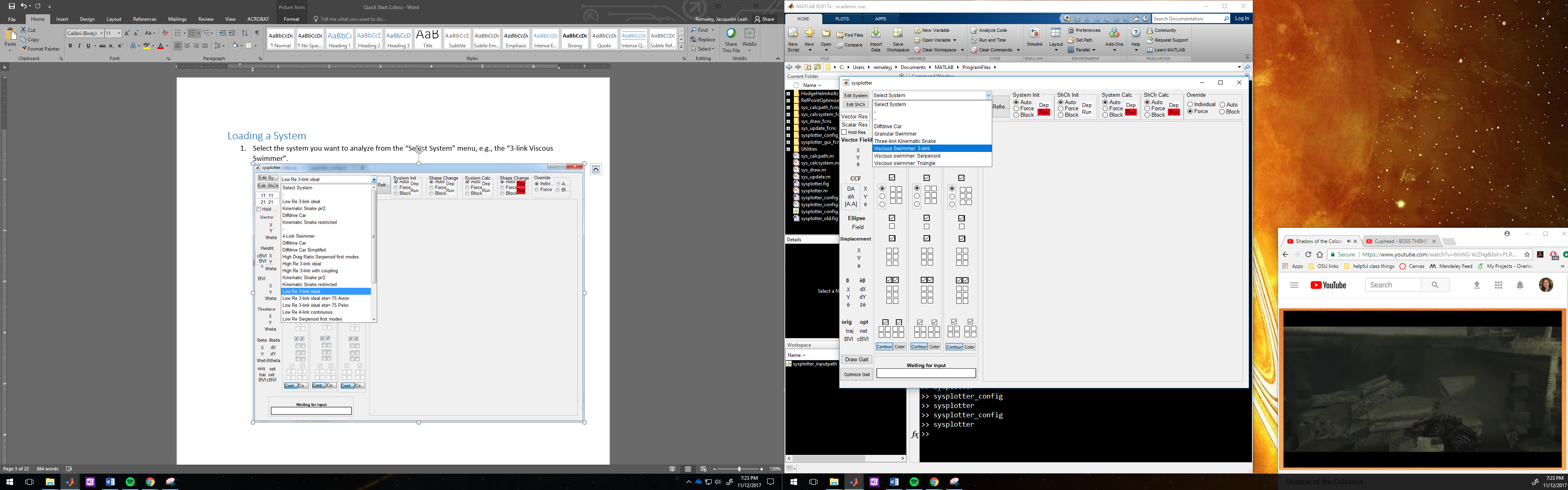
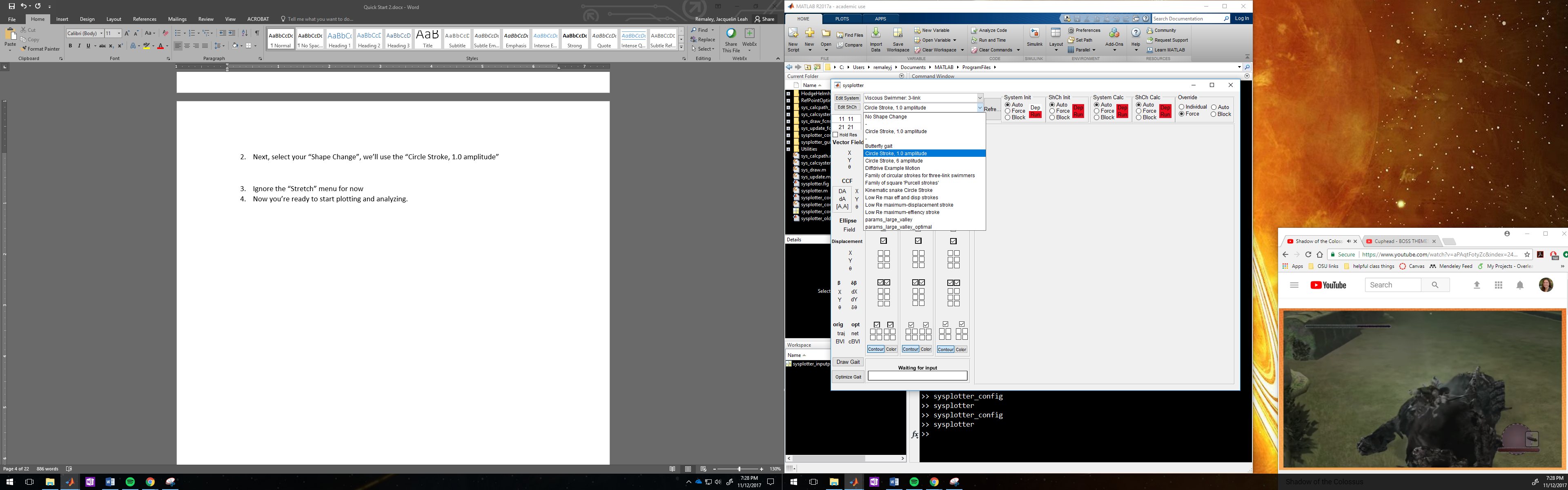
# **Geometric System Plotter Quick-Start** Start-up

1. Go to the …/GSP/ProgramFiles in your MATLAB directory.
2. Run sysplotter.m, which is in …/GSP/ProgramFiles.
3. The first time you start it up, the following screen will appear. (if you want to get back to this screen, you can run sysplotter\_config.m) Note that working in different operating systems will cause sysplotter to look slightly different.
4. Use the “...” box on the right of the dialog box to select the folder containing the appropriate directory. You can use the GenericUser directory in UserFiles/v4/, but we recommend that you make a copy of this directory with your name and work in the copy. This lets you share your work environment with someone else by simply giving them your user folder.
5. The last section of the dialog box allows you to select a property file for customizable display options. The default option is automatically detected depending on the operating system, though you may select any property file you wish for a personalized display. These files can be created by running GUI\_property\_scraper once the GUI has been edited to your liking, either manually or through GUIDE. Selecting “custom” in the dropdown menu opens a window to select the desired file, and the visual properties will be loaded the next time Sysplotter is opened.
6. This dialog checks to make sure that you have selected a user folder with the necessary subfolders, then lets you click “OK”
7. Your screen should now display the sysplotter



# Loading a System

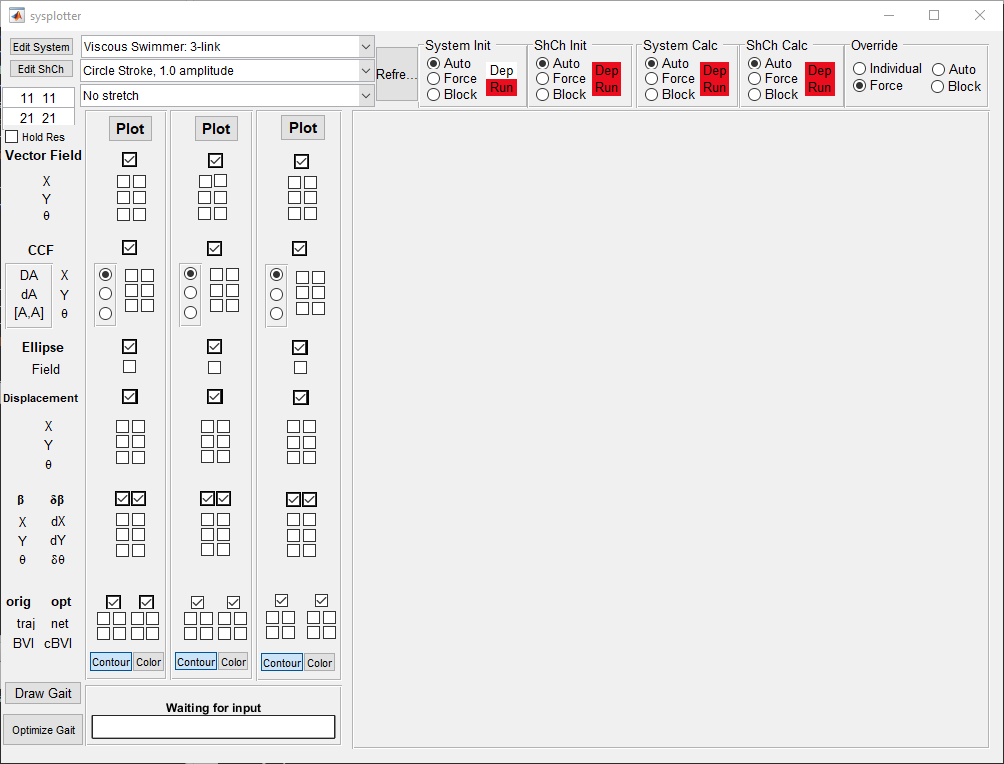
1. Select the system you want to analyze from the “Select System” menu, e.g., the “3-link Viscous Swimmer”.
2. Next, select your “Shape Change”, we’ll use the “Circle Stroke, 1.0 amplitude”

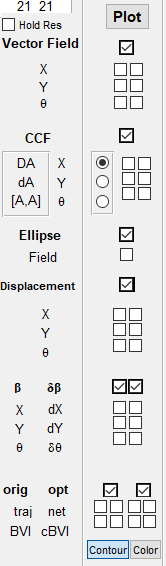


1. Ignore the “Stretch” menu for now
2. Now you’re ready to start plotting and analyzing.

# Plotting and Visualizing

Generate plots by selecting the corresponding checkboxes, then clicking the “plot” button. There are three sets of checkboxes (each with their own plot button), so that you can easily go back and forth between different sets of plots.

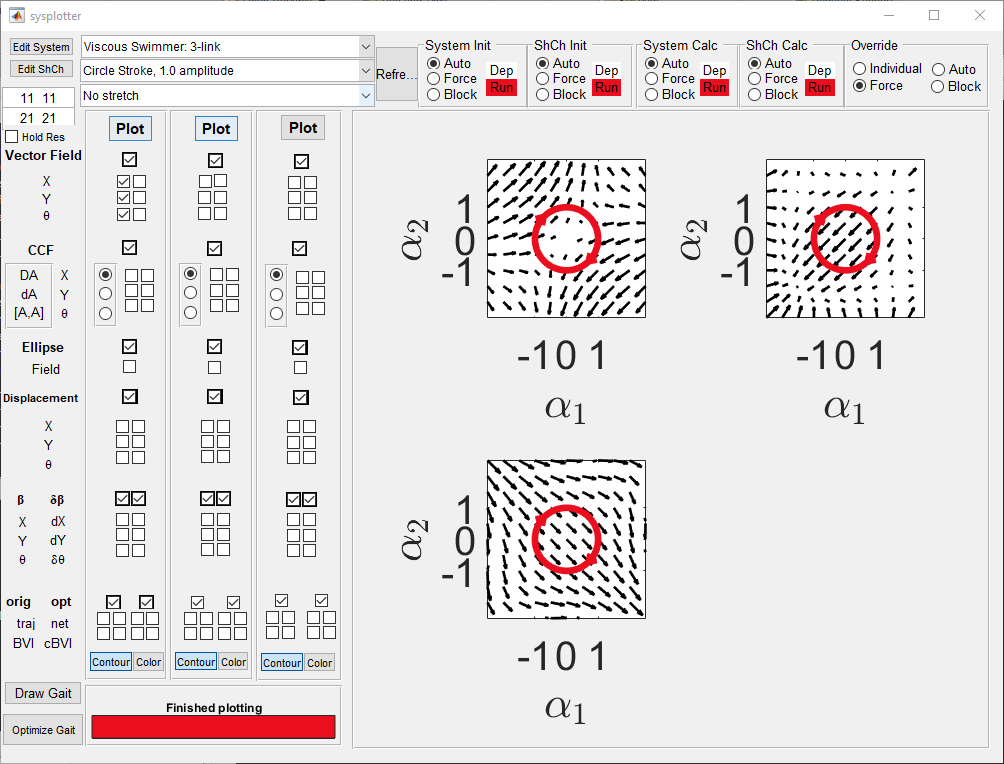


Each category of plots has a master-checkbox that toggles the whole block on and off.

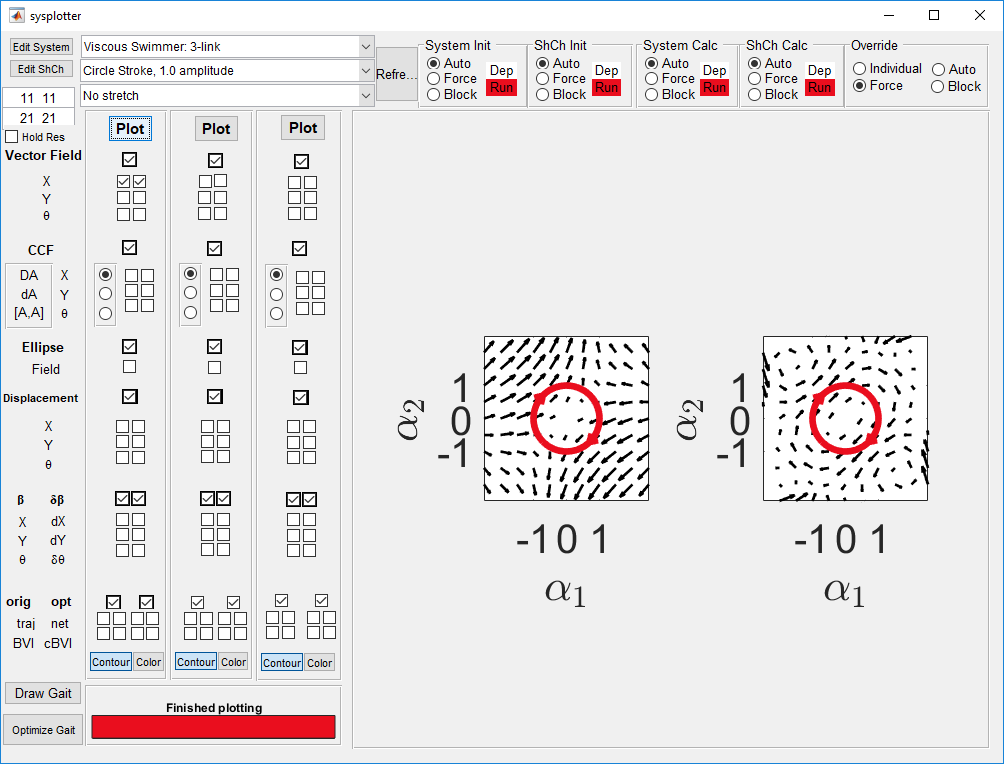
The Vector, Height function, Ellipse, and Displacement each have two columns; the left column plots the data exactly as it is specified in the system file, the right column plots it in the optimized coordinate system. Each of the three rows are their x, y, and theta components.

## Vector Fields

The “Vector” block plots the connection vector fields for the system, overlaid with the gait cycle selected. Here, all three fields are plotted in original coordinates



and here, the x component is plotted in original and optimized coordinates.



## Height Functions

The height functions work like the connection vector fields, except that there is a radio button for toggling between the height functions giving DA (the total curvature, with both curl and Lie Bracket components), da (curl only) and the correcting factor [A,A] (the Lie bracket only).



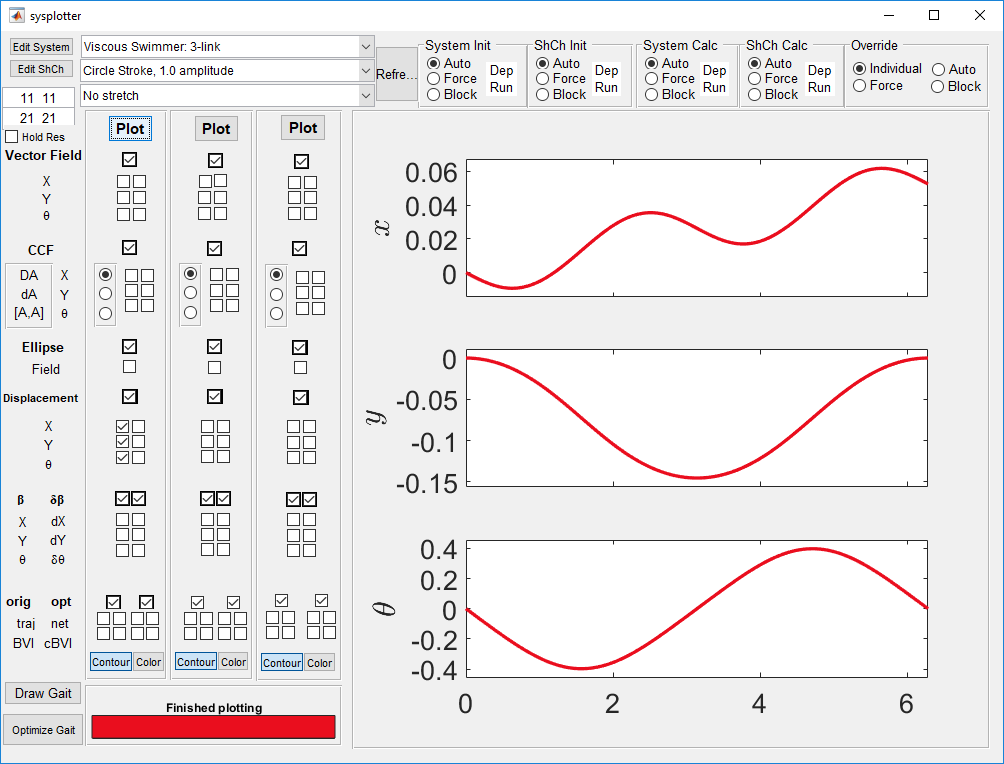
## Ellipse Field

Selecting the metric field checkbox plots an interpretation of the metric over the shape space. Here, circles at uniformly spaced grid points are modified using the metric and displayed as ellipses. This block only has one option, as the metric is defined for the overall system.

## 

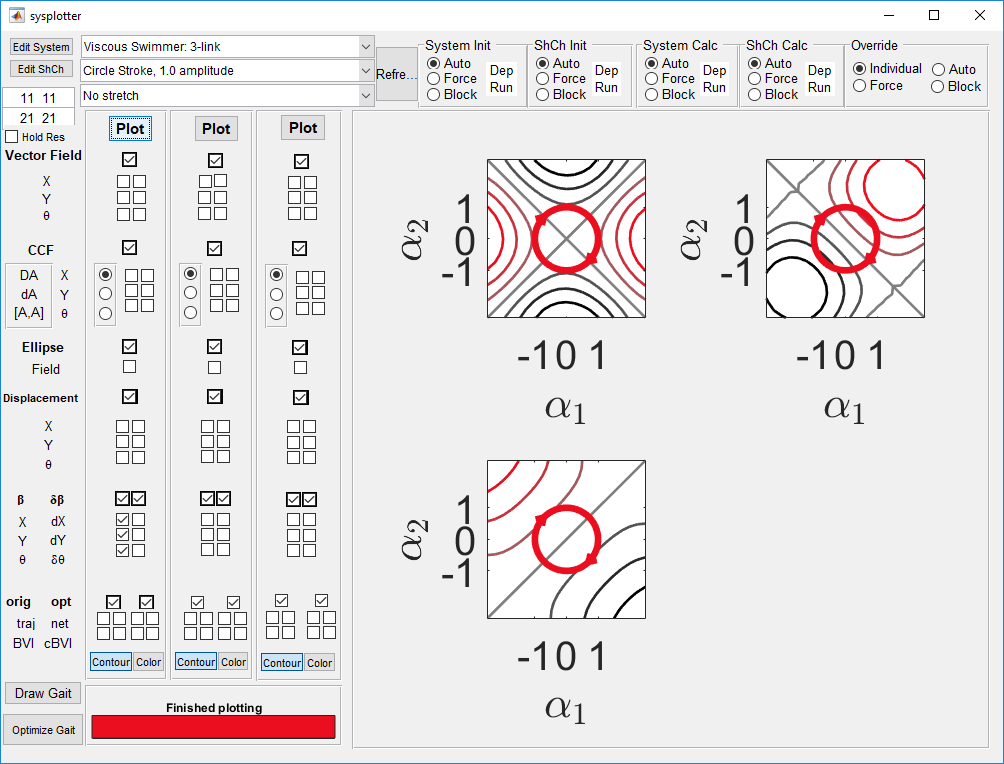
## Displacement

Time history of the x, y, and theta components of the displacement.



## Beta\

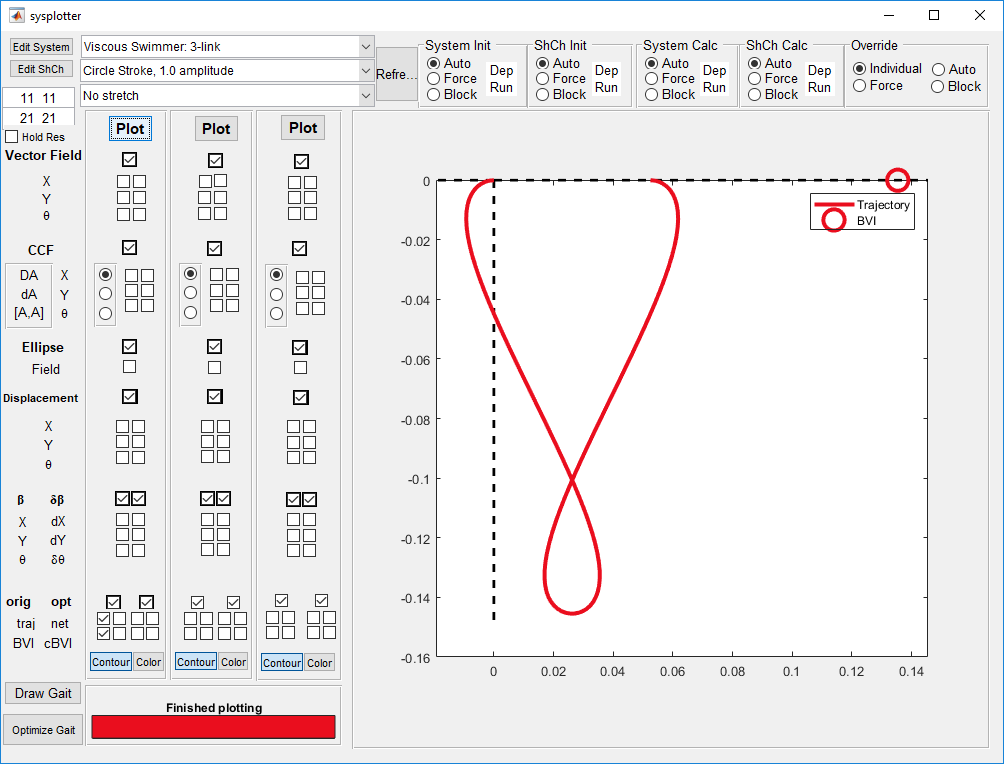
This shows the components of the transformation between original and optimal coordinates.

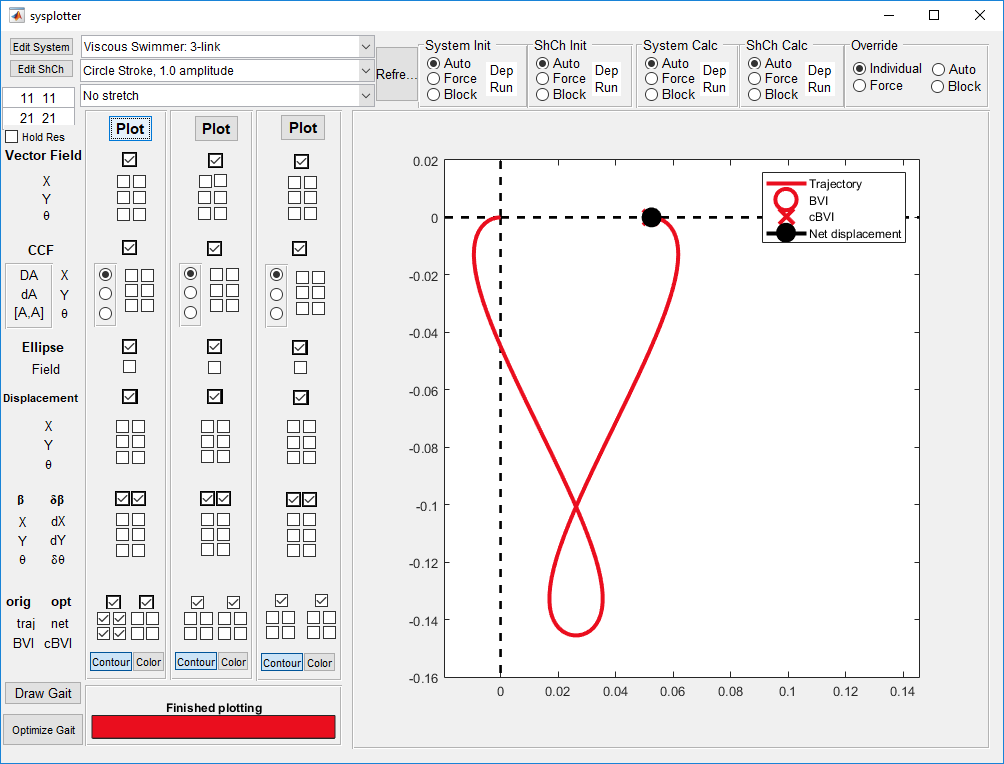


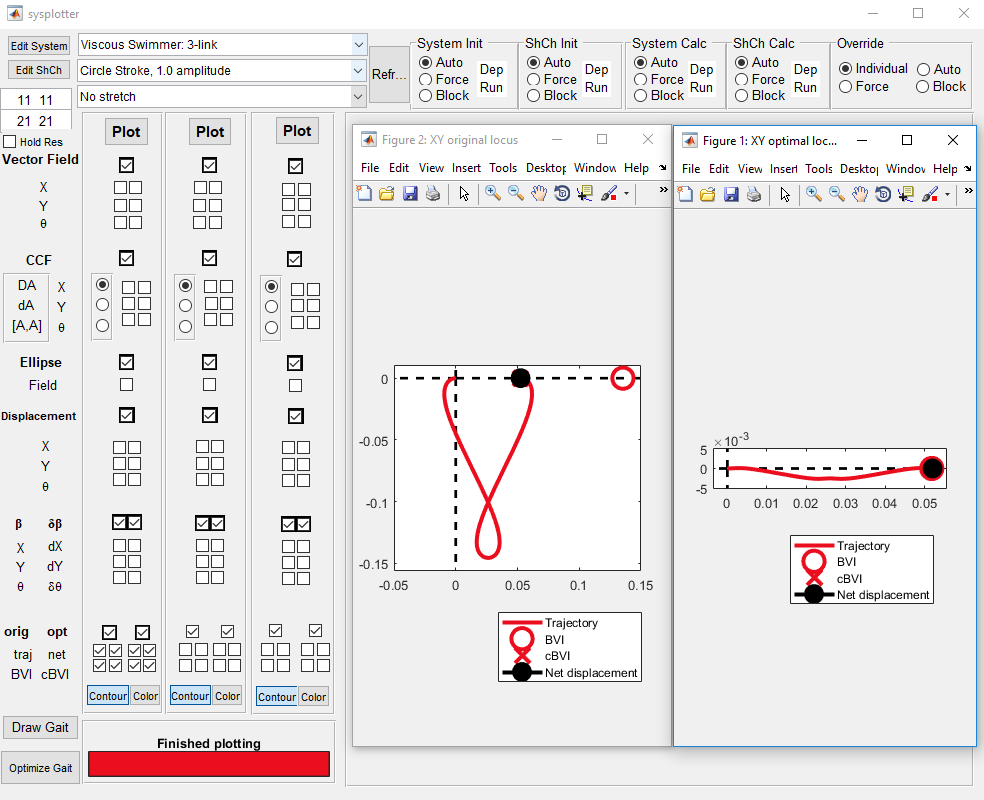
## Trajectories and Displacements

The bottom set of check operates somewhat differently from the rest. They are organized into two, 2x2 clusters. Each 2x2 cluster is organized as follows from top left to bottom right: trajectory of the body coordinate system, net displacement, body velocity integral estimate, and corrected body velocity integral estimate. The left cluster is in original coordinates, while the right cluster converts to optimized coordinates.

The image below shows the trajectory taken by the original coordinate frame, and the estimate the BVI gives for the displacement. (Notice the error).



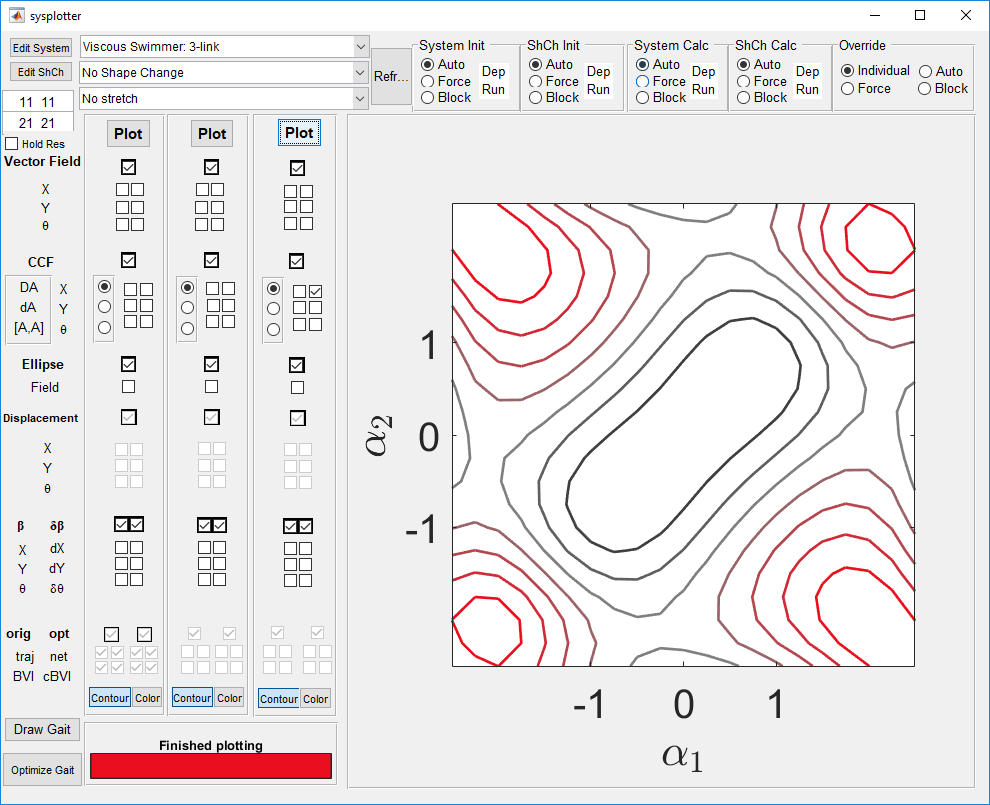
By adding the two check boxes to the right, we see the estimate given by the corrected-BVI.   
  


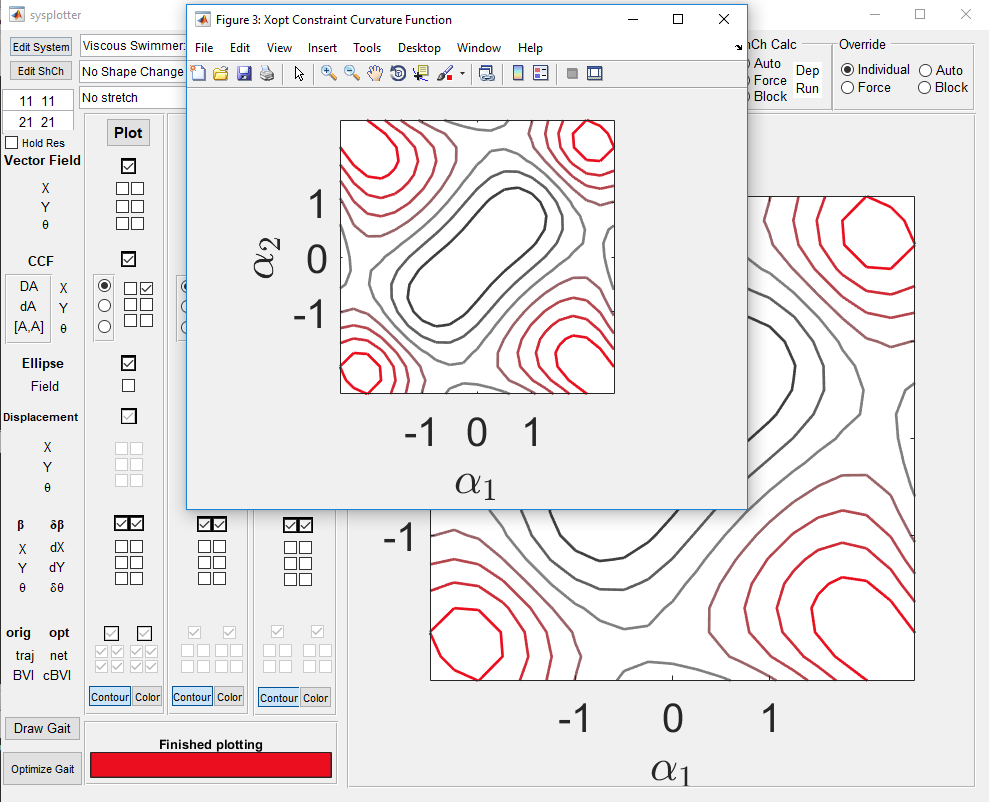
The 2x2 cluster to the right shows the trajectory taken by the optimized coordinate, as well as the estimates from the BVI and cBVI. Here, we have shown the plot results in external windows to move the legend off of the plots. Opening external windows is explained in a later section.  


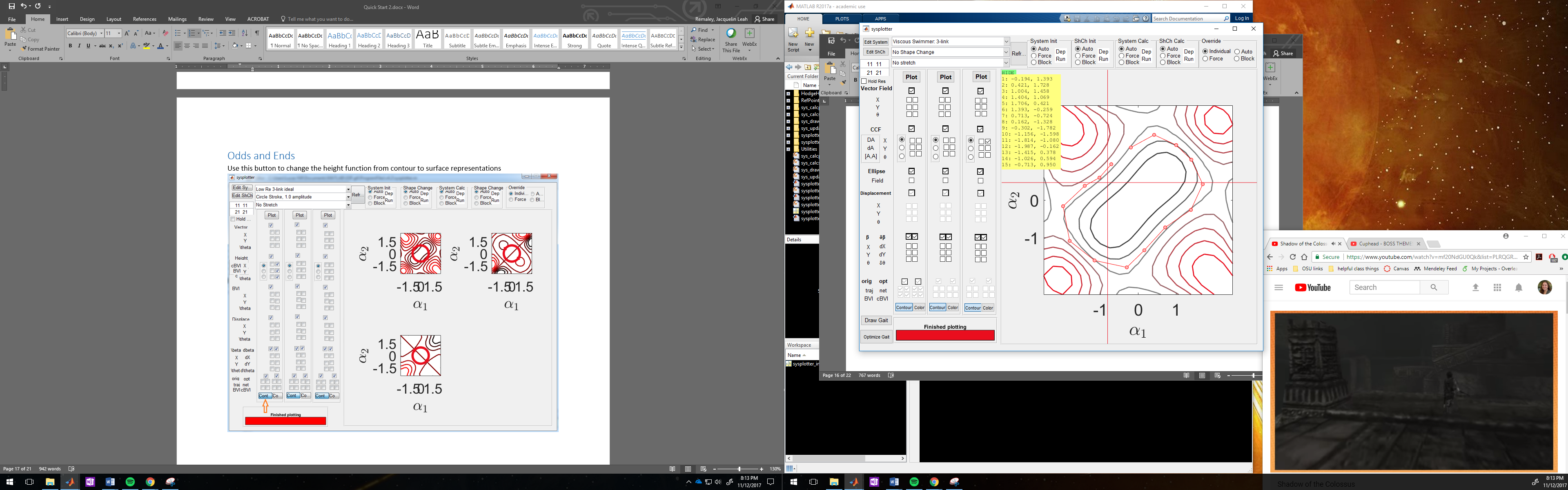
# Custom Closed-Gait Generation

**Gait generation can only be performed using the checkboxes in the third column.**

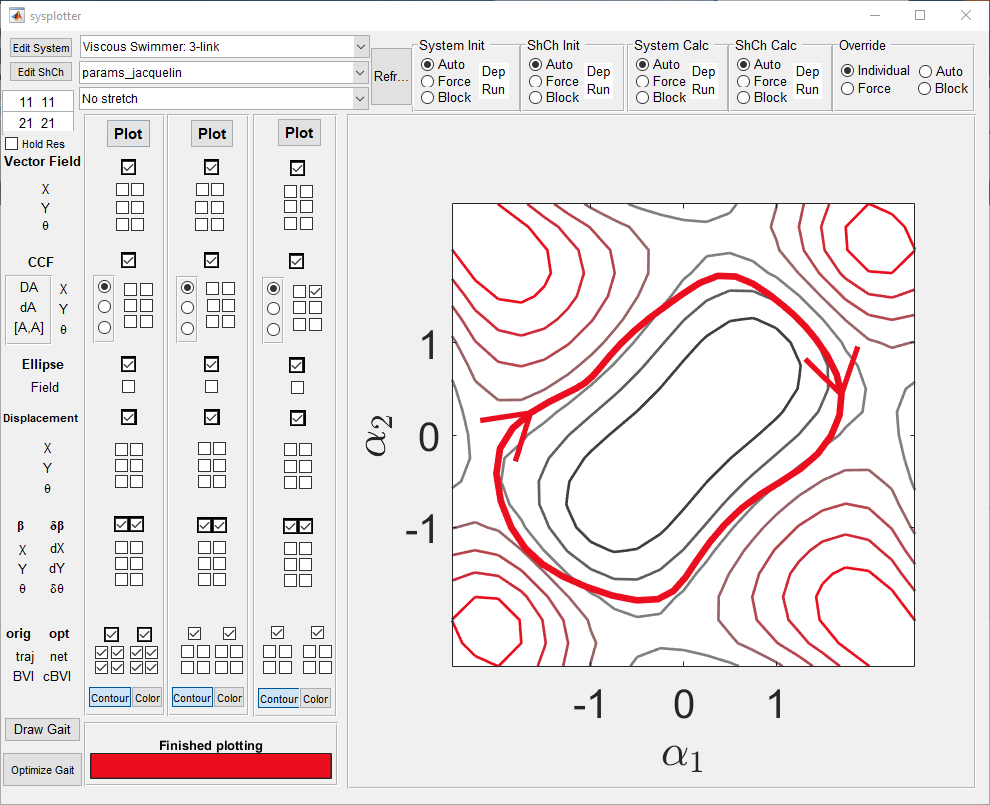
1. Generate a height function.



1. Select the “Draw Gait” button in the lower left of the GUI.
2. Click inside the figure of the height function to add points, hit enter when done, a save dialog will appear to save your gait contour in your userfiles folder. Before saving, you can preview the spline fit to your points in the same window used for entering them.



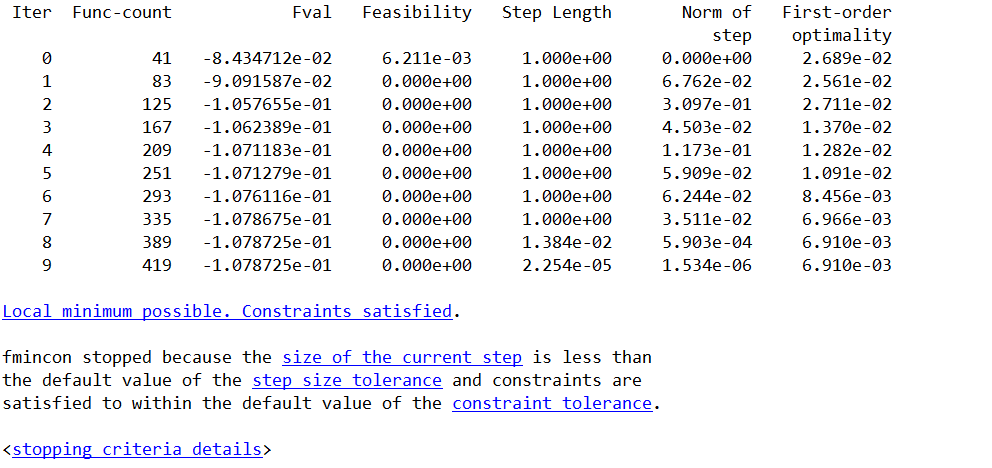
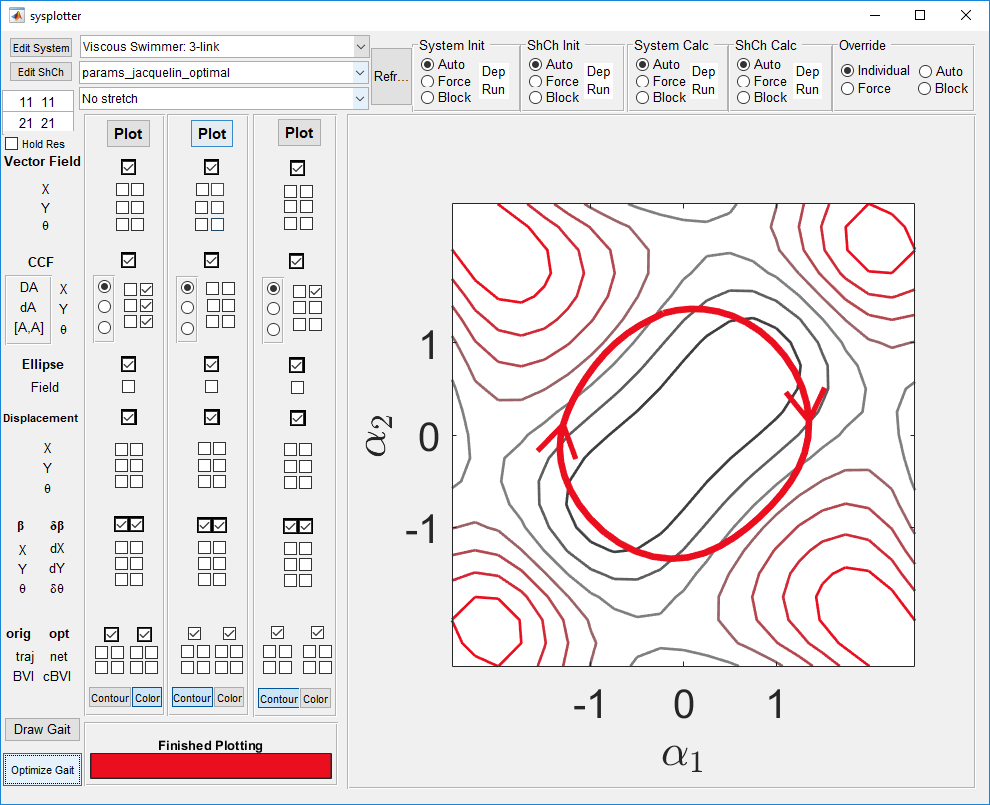
1. Now, in the sysplotter main gui, you can select your gait, through the Shape Change menu.



# Optimizing Gaits

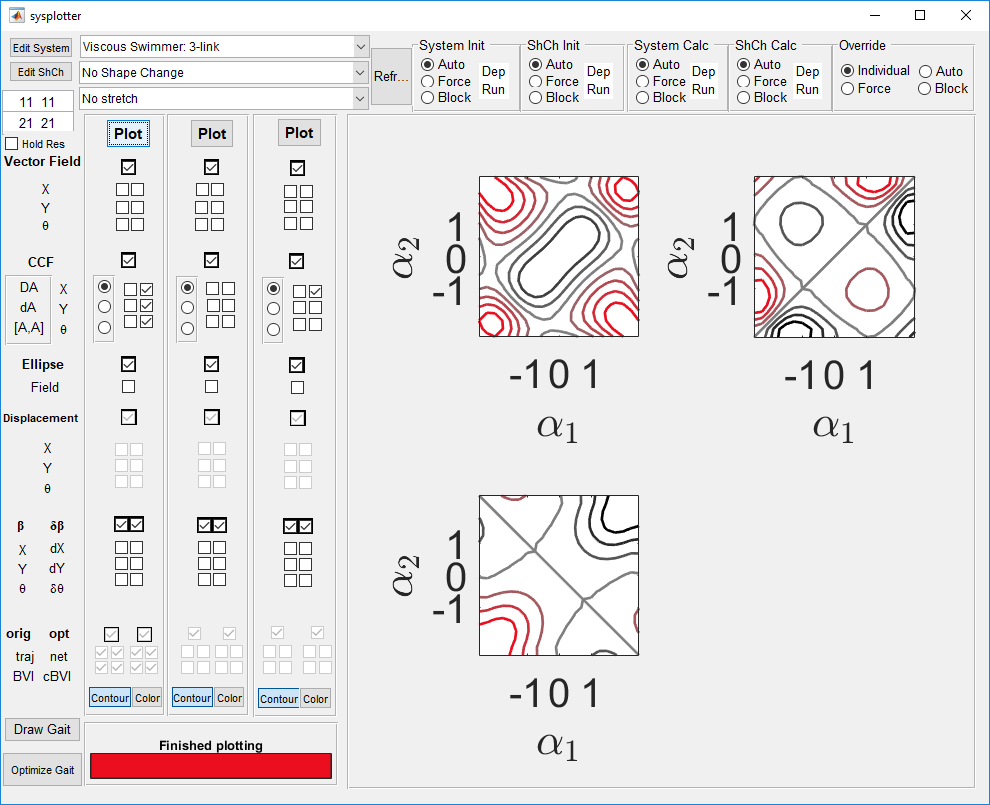
The optimize gait button can be used to optimize a user-defined gait for the selected system. **This button will only function when the third column of checkboxes is used.**

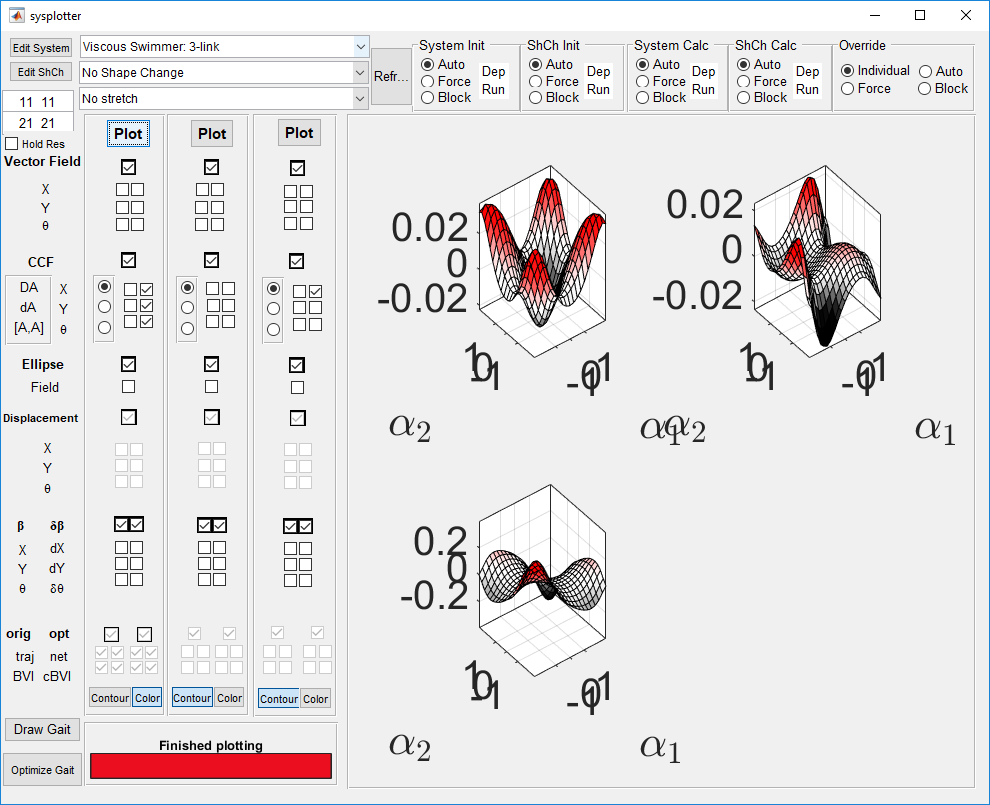
With any gait plotted, select the optimize gait button located in the bottom left corner. This will use the existing gait as a seed for the optimizer, which will periodically display information in the command window until a solution has been reached. As an example, after running for roughly two minutes, this was the output of the display window:

  
  
The optimized gait is automatically displayed in Sysplotter.   
  


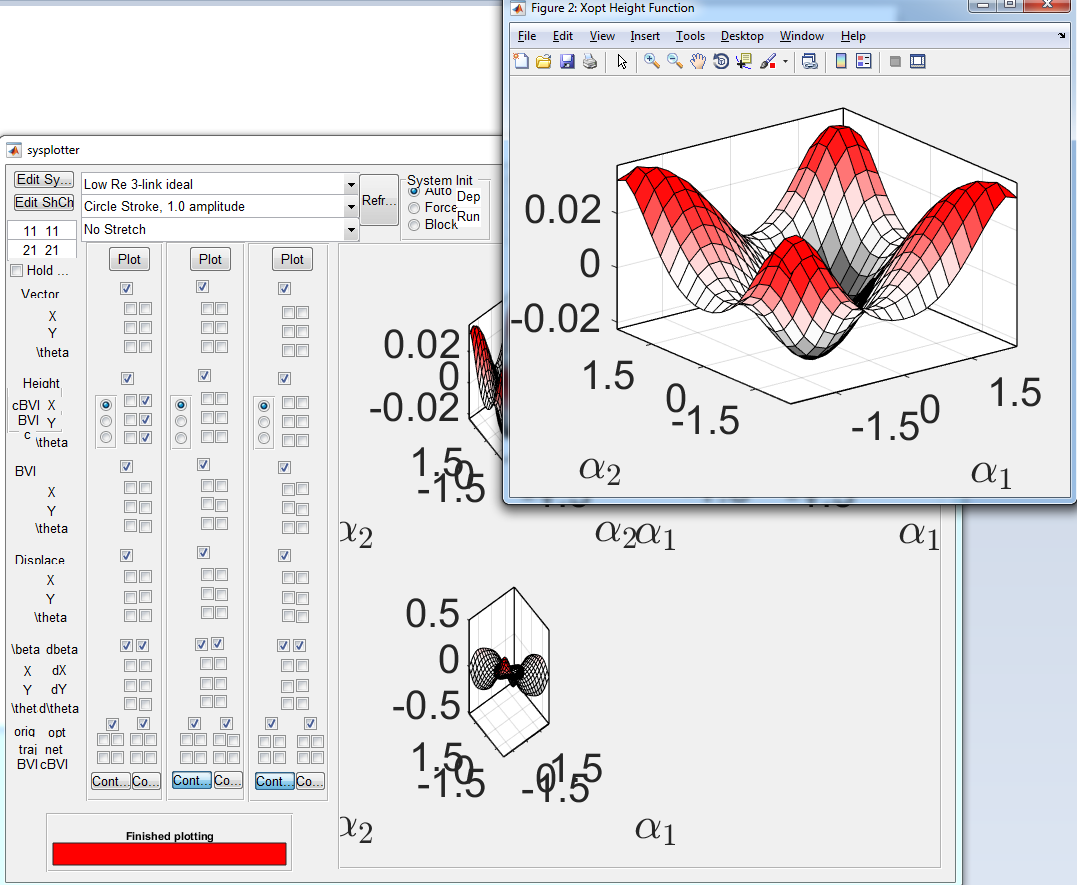
# Odds and Ends

Use this button to change the height function from contour to surface representations.



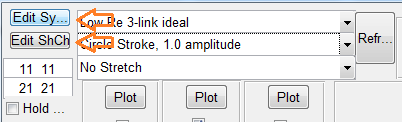


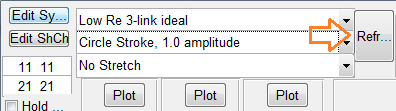
Also, all plots can be clicked on to produce a single figure containing that graph. These figure windows have the plotted contents title as the window name, and do not close when Sysplotter is run again.



# Adding your own files

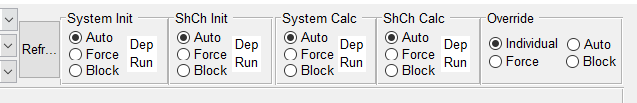
You can add new systems and shape changes to sysplotter by:

1. Clicking the “Edit” buttons to the left of the selection menus 
2. Saving a copy of the file into the appropriate directory. Make sure the file starts with sysf\_ or shchf\_ (system file or shape change file) so that sysplotter will recognize it
3. Changing lines of code in the file to get the behavior you want. Be sure to update the display name.
4. Click the Refresh button to the right of the selection menus to have the new file adopted into the menus.



# Re-calculating System and Shape Change data

# The boxes on the top right of the interface contain radio buttons which can modify how Sysplotter calculates the selected data. By default, to save time, Sysplotter will check to see if any of the system or shape change files have been changed. If they have not, the most recent data will be called and no calculations will be performed. If they have, Sysplotter will run the new files to generate fresh data which will override the old.



The refresh button causes Sysplotter to check the user directory for systems and shape changes for new files or changes to existing files.

Initialize and Calculation panels contain three options:

* Auto: by default, allows Sysplotter to automatically decide whether or not to run fresh calculations.
* Force: Sysplotter is required to run all calculations, regardless of existence of previous data.
* Block: Ignores changes in files, and uses only previously calculated data.

The override panel acts as a controller for the other panels. When set to individual, each of the other panels acts on its own and the settings can be changed individually. When set to Auto, Force, or Block, all of the other panels act the same according to the selection.

# Empirical Local Connection Calculation

# This toolbox, located under Utilities, allows the user to create local connection matrices from motion capture data.

# **clean\_mocap\_data** After obtaining the motion capture data, the user must convert it using their own methods into a form usable by “clean\_mocap\_data”:

* Three separate position arrays, X Y Z, whose dimensions are (n time steps)x(m links)
* A time array t with dimensions (nx1).

This file takes in position data and moves it around, first aligning it with the proper coordinate axes, and then in point order. If the variables ‘fix\_axis\_order’ and ‘fix\_point\_order’ are specified when the function is called, the data will be immediately rearranged and a sample plotted. If these variables are not given, both rearrangements will plot a sample coordinate set and ask the user if they look as expected.

Responses to this question include:

* 0 – No, the plotted image does not look correct.
* 1 – Yes, the plotted image is right.
* 2 – No, the plotted data looks very wrong and I want to stop the analysis altogether.

For response 0, the code will proceed to ask the user to input an array for rearranging the data. Both the new axis and point order must be input as arrays within square brackets. The values input are up to the user’s judgement, but the quickest way to reorder correctly is to input the same order as is displayed. For example, if the points are displayed out of order as [3 4 1 2], this is the exact array the user would enter when prompted.

For response 1, that portion of the reordering will finish and the code will move on.

Response 2 is meant primarily as a debugging tool and will error out of the code, cancelling any other processes. Use only if something appears drastically incorrect.

**empirical\_local\_connection**  
This is the primary local connection calculation function. It expects a very specific data format, so use the data cleaner if necessary.

This code requires the following inputs:

* **x\_pos, y\_pos –** (nxm) mocap position data.
* **time\_array –** (nx1) mocap time array.
* **grid\_axis –** (4x1) axis limits array. Same format as figure axis setting, [xmin xmax ymin ymax].
* **num\_grid\_points –** the number of grid crosses in each direction. Generates pxp meshgrid arrays
* **save\_file\_name –** string which specifies the name the local connection data will be given, for a sysf file to reference.
* **downsample\_rate -** integer >= 1. For selecting only every few points. Arrays will be re-defined using (1:downsample:end) in an effort to smooth noisy mocap data and decrease array size.
* **debug –** (2x1) array of logicals.   
  First specifies **plot\_neighborhood**, which will display the shape array and all points found within a given gridpoint. Second specifies **plot\_valid\_connection**, which displays a pxp grid of points that are color coordinated depending on the status of the local connection value at that point. If there is a NaN generated from not having enough neighborhood points, for example, the point will be red. This will cause sections of missing or incorrect data when Sysplotter plots the local connection data. To alleviate this, modify grid\_axis to encompass enough points.

The functions referenced within this code are:

* **threeLink\_info\_from\_mocap**  
  Assuming three-link geometry, extracts shape values, shape velocity, and body velocity. Body position is also calculated, but is not reported.
* **A\_from\_data**  
  Performs the matrix multiplication necessary to extract the local connection from the body velocity and shape velocity according to the user-defined grid parameters and debugging preferences.
  + **neighborhood\_points** – find all shape coordinates within range of the grid point
  + **empirical\_local\_conn\_calculation** – pseudo-inverse matrix calculation with points within range