Need to ask Shawn if I should be storing Attitude data in separate file or in the same csv as everything else is recorded in.

- From a creating the dataServer in the EQVIO I think it is much easier to have it as a separate file.
- Not sure how hard it would be to modify the recording script but I doubt it is to hard

Also ask Shawn about if I need to include functionality for simulation.

Should I include some sort of if statement based on a setting to add or remove the ability to use attitude innovation. If so is there a cleaver way of doing this.

- Would need at least one if statement in the APDatasetReader to make sure we don't look for a file that doesn't exist
- Probably will need an if statement in ThreadedDataServer.cpp to make sure that getAttitude does not cause any issues

Check with Shawn that the measuredresidual variable is $\log(R_{p0}^T R_P R_{\hat{A}}^T)$, L221 of VIOFilter.cpp. Or can I just get rid of the measuredresidual variable and directly define Ytilda as $\log(R_{p0}^T R_P R_{\hat{A}}^T)$. Also what type would this be?

Good

Maybe/maybe not

Definitely has issues

Data Server

DataSetReaderBase.h:

- L29 add another measurement type "Attitude"
- Add another virtual function called nextAttitude similar to nextImage
    - Had to add a data class called StampedAttitude

APDatasetReader.h:

- Overide the function defined in DataSetReaderBase.h similar to L36 and L37 but for nextAttitude
- Add in a CSV file for attitude

APDatasetReader.cpp:

- Add in the nextAttitude definition. This is implementation specific and will change if I have a separate file for attitude or if it is a part of the mav_imu.csv
    - Added in a setting called UseAttitudeInnovation into the settings file at this point
    - Made changes to VIOFilterSettings.h at this point
    - ***Don't know how to get the setting into APDatasetReader.cpp properly. Will currently just always look for that file.***
    - Added in the line that links AttitudeCSVFile and attitude.csv in APDatasetReader. Not sure if I got this correct, needs testing.
- Added in the nextAttitude() definition
    - This is based of the nextIMU()
    - ***Need to get help with this, L82***

ThreadedDataServer.h:

- Create all the necessary protected variable for the getAttitude()
- Need to create the queue for attitude, similar to L49
- Possibly will need to add a new class like IMUVelocity in L48
- Add the getAttitude() to the public functions

ThreadedDataServer.cpp:

- There is a lot of stuff that needs to be done in this file.
- Will need to modify:
    - nextMeasurmentType()
        - Modified the three different things
        - Had to change the conditions around when a certain measurement should be returned so that it worked for three measurements.
        - May need to add in some logic to see if one of the queues is empty rather then if two or if none are empty
    - nextTime()
        - Just added in another if statement for the attitude
    - fillQueues()
        - Copied and pasted in the attitude file queue.
    - queuesFilled()
        - Just added another logic statement for attitude. Copied and pasted from image/IMU
- Will also need to add in a new version of getImage/getIMU() for getAttitude()

- This was just compying a pasting it in with little to no changes

DataServerBase.h:

- This will need to be changes so that it has the appropriate functions defined
- I don't think that the .cpp file will need to be changed

SimpleDataServer.h:

- This needs to be changed so that it has the appropriate functions defined
  - Added in necessary functions declarations

SimpleDataServer.cpp:

- Modified nextMeasurmentType() to be correct.
  - Had to change the logic significantly here as it was a binary choice rather then allowing for three choices.
  - Also had to allow for the case where one queue might be empty
- Added getAttitude function. This is copied and pasted
- Corrected nextTime to allow for attitude times
- Corrected SimpledataServer to have attitude

Filter

Main_opt.cpp:

- Will need to add an else if (measureType=Attitude) with the appropriate code following
- May need to add in some loop timer stuff

VIOFilter.h:

- Will need to declarer in the processAttitudeData function

VIOFilter.cpp:

- Will need to define processAttitureData()
  - This will be based on the processVisionData with most of the timing stuff removed
  - Check with Shawn that the measuredresidual variable is log $(R_{p0}^{T} R_P R_{\hat{A}}^{T})$, L221 of VIOFilter.cpp

EqFMatrices.cpp:

- Will need to add in some sort of function that generates the correct sized C matrix for an attitude adjustment

EqFMatrices.h:

- Add new function defined in EqFmatrices.cpp

VIOFilterSettings.h:

- Need to add in the setting for turning on and off the attitude innovation
-