Slosh Testing

This repository includes the following scripts and functions.

MATLAB Scripts:

- `calibration.m` : Finds the K96 and inverted K66 matrices to convert sensor voltages to system's forces and torques. This is done using the LabVIEW data collected from the 3 three-axis force sensors after performing the static system calibration of the system. For more information, see the 'calibrationREADME' file.
- `postprocessing.m` : This script reads LabVIEW CSVs containing the 9 voltages corresponding to the three axis of the three Kistler force sensors and uses the K96 or K66 matrix to calculate the net forces and torques of the system. Then it reads the empty tank data and finds the slosh forces caused by the liquid. This script also creates plots for position, acceleration, forces and torques. For more information, see the 'postprocessREADME' file.

MATLAB Functions

- `diagramcreate.m` : Plots a top-down view of the position of the force sensors on the triangular plate and shows the 3 forces acting on each sensor. It also shows the non-calibrated net forces and torques of the system. It takes the following inputs:
    - (Array) **fdata**: Array including the forces from the force sensors in the format [Fx1,Fx2,Fx3,Fy1,Fy2,Fy3,Fz1,Fz2,Fz3]
    - (Vector) **startpts**: Start points of the load steps.
    - (Vector) **endpts**: End points of the load steps.
- `filter1.m` : Filters force data using a Type II Chebyshev filter and returns the filtered data. It takes the following arguments:
    - (Float) **cutoff**: Cutoff frequency of the filter in Hz.
    - (Int) **order**: The order of the filter.
    - (Int) **fs**: Sampling rate in Hz.
    - (Vector) **dataVec** and the detrended data as arguments.
- `fitline.m` : Creates a linear fit of the s-sensor force data and Kistler sensors force data and returns the slope as K coefficients. It takes the following arguments:
    - (Vector) **x**: S-Sensor force data.
    - (Vector) **y**: Kistler sensors force data.
    - (Vector) **weights**: Weights for the fit.

    - (Boolean) **plotBool**: Whether to create plots of the linear fit.

- `getCalibratedLoadsK96.m` : Returns the time array, and an array with the 6 calibrated loads (Fx, Fy, Fz, Tx, Ty, Tz) using the 9 sensor forces (which it filters using `filter1.m`) and the K96 matrix. It takes the following arguments:

    - (Array) **K96**: 9x6 Matrix obtained with `calibration.m` .
    - (String) **filePath**: Path to the CSV file obtained with LabVIEW.
    - (Vector) **filterParams**: Parameters of the filter in a vector ([Sample Rate in Hz, Cutoff Frequency in Hz, Order of the filter]).
    - (Float) **sf**: Scaling factor to apply to the 9 forces before using them with K96 (if necessary).

- `getCalibratedLoadsK66.m` : Returns the time vector, an array with the 6 calibrated loads (Fx, Fy, Fz, Tx, Ty, Tz) using the 9 sensor forces (which are converted to preliminary loads) and the K66 matrix, and the acceleration vector from the accelerometer. If there is no accelerometer data, it returns an empty string ("") for acceleration. It takes the following arguments:

    - (Array) **K66**: Inverted 6x6 Matrix obtained with `calibration.m` .
    - (String) **filePath**: Path to the CSV file obtained with LabVIEW.
    - (Vector) **filterParams**: Parameters of the filter in a vector ([Sample Rate in Hz, Cutoff Frequency in Hz, Order of the filter]).
    - (Float) **sf**: Scaling factor to apply to the 9 forces before using them with K66 (if necessary).

- `getThFreqAccDoubleAmpFill.m` : Returns a vector containing the target frequency in Hz, acceleration in g's, double amplitude in mm, and tank fill percentage. This is achieved by reading an Excel spreadsheet named "Filenames.xlsx" that contains the tabulated names of the tests according to these parameters. It takes the following arguments:

    - (String) **fileName**: Name of the test (e.g., "test124").
    - (cell) **fileNamesCell**: Cell containing the data read from "Filenames.xlsx".
    - (Boolean) **logParams**: Whether to print to the console the parameters of each file that is being processed (true/false).

- `findPeaks.m` : An "improved" version of the MATLAB's built-in `findpeaks` function. It uses `findpeaks` to find both high and low peaks. It can take in a vector or an array, and it will return a cell array including the location and magnitudes of the high and low peaks.

    - (Vector/Array) **ftArray**: Vector/Array of the force(s) and/or torque(s), or any other dataset.
    - (Int/Float) **thFreq**: Target frequency of the test in Hz, obtained with `getThFreqAccDoubleAmpFill.m` .
    - (Int) **sr**: Sample rate in Hz.
    - (Boolean) **plotCycles**: Whether to plot the dataset showing the position of the peaks.

- `getSSFilledTankLoads.m` : Returns the loads of the entire system (filled tank) trimmed from the first peak to the last peak of the portion considered as the `ssPercentage` set in line 4 of `postprocessing.m` . It takes the following parameters.

    - (Array) **filledTankLoads**: 6-column array containing the three forces and three torques trimmed according to `ssPercentage`.
    - (Int/Float) **freq**: Target frequency of the test in Hz.
    - (Int) **sr**: Sample rate in Hz.

- `getEmptyCyclesAmplitudes.m` : Returns the peaks (high and low) of the empty tank loads, and the empty loads dataset. It takes the following parameters.

    - (Array) **K96**: 9x6 Matrix obtained with `calibration.m`.
    - (Int/Float) **theFreq**: Target frequency of the test in Hz.
    - (String) **emptyTest**: Path to the empty tank CSV file.
    - (Vector) **filterParams**: Parameters of the filter in a vector ([Sample Rate in Hz, Cutoff Frequency in Hz, Order of the filter]).

- `getEmptyTankLoads.m` : Creates and returns an array containing the dataset for the loads of the empty-tank test based on the amplitudes obtained with `getEmptyCyclesAmplitudes`. It takes the following parameters.

    - (Vector) **maxA**: Vector including the maximum peaks (returned value from `getEmptyCyclesAmplitudes`).
    - (Vector) **minA**: Vector including the minimum peaks (returned value from `getEmptyCyclesAmplitudes`).
    - (Int/Float) **freq**: Target frequency of the test in Hz.
    - (Int) **sr**: Sample rate in Hz.
    - (Int/Float) **duration**: Duration of the dataset in seconds. Determines how many data points to create.

- `reshapeEmptyData.m` : Returns the empty tank data array trimmed to the same size of the filled tank data array (with peaks aligned), and the start and end indices of where it's being trimmed. It takes the following parameters.

    - (Array) **emptyData**: Array containing the empty tank single cycle data repeated to be greater or equal in size to the filled tank data.
    - (Array) **filledData**: Array containing the filled tank data. When used in `postprocessing.m`, this parameter is set to the 'Steady State' loads, which is x% of the total cycles that are considered to be in steady state.
    - (Int/Float) **theFreq**: Target frequency of the test in Hz, obtained with `getThFreqAccDoubleAmpFill.m`.
    - (Int) **sr**: Sample rate in Hz.

- `getPositionData.m` : Returns the time and position vectors corresponding to the acceleration data obtained by the linear encoder of the linear stage. It takes the following parameters.

    - (String) **filePath**: Path to the txt file containing the time and position data.

- `createReportPlots.m` : Creates a total of 14 plots and stores them as png images in "./Plots/Report/{TestNumber}". It takes the following parameters.
    - (Vector) **posT**: Time vector (in seconds) corresponding to the position vector.
    - (Vector) **position**: Position vector (in meters).
    - (Vector) **acceleration**: Acceleration vector (in g's). Use an empty string ("") to calculate the acceleration based on the position vector.
    - (Vector) **ssT**: Time vector (in seconds) corresponding to the 'Steady-State' loads.
    - (Array) **ssLoads**: 'Steady-State' loads. This is obtained after using `getSSFilledTankLoads` .
    - (Array) **emptyLoads**: Empty tank loads. This is obtained after using `getEmptyTankLoads` and `reshapeEmptyData` .
    - (String) **testName**: Name of the test. Ex: "test123".
    - (Int/Float) **fillLevel**: Fill percentage of the tank for the corresponding test.
    - (Int/Float) **thAcc**: Theoretical acceleration (in g's) of the test.
    - (Int/Float) **thFreq**: Theoretical frequency (in Hz) of the test.

Python Scripts

- `docBuilder.py` : Creates a Word (.docx) document with the images produced by `postprocessing.m` (store in './Plots/Report/{testName}') and the screenshots of the tables of the test names inside './TableSCs/'.
    - This script requires `pandas` and `python-docx` . To install them run:
        - `python -m pip install pandas python-docx`
    - A Word document titled "Report_Plots.docx" is created using all the images obtained with `postprocessing.m` . For the 'zoomed' images, it also includes text mentioning the zoomed portion. To edit the zoomed portion text, edit line 51.
    - The script will take a few minutes to finish executing since it adds 14 images per test.
    - To run the script:
        - `python docBuilder.py`