HSE·AG

Exploratory Data Analysis & Visualization

After clean-up and noise reduction we have already seen some curves that seem to behave differently than all the others. A closer look shows some slight different behaviour of a bunch of curves around 7s. It looks like there are two groups of curves in that region. When signals have steep slopes, it will be more difficult to fit a smoothed curve due to large y-changes. Let's use a model $y(t) = s(t) + n(t)$, where $y(t)$ is the measured signal as function of time, modelled as sum of a true underlying signal, s(t), and some noise, n(t). Then, the estimated signal $\hat{s}(t) = smooth(y_i = y(t_i), i = 1, ..N)$ obtained with a (suitable) smoothing filter with the set of our measured data points y_i at times t_i. The error residuals $r_i(t_i) = y(t_i) - \hat{y}(t_i)$ are then a kind of noise measure. At steep slopes this error residuals will get usually larger, particularly when there is some inaccuracy of the sampling times t_i. Therefore, when there are large residuals (noise values), there is normally also more noise variance and the signal estimation is less precise. As our goal is to find over all times a good signal estimate, we exclude large noise variations at the end (or, we might split up the signal in sections where the noise variations are similar and process these sections separately). For now, we will exclude the data points beyond 9.46s. The reason is that the pumping out phase with large and very rapid changing values starts around 9.5 seconds, so it is not desired to include those points in the PCA, otherwise the large variational parts during this phase would account for most signal variation, obscuring smaller variations in prior sections. Therefore, only the first ''idxCutOff'' (473 in this example) points (= 1point * 50Hz * 9.46s) are used.

Update: As there are data sets with a much longer aspiration phase, dynamic cutting off may be used. For this, the initial pressure is determined and as cut-off level the 1.1*99-percentile used to determine the time index when data will be cut off.

Principal Component Analysis (PCA)

In (statistical) computing packages like MATLAB or R, PCA algorithm is already available as library function. So, here we concentrate on our goal of finding a good data model described by a few principal components that are linearly uncorrelated and that capture most of the data bundle of time-varying pressure signal's variance. This is done by projecting the data to the sub-space spanned

by the first k principal components. As PCA is a linear method where the data matrix needs to be zero mean, achieved by subtracting the column mean, assuming standard data format.

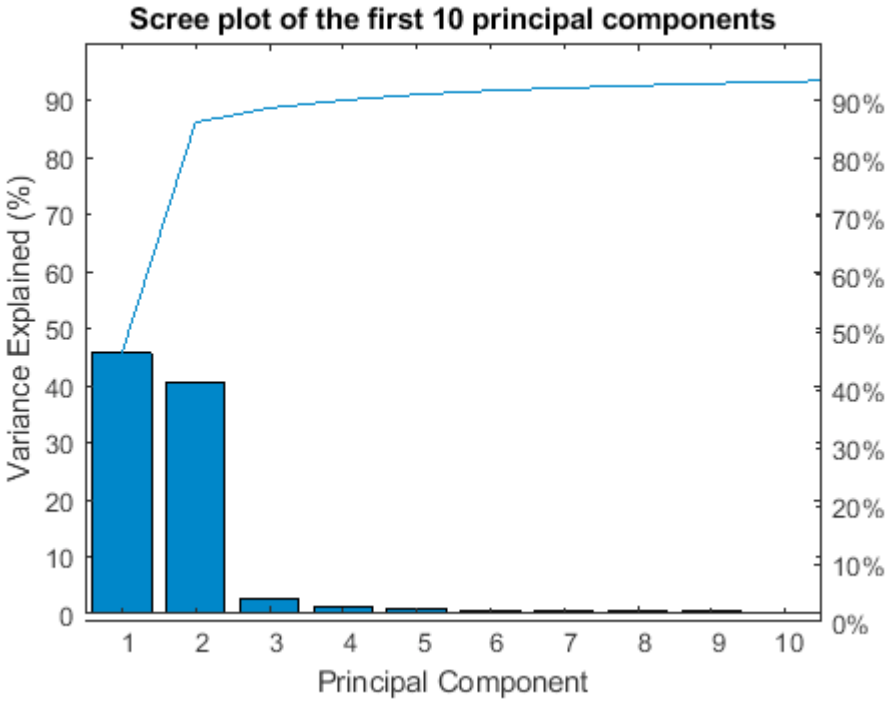
For a more mathematical formulation and lot's of background see https://en.wikipedia.org/wiki/Principal_component_analysis.

Warning: Columns of X are linearly dependent to within machine precision.

Using only the first 107 components to compute TSQUARED.

PCA Approximation

To check whether the PCA reasonably describes the signals, a scree plot of the first 10 principal components is made. The scree plot shows graphically how much variability of the original signal data are captured by the first k components. Bars show the amount of variability captured by the corresponding component and the cumulative sum, shown by the line shows the total variability captured.



The scree plot shows that over 85% of variability is explained by the first two principal components.

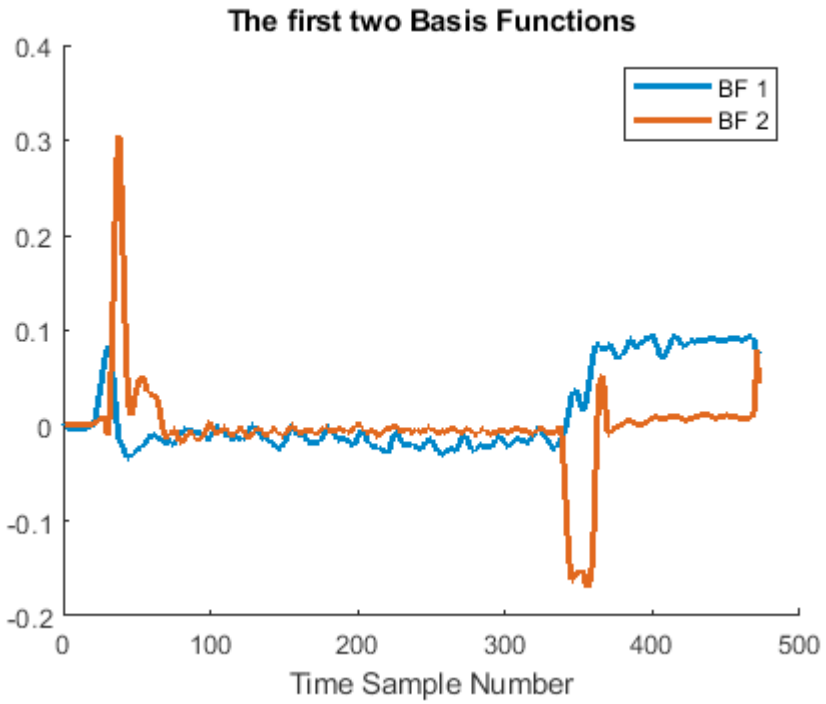
Sparse Signal Reconstruction

If signals can be approximated by a sparse representation, it can be seen as building a model with only a few components, just from data. An explicit signal approximation can be done by reconstruction with only two principal components. When doing a signal reconstruction, the linear sub-space spanned

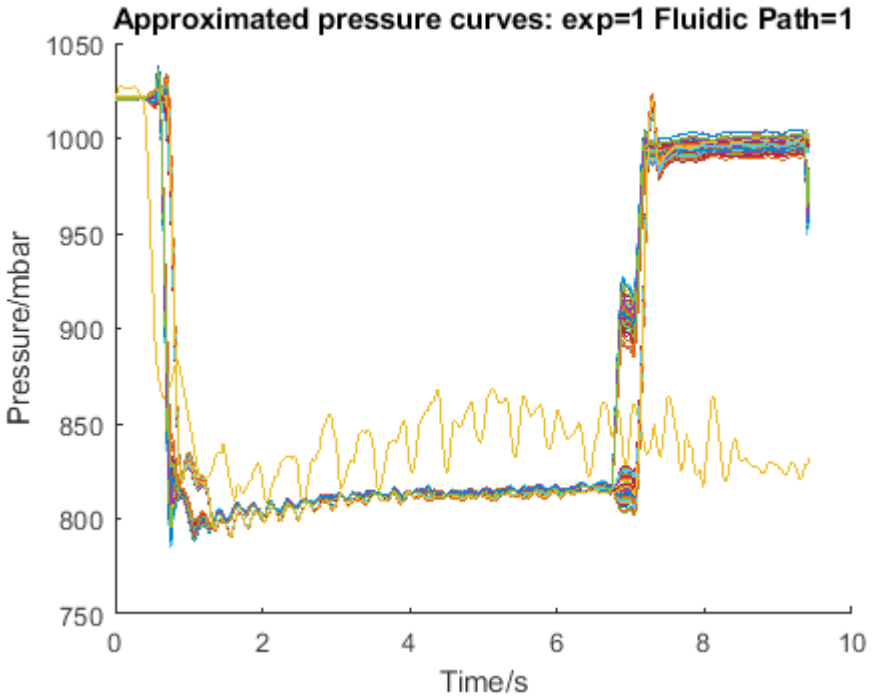
by the first k principal components is described by a linear combination of the basis-vectors represented in the original space, i.e. where the original signal data lie. In our case it is the 472-dimensional space containing our 110 signal vectors, each 472 elements long. Let the input to the PCA algorithm be a $n \times p$ data matrix X (n: number of observations, i.e. one row contains one observation given is the complete time-signal of length p). Then the maximum number of possible principal components is $K = min(n, p) - 1$, where the -1 is due to the mean removal. Geometrically the PCA is a rotational tranformation of the orignal, mean-subtracted data matrix, such that in the direction of the first component the signals have maxium variability and in the direction of the second component the signals have again largest variation, given the selection of the previous components and so forth. The outputs of the PCA are a $p \times K$ matrix of principal component coefficients (also called loadings), where the k-th column corresponds to the k-th principal component and is the corresponding basis-function, reprented in the original p-dimensional space. Another output, the $n \times K$ matrix of principal component scores yields the weights for the linear composition of basis-functions to approximate the orignal signal. The k first elements of $i - th$ row of the score matrix are the weights for the first k basis-functions and the weighted sum plus the original signal mean is the signal approximation with the first $k% principal components.

The first two basis-functions look as follows

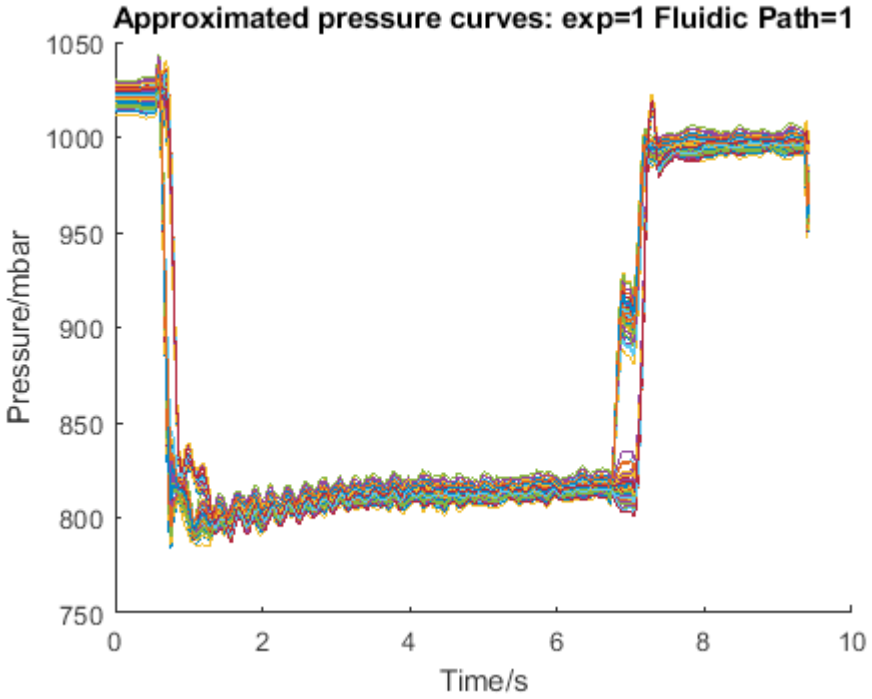
```
#pricipal components for reconstruction: 2
```



All the reconstructed signals with two principal components look as follows

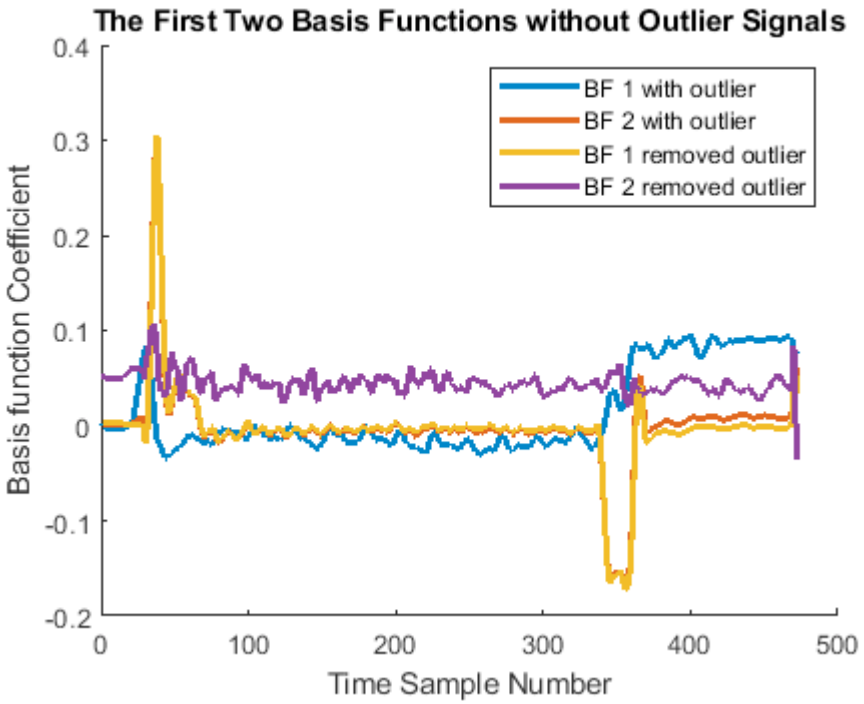


which isn't too bad compared to the original signals, but clearly most of the variability is used to reproduce the yellow outlier signals. Let's remove them and do the PCA reconstruction again.



As predicted the approximation is now much better, i.e. we have a better model as we have less variability in the data by removing the outlier curves.

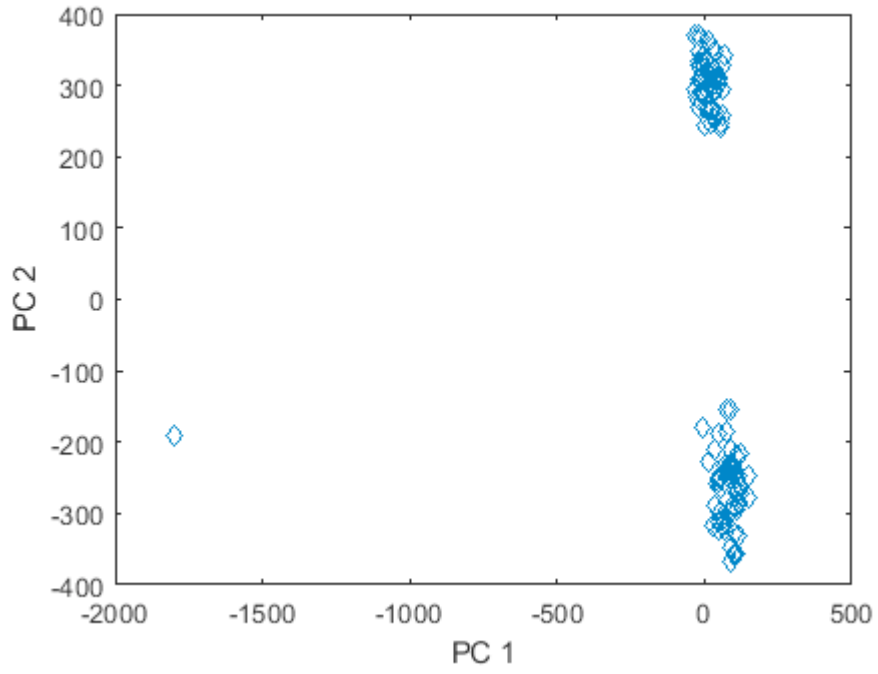
And as we have now less signal variation by excluding the outlier signal, also the basis functions become better suited to represent the remaining signals



When leaving away the outlier signals in the data matrix X, the basis functions start to capture more the local variations in the signal than larger trends due to the big differences, clearly seen after the

Low Dimensional Visualization

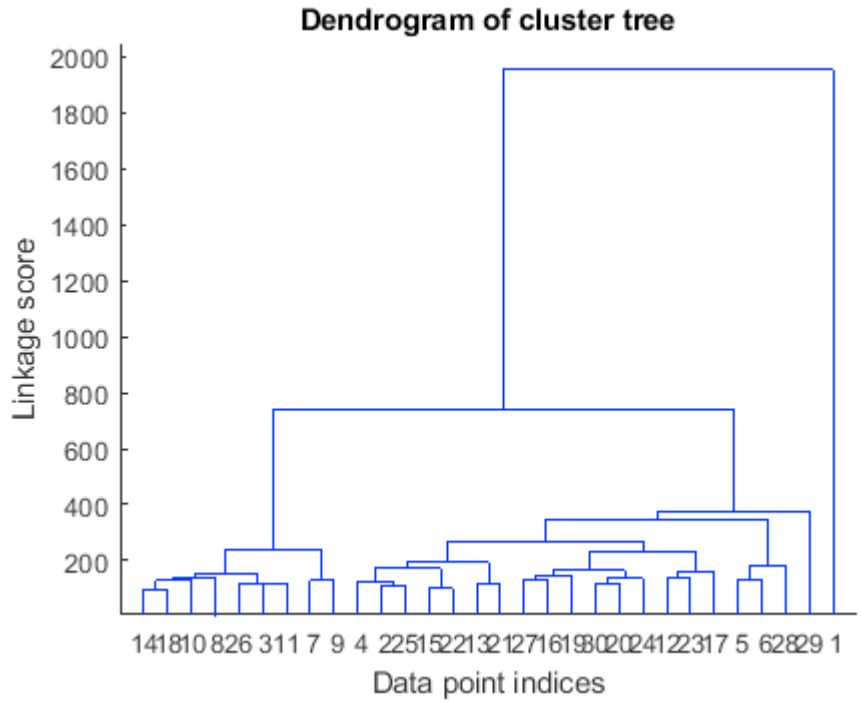
As was seen with PCA we can reduce our over 472-dimensional time signals (each sampling time-step adds one vector dimension) to only 2 dimensions, namely the 2-dimensional subspace spanned by the first two principal components.



Clearly, there are three clusters visible. A simple kNN (k nearest neighbour) clustering could partition this two-dimensional subspace as it is obvious by visual inspection that there are three clusters. However, let's see if we can explore this data structure also with a cluster tree.

Clustering

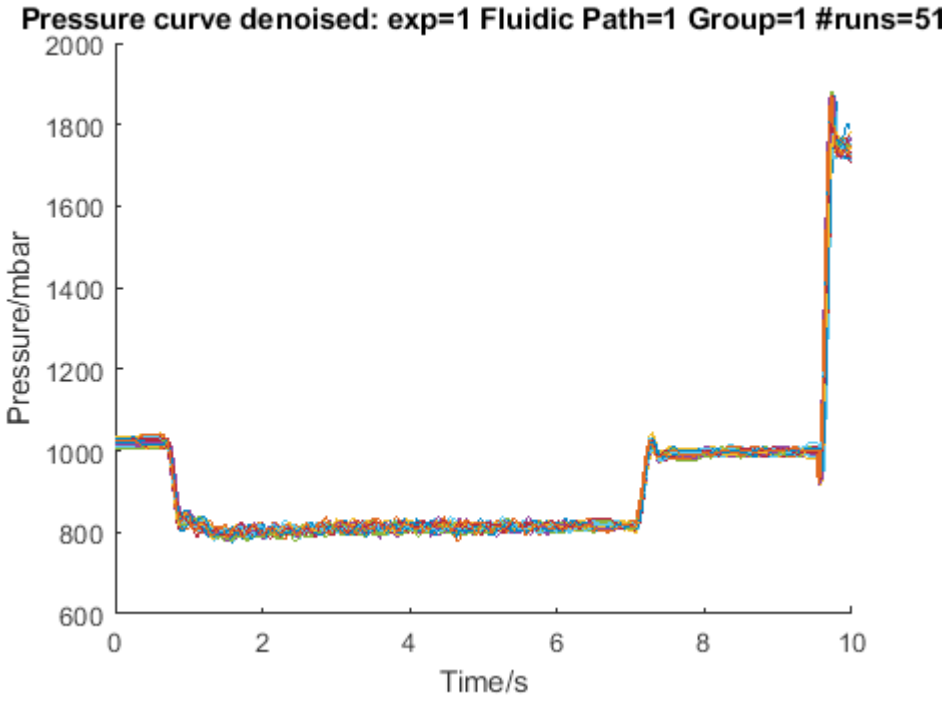
As the data acquisition seems to behave sometimes differently and some curves following distinctively different patterns, clustering is carried out by a cluster tree. The goal is to differentiate these different acquisition modes. Again we take the first 10 principle components (or all available in case there are fewer). For the cluster tree we take the ''Chebychev'' distance metric (maximum coordinate difference) and the linkage method as ''complete'', i.e. the furthest distance from one element to the other elements in the data set and show the dendogram of the cluster tree.

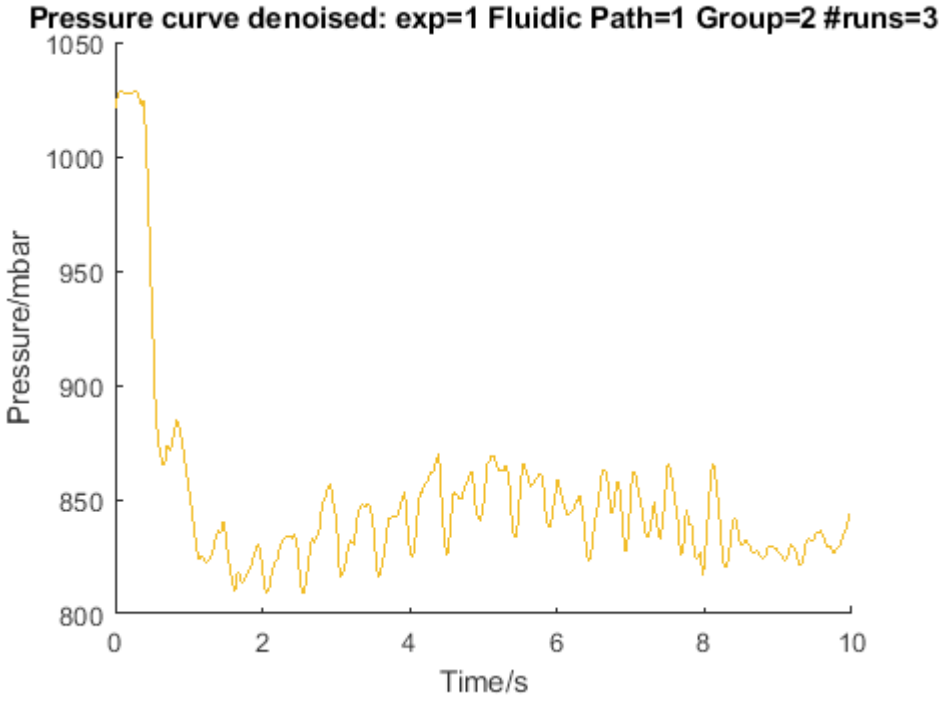


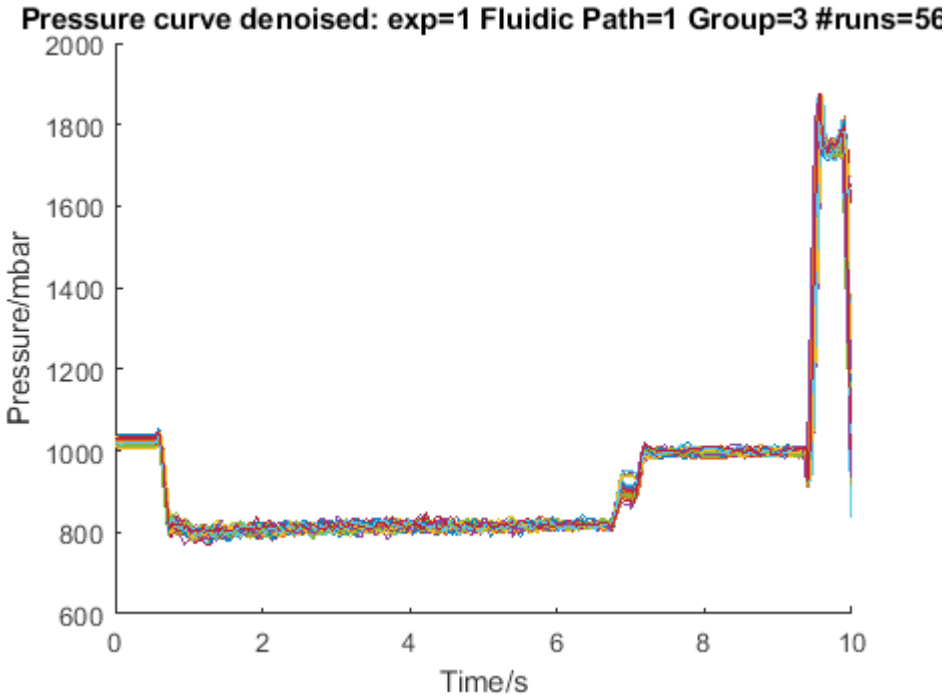
Clustering is carried out by a level-cut, i.e. a simple threshold is applied, which is determined such that 3 clusters are obtained. Graphically, a horizontal line that cuts the cluster tree three times needs to be selected to get three sub-trees to represent the signal groups. This level would roughly be somewhat between 400 and 750.

Using maxium 3 of levels.

HSE·AG

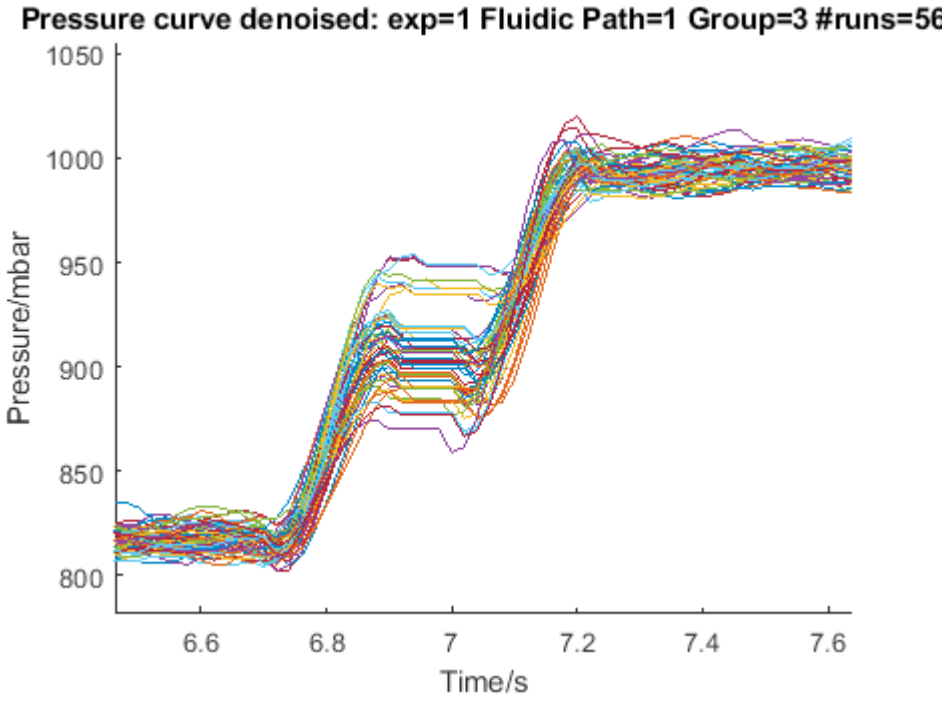

Pressure curve denoised: exp=1 Fluidic Path=1 Group=1 #runs=51
Pressure/mbar
Time/s
2000
1800
1600
1400
1200
1000
800
600
0
2
4
6
8
10

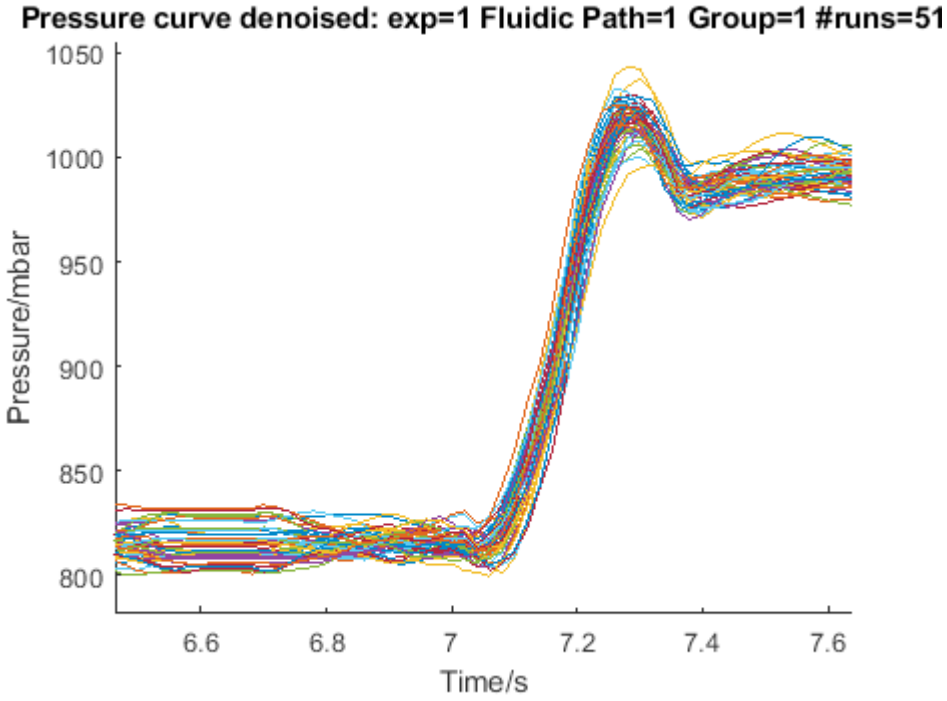

Pressure curve denoised: exp=1 Fluidic Path=1 Group=2 #runs=3
Pressure/mbar
Time/s
1050
1000
950
900
850
800
0
2
4
6
8
10



Three groups can be clearly seen. Group 2 is the priming run (note there are 3 curves on top of each other (indicated by the #runs=3) due to a software bug which recoreded the first signal 3 times. Group 1 and 3 distinguish themselves around the 7-second recording time by a jitter and with a constant measurement phase, which is lacking in group 1.

Enlarging Group 1 and Group 3 around the 7s time stamp shows clear differences that are strongly picked up by the clustering. Again, it turned out to be a software/firmware bug when switching the fluidic path selection valve where for some signals a little delay occurred in 56 cases. Also, note that the delayed signal is constant in this group (Group 3) for about 0.1s just before the 7s mark, another clear indication of a software bug, when the register didn't get updated and appeared as a constant value for a few time samples.





Summary Data Visualization & Data Exploration

This section of data visualization and investigation with some relative simple algorithmic tools has not only shown a parsimonious representation by reconstructing original signals by only two weighted model waveforms (the bases functions weighted by the principal components), how to group similar signals and, last but not least, highlighted that such an analysis can quickly discover software bugs which otherwise may stay undetected for a long-time. This is what

I call "Statistical Software Debugging", which is far more powerful than debugging single memory locations in a C++-Debugger, were one is hardly able to see the next few data points and with today's increasing amounts of data, this is a preferred way of finding root causes, at least as a starting point.

Published with MATLAB® R2018a