In order to view all files in a folder, conduct the following:  
C:\Users\admin\Documents\GitHub\GradProject>tree /f  
In the cmd command, as explained in [this link](https://stackoverflow.com/questions/15214486/command-to-list-all-files-in-a-folder-as-well-as-sub-folders-in-windows).   
  
Self-Assembly Simulator Code Summary

**The Main Folder**

* AddLetters2Plots.m – Graphic sssistance function for figures.
* board.py- The simulation board definitions and iterations.
* BubblePlot.m- This plots a bubble plot of the data in 2D, to simulate occurrence in the data of a certain points by changing its size accordingly. Taken from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/6805-bubbleplot?s_tid=srchtitle).
* buseg.m- A function for Fisher information matrix based time-series segmentation of process data assistance function, taken from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/47194-fisher-information-matrix-based-time-series-segmentation-of-process-data).
* CallBubblePlot.m- This calls a BubblePlot.m of the data, to simulate occurrence in the data of a certain points by changing its size accordingly. Taken from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/6805-bubbleplot?s_tid=srchtitle).
* centerOfMass.m- calculate the center of mass of data. Taken from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/41675-center-of-mass?s_tid=ta_fx_results).
* cfg.json- The configuration file of the Python Simulation.
* Collection\_Tfas\_DATA.m- Collects all the data created from Gather\_Tfas….m, and creates a structure for TFAS estimation called "all\_data" with the number of requested dimensions.
* entropy.py- Defines entropy class for the Python simulation.
* exceptions.py- Common errors created in the Python Simulation.
* Fisher\_resid\_bu.m- A function for Fisher information matrix based time-series segmentation of process data assistance function, taken from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/47194-fisher-information-matrix-based-time-series-segmentation-of-process-data).
* FitEXPDecaytoData.m- Fits exponential function to data.
* fnn\_deneme.m- Minimum embedding dimension assistance function, [taken from this link.](https://ch.mathworks.com/matlabcentral/fileexchange/37239-minimum-embedding-dimension?s_tid=srchtitle)
* fnn\_deneme\_call.m- Minimum embedding dimension assistance function, [taken from this link.](https://ch.mathworks.com/matlabcentral/fileexchange/37239-minimum-embedding-dimension?s_tid=srchtitle)
* fullscreen.m- creates a fullscreen in the code automatically, taken from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/48085-fullscreen-h?s_tid=srchtitle).
* Gather\_Tfas\_DATA\_Many\_Drives.m- This is the MAIN function that needs to be run AFTER THE PYTHON SIMULATION, once the simulation results are in the right directory of ResultsDir. This function contains the transformation of the stochastic coordiantes (post BEAST algorithm) into a hyperplane with the relevant statistical moments for the trajectory as coordinates. This does so for many drives one after antoher, based on what simulated.
* Gather\_Tfas\_DATA\_Single\_Drive.m- This is the MAIN function that needs to be run AFTER THE PYTHON SIMULATION, once the simulation results are in the right directory of ResultsDir. This function contains the transformation of the stochastic coordiantes (post BEAST algorithm) into a hyperplane with the relevant statistical moments for the trajectory as coordinates. This does so for a single drives, based on what simulated.
* GetFullPath.m- Gets the full path name for partial path. This is taken from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/28249-getfullpath?s_tid=srchtitle).
* give\_vecs.m- A relevant function for fetching the stochastic coordinates after the BEAST algorithm, this is called from the Gather\_Tfas\_DATA…m functions.
* Initial\_Seed\_Start\_Results.m- This shows past simulation results of starting from the seed in the Python simulation, where a different seed was simulated other than random initial conditions.
* InstallMex.m- This function installs a relevant compilation for the code to interface with, might be relevant for the BEAST activation.
* LogTFasDraftEstimation\_LOGTriangulationUP\_Many\_DrivesWith2D.m- This gives the TFAS prediction of the remaining time to self-assemble in a log-log presentation for all drives and strong energies simulated. This requires conducting the Python simulation, moving its results into the result directory, and then activate GatherTFASData..m. Afterwards, the function Collection\_Tfas\_DATA.m needs to be activated. Then, this simulation would give the prediction results.
* main.py- The main of the Python simulation, this is where the simulation starts running and all of the parameters of relevance defined.
* mArrow3.m- This is a graphical function that displays trajectories on a scatter or 3d, along with the arrow of order between points. This is taken from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/25372-marrow3-m-easy-to-use-3d-arrow).
* mat2np.m- converts a .mat file to python array file for MATLAB usage.
* Mat2Py.py- converts a.mat file to python array file for Python usage.
* MovieMakerCall.m- This call a movie creation of the simulation results. This function is probably outdated and remained in case needed for other functions.
* MovieMakerCallUP.m- This call a movie creation of the simulation results, and saves it in memory as avi file. This is currently relevant only for two stored targets in memory.
* particle.py- This is the class of particle for the python simulation.
* partitions.m- This functions computes all the possible combinations of a sum counterparts with a constraints of the sum results. This is taken from [this link.](https://ch.mathworks.com/matlabcentral/fileexchange/86277-partitions?s_tid=srchtitle)
* partitions\_demo.m- This makes an example of the partitions.m function.
* PeakDetector.m- This function is an automated detector of peaks in data, was used to test whether peaking can lead to a time series segmentation in data.
* PlotAllMovieUPDATED.m- This plots all the simulation movies, as it calls MovieMakerCallUP.m and saves the results in avi files.
* PlotFigure2a\_Left\_Side\_GiliOriginalPaper.m- This function attempts to replicate the figure 2a left side from the paper "Nonequilibirum assosiative retrieval of multiple stored self-assembly targets".
* PlotFigure2a\_Right\_Side\_GiliOriginalPaper.m- This function attempts to replicate the figure 2a right side from the paper "Nonequilibirum assosiative retrieval of multiple stored self-assembly targets".
* PlottingBEASTEnergyTrajectories.m- This function is plotting all the energy trajectories in the relevant dataset, along with the BEAST algorithm trajectory time segmentations as the BEAST algorithm computes. Relveant also for plotting this result.
* Plotting\_Tracjectories\_Ensemble\_No\_BEAST.m- This plots the trajectories of any Macro parameter of interest (Total entropy, energy and distance from the targets) versus time for many trajectories at once.
* Plot\_All\_SImulation\_Macro\_Parameters\_Figures.m- This function plots a certain simulation macroparameters versus time trajectories.
* Plot\_Histograms\_KLD.m- This plots the histograms using the KDE method for any drive and strong energy of interest. This function is called from the TFAS estimation MATLAB functions.
* Plot\_Histograms\_KLD\_mu\_sigma.m- This plots the histograms using the KDE method for any drive and strong energy of interest. This function is called from the TFAS estimation MATLAB functions. This also returns the STD and Mean of the ditrubtion in log scale.
* ReadResultsFromFoldersJfunction\_\_tfas\_results\_Js.m- This function is quite old, but assists in plotting data of Tfas vs Js under different simulation conditions.
* ReadResultsFromFoldersJfunction\_\_tfas\_results\_mu.m- This function is quite old, but assists in plotting data of Tfas vs drive under different simulation conditions.
* RecurruenceRateETC.m- This plots the recurrence plot of the data. Using this was used when we thought of using this idea for deducing the time series segmentation out of standard dynamical system theory.
* Resize\_figure\_for\_full\_screen\_attempt.m- correct resize function just for the screen in our Lab, Michael's computer.
* scriptToExc- This function allows us to activate the simulation from the university services from afar.
* seed.json- This is a .json file to describe the initiated seed for the simulation to start with (initial conditions).
* SeedJson.py- This assits the user to create a seed .json according to graphical interface and creates the .json file.
* shade.m- a plotting function for the STD in a 2d plot in a pleasant way. This is taken from [this link.](https://nl.mathworks.com/matlabcentral/fileexchange/13188-shade-area-between-two-curves) Consider taking it from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/58262-shaded-area-error-bar-plot?s_tid=srchtitle) instead.
* Simulation\_cfg.py- A simulation configuration file relevant for its work, no need to update or change at the moment.
* Smoothing\_Using\_Mutual\_Information.m- This function created out of an attempt to smooth the data using mutual information as in classical dynamical system approach.
* stdshade.m- Conducting the shade function for plotting the standard deviation.
* take\_change\_points.m- This function changes the way changepoints are taken from the BEAST algorithm. Might needs to be considered for future methods.
* take\_change\_points\_option2.m- This function changes the way changepoints are taken from the BEAST algorithm. Might needs to be considered for future methods.
* target.py- The target class relevant function for the Python simulation.
* TdraftEstimation\_add\_on\_Weighted\_Euclidian\_Matrix.m- This is a supportive function in case we give up the current way of log-triangulation for the data, and prefer to use a weighted Euclidian matrix.
* TFasDraftEstimation\_LOGTriangulationUP\_Many\_DrivesWith2D.m- This gives the TFAS prediction of the remaining time to self-assemble for all drives and strong energies simulated. This requires conducting the Python simulation, moving its results into the result directory, and then activate GatherTFASData..m. Afterwards, the function Collection\_Tfas\_DATA.m needs to be activated. Then, this simulation would give the prediction results for the TFAS.
* TFasDraftEstimation\_LOGTriangulation\_Many\_DrivesWith2D\_simple\_figures.m- This gives the TFAS prediction of the remaining time to self-assemble for all drives and strong energies simulated. This requires conducting the Python simulation, moving its results into the result directory, and then activate GatherTFASData..m. Afterwards, the function Collection\_Tfas\_DATA.m needs to be activated. Then, this simulation would give the prediction results for the TFAS in a very simplified form.
* Tfas\_Distrubtion\_Drives\_Margi\_UPDATED.m- This plots the estimation for the pdf of the Tfas versus drive using the KDE and standard histogram methods, with relevant adjustments.
* The\_one\_choosen\_trajectory.m
* Time\_Lag\_Visual\_Recuurence\_Plot.m- This plots the recurrence plot versus time lag 2D representation, not relevant for our methods today.
* TrajectoryPlot.m- This plots the trajectory in the hyperplane of the simulation results. This function is the original one, taken from [this link](https://ch.mathworks.com/matlabcentral/answers/441886-to-plot-x-y-z-according-to-time?s_tid=prof_contriblnk).
* TrajectoryPlotVecs.m- This is the utilization of TrajectoryPlots.m for our code in general.
* TrajectoryPlotVecs4TfasMap.m- This is the utilization of TrajectoryPlots.m for our code for remaining TFAS map.
* turn\_callbacks.py- The turn callback functions in each of the turns taken in the Monte Carlo simulation, called by Board.py after starting the simulation via main.py.
* uTest\_GetFullPath.m- Assistive function to get the full path name for partial path. This is taken from [this link](https://ch.mathworks.com/matlabcentral/fileexchange/28249-getfullpath?s_tid=srchtitle).
* utils.py- utility functions relevant for the Python simulation, called when the main simulation runs.

**Key Figures SubFolder**

* ConferenceDisplay.m- Assistive function for a conference display if needed.
* EnergyDepictionBeast.m- This depicts the energy trajectories after the BEAST algorithm activation on them for the energy trajectories.
* PlotBEAST.m- This plots the BEAST algorithm for a macro-parameter of choice of the simulation for one example trajectory.
* PlotLogMapImage.m- This plots a log map of the abstract 2D space embedding the remaining time to self-assembly in it.
* PlotMeanTFAS\_JS.m- This plots the mean TFAS versus Js for 2 stores targets figure.
* PlotMeanTiT\_JS.m- This plots the mean TFAS versus drive for 2 stores targets figure.
* Plotting\_Tracjectories\_Ensemble.m- This plots the trajectories of any macroparameter of the simulation versus time, without the BEAST activation on it.
* TdraftEstimation\_add\_on.m- This is a supportive function might be needed in the future for conducting weighted Euclidian matrix prediction of the data.