# Introduction

The package provides feature extraction for vibration signals. The idea is that this package will be extensible enough to support nX vectors, generic vibration signal, unit-specific signals (acceleration, velocity, PV curves, etc.). I believe the best way to provide this structure is to us a class-based OOP approach. This document provides an overview of the structure and comments explaining the rational for the organization.

# Class UML diagram

Figure 1 shows the class UML diagram for the package.

Figure –UML class diagram (methods shown)

## ClSigFeatures

This object contains a generic signal (voltage, acceleration, whatever), methods to extract features from the signal, and the extracted features themselves. Need to decide how to structure the features. For example, the 1X, 2X,… vectors can stand on their own and may not be derived from a signal so I think these features need to be their own class that is extended into the cl\_sig\_features class. A feature like amplitude (Direct)) could stand alone (could be derived by recording a DMM output for example) so these probably warrant their own class. Thinking through this, I believe the features types will need their own abstract base class. Some of these include the following.

## CLSigReal

Real-valued features (direct, pk-pk, bias, mean, kurtosis, etc.). The values could be calculated on a sample-by-sample basis (i.e. amplitude) or could be cyclo-stationary real-valued signals. This class would be used to contain real-valued features extracted from a defined sub-set of sample in the signal. For example, air-gap monitoring on hydro requires the minimum and maximum distance between stationary and rotating poles to be calculated after each revolution is complete. Likewise the peak rod load for a reciprocating compressor can only be calculated after each revolution is complete.

### Workflow: Estimating trigger event times

The class supports methods for estimating trigger event times, given a signal, the threshold, and the hysteresis. The method construction allows the user to

## CLSigComp

Complex-valued (i.e. vector) features (1X, 2X, nX, etc). These could be calculated on a sample-by-sample basis (i.e. poles/zeros of a vibrating mechanical system) or cyclo-stationary (i.e. the 1X/2X for each revolution).

## ClSig

This abstract base class hold methods and data to provide structure for the signals. A few of the key fields and methods are discussed below.

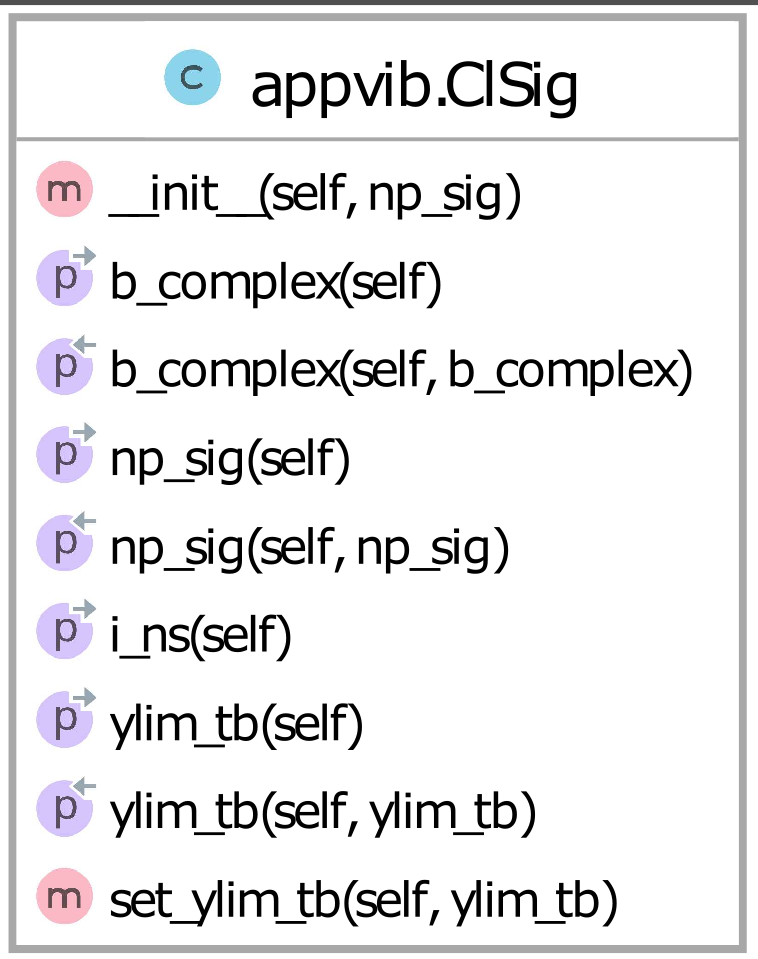


Figure - Methods in ClSig class

## b\_complex

The motivation in creating the **ClSig** class is to create a base class that can be abstracted into either a real or complex class. The **b\_complex** field is used to identify either real- or complex-valued arrays. It must be set in the derived class.

## np\_sig

This field contains the numpy array, either real- or complex-valued depending how the derived classes use it.

## ylim\_tb

It may seem odd to have the y-axis plotting limits stored in this class since it initially appears to only apply to real-valued signals. For vibration analysis it is comment to extract the magnitude of the complex-valued signals (for example, the magnitude of a 1X vector) and plot that. I am thinking that when this class is extended into the **CLSigComp** the **ylim\_tb** will be used to store limits for the magnitude plot.