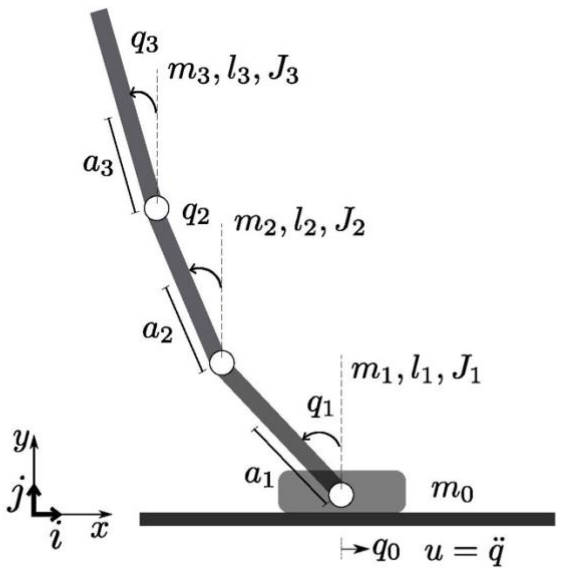
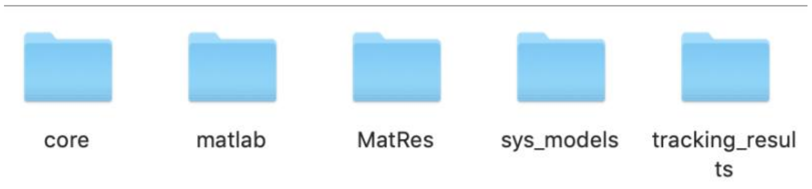


Generalized coordinate system, used in this Code.
States **X=[**$q_0,\ q_1,\ q_2,\ q_3,\ \dot{q}_0,\ \dot{q}_1,\ \dot{q}_2,\ \dot{q}_3$ **]**

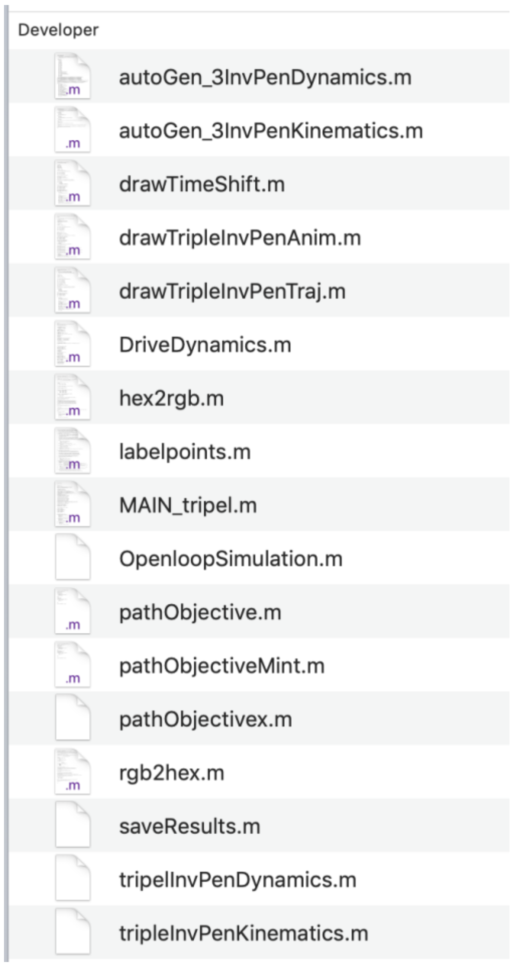
triple_pendulum Directory

The main directory of the project (**tirple_pendulum**) contains these sub directories:



Matlab Directory

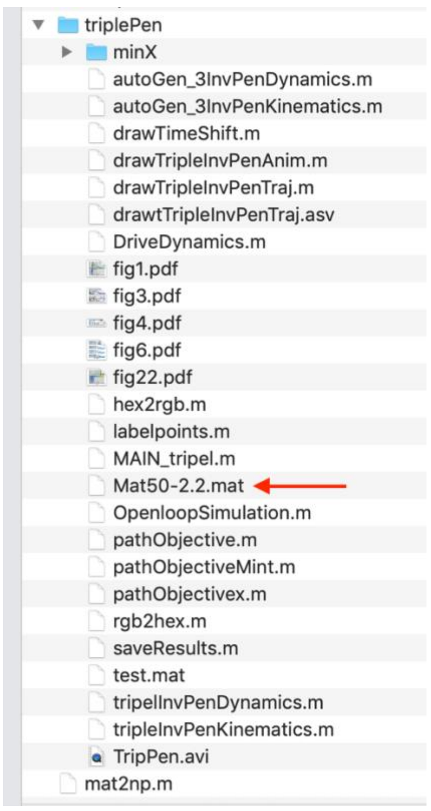
Matlab directory contains **OptimTraj**, which is a Library we use to calculate the Trajectory, and a directory called **triplePen**. **triplePen** should contains these files:



Developer

autoGen_3InvPenDynamics.m
autoGen_3InvPenKinematics.m
drawTimeShift.m
drawTripleInvPenAnim.m
drawTripleInvPenTraj.m
DriveDynamics.m
hex2rgb.m
labelpoints.m
MAIN_tripel.m
OpenloopSimulation.m
pathObjective.m
pathObjectiveMint.m
pathObjectivex.m
rgb2hex.m
saveResults.m
tripelInvPenDynamics.m
tripleInvPenKinematics.m

MAIN_triple.m : the main matlab file, that produce the trajectory and plots. The resulting trajectory will be saved in the same folder as a .m file (example: Mat50-2.2.mat) which later could be used in python main for tracking. Copy this file to the MatRes manually if you want to use it for tracking. You can change the file name in save section in MAIN_triple.m. The Name convention is Mat + Number of Segments of the Trajectory + "_" + the value of finalTime.upp.


triplePen
minX
autoGen_3InvPenDynamics.m
autoGen_3InvPenKinematics.m
drawTimeShift.m
drawTripleInvPenAnim.m
drawTripleInvPenTraj.m
drawtTripleInvPenTraj.asv
DriveDynamics.m
fig1.pdf
fig3.pdf
fig4.pdf
fig6.pdf
fig22.pdf
hex2rgb.m
labelpoints.m
MAIN_tripel.m
Mat50-2.2.mat
OpenloopSimulation.m
pathObjective.m
pathObjectiveMint.m
pathObjectivex.m
rgb2hex.m
saveResults.m
test.mat
tripelInvPenDynamics.m
tripleInvPenKinematics.m
TripPen.avi
mat2np.m

(see the comments in every section of the **MAIN_triple.m**, for more details about functions used there)

ATTENTION: make sure you are in a triplePen directory as you run **MAIN_triple.m.**

DriveDynamics.m : Symbolic derivation of Equations of motion. The results are saved in **autoGen_3InvPenDynamics.m** and **autoGen_3InvPenKinematics.m** (Think of them as a result of lamdify() on sympy expressions in python!)

pathObjective.m : the Cost function which be used in Trajectory design.

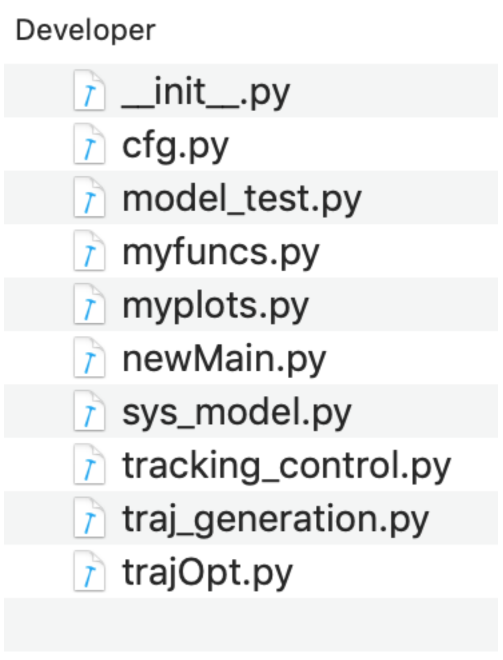
pathObjectiveMint: you could use this instead of **pathObjective.m** to produce a trajectory with the minimum time.

pathObjectivex.m : use this instead of **pathObjective.m** to produce a trajectory with minimum horizontal movement of cart.

OpenloopSimulation.m: use this to test the calculated trajectory in open loop control. (closed loop is implemented in python. see **newMain.py**)

Core Directory

core: This folder contains following python files:


Developer

```
__init__.py
cfg.py
model_test.py
myfuncs.py
myplots.py
newMain.py
sys_model.py
tracking_control.py
traj_generation.py
trajOpt.py
```

cfg.py : this is just a container, which be used to store the parameters and results.

newMain.py: the main file that we run to generate model, tracking control and plotting the results. The following steps need to be taken: (see also the documentation of each function /class for details about inputs and outputs)

1. run **Pen_Container_initializer(3)** method (3 indicates the number of pendulums)

2. define a variable **ct= cfg.pendata** you need it to path to different functions. It is just a container to store/load results instead of using many inputs and outputs for every functions (you can think of it as a **self** in python class!).

3. Use one of the methods below to generate / load the model:
   - **system_model_generator()** → to generate a single model without any deviations
   - **generate_sys_model_with_parameter_deviation()** → to generate a model with parameter deviation or alternatively use one of thease functions:
   - **paralleliyed_model_generator() , load_sys_model()**

4. define the function called dynamics which returns system dynamics (see the example in newMain.py)

5. create an object of the class **TrajOptimization()**
   example: **traj= TrajOptimization(**dynamics=dynamics**)**

6. use the method **traj.covertGridsTofunc (**"name of the matlab trajectory stored in **MatRes**"**)**. It reads the matlab file containing trajectory data and converts collocation points to splines. the result is two functions for states and input.
   example: **xFunc, uFunc= traj.convertGridsToFunc(**'Mat50-22.mat'**)**

7. use **parallelized_tracking_control()** to generate tracking results for multiple models or different deviations you need to use this. there are different examples for generating different types of trajectories. (with parameter deviations or with boundry(state) deviations)

8. if you just want to simulate tracking for a single model (for example original model without any boundary(state) deviations you have to use **_tracking_contorl()** (see the example implemented in **newMain.py**)

9. use **myplot()** to plot the results. In this function you should specify the trajectory name which is also the name of the folder, we use to store the tracking results in **tracking_results.** Then specify the deviation_list which could be 'None', 'parameter', 'boundry' or 'both'. you should also change model_types , xt_keys and qr_keys .
   (you find details about deviation_list , model types, xt_keys and qr_keys in **_tracking_control()**)

ATTENTION: for the code to be able to find the proper folders to read/save files you need to run the **newMain.py** in main directory **triple_pendulum** or in **core.**

Model_test.py: compares model generated with kanes method, with the model generated with lagrange method in matlab, and another model generated in python directly!

myfuncs.py : contains all of the functions needed to generate model, for tracking control etc.

sys_model.py: generates model with kanes method and store the result in **sys_models.**

trajOpt.py: generates trajectory optimization in python directly (not completely implemented) or use trajectory that created in matlab. It should be saved in **MatRes** folder.

tracking_control.py: contains two functions(_tracking_control, and parallelized_tracking_control) which do tracking. As the name suggest the second one can do simultaneous tracking for different models or parameters. (see newMain.py)

myplots.py: reads the tracking results and plot the results.

Other Directories

MatRes: MatRes folder contains the trajecotries, which produced with optimTraj. You should copy your desired Trajectory manually to this folder. (see **MAIN_triple.m**)

sys_models: contains model files (original model or models with parameter deviations etc) this folder will be generated automatically and contains every models you created in python. You can load your model form this folder instead of creating them each time you want. (see **Core** folder for further description)

tracking_results: This folder is generated automatically and will be used each time you use tracking. (see **Core** folder for further details)