

**PSAM 14**

Probabilistic Safety Assessment and Management

16–21 September 2018 • UCLA Meyer & Renee Luskin Conference Center, Los Angeles, CA

UCLA  LUSKIN  
CONFERENCE  
CENTER



# On the Application of Machine Learning Techniques in Condition Monitoring Systems of Complex Machines

Dr.-Ing. Marcin Hinz,

B.Sc. Dominik Brueggemann,

Univ.-Prof. Dr.-Ing. Stefan Bracke

**Chair for Reliability Engineering and Risk Analytics**

Faculty for Mechanical and Safety Engineering

University of Wuppertal, Germany

Kontakt: [m.hinz@uni-wuppertal.de](mailto:m.hinz@uni-wuppertal.de)



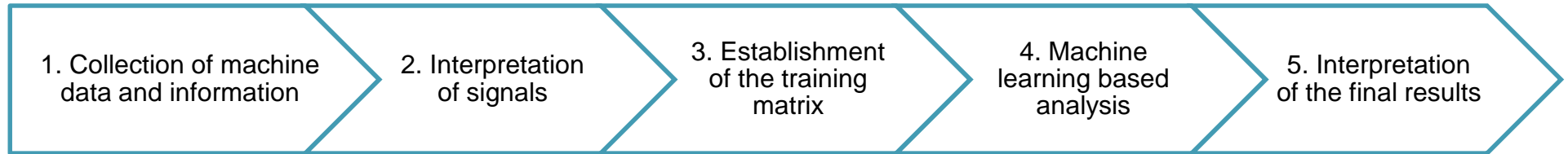
# Agenda

- Introduction
- Condition monitoring system – approach
- Interpretation of the signals – example
- Machine learning based analysis
- Software package for condition monitoring
- Further development

# Introduction

- By knowing the exact condition state of a product, it is possible to:
  - Improve the maintenance (and reliability)
  - Increase the operability of the system (better prediction of failures based on improved understanding of the product)
  - Increase the satisfaction of the customer (higher availability of the product)
- Obviously, in case of safety critical systems, a precise condition state leads to increase of the system safety
- Development of new product market possibilities – pay per X

# Condition monitoring system – introduction



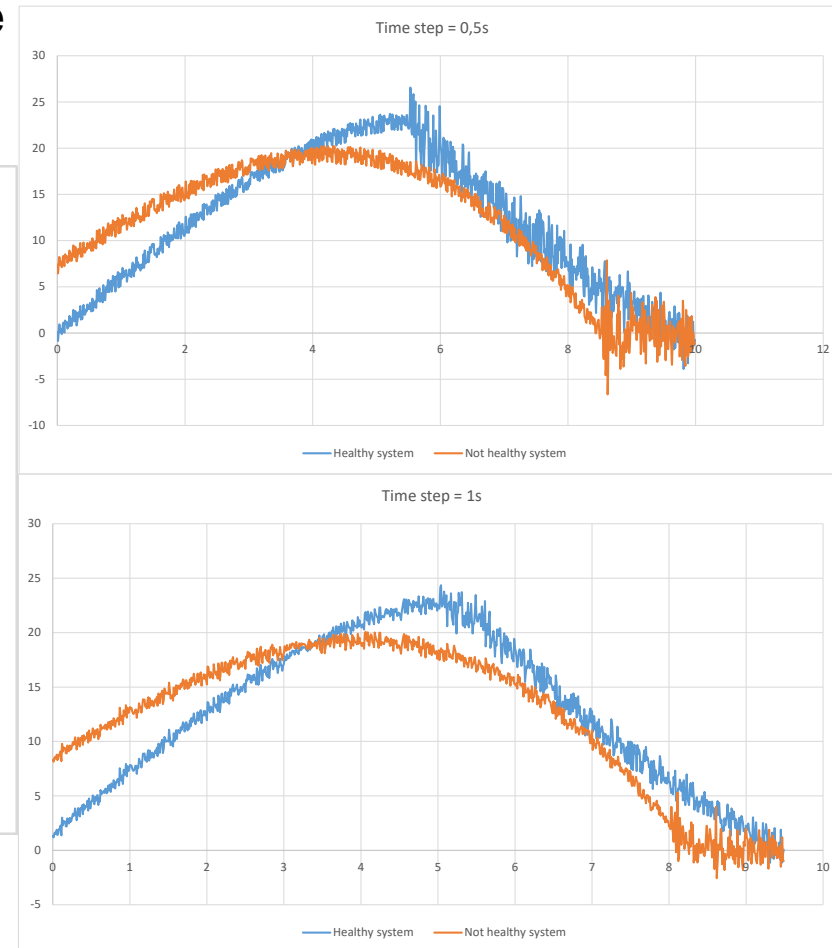
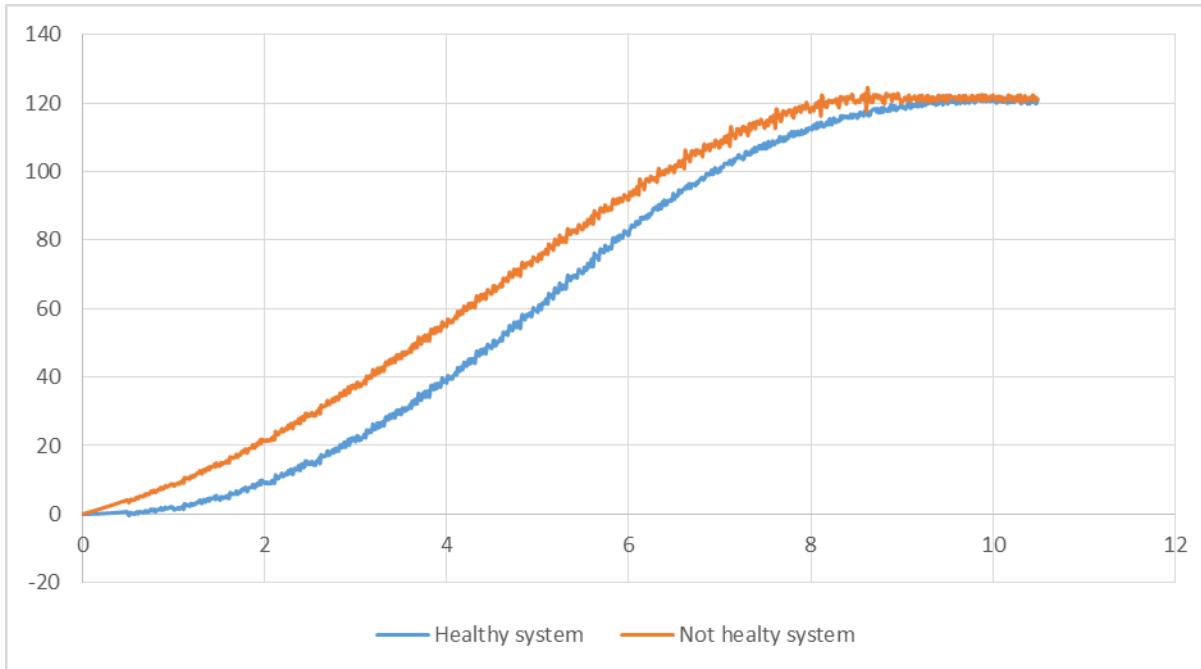
- The data itself can be extracted from the following data sources:
  - Product information in form of a constant data
  - Cumulative data as a single variable
  - Signals representing the change of a variable over time – sensor data
- Data treatment → NA values, missing values, outliers etc.
- Modell fit (incl. model preparation for the prognosis purposes)
- Numerical mathematics (machine learning) → various algorithms (e.g. C4.5, NNge, Neural Networks)

# Condition monitoring system – introduction

- Data interpretation:
  - Standardized statistical and mathematical methods for the analysis of signals like fast Fourier transform (FFT), time series analysis, descriptive statistics (mean, median, dispersion, variance), trend analysis and many more
  - Specified analysis of certain signals in combination with the specific knowledge of a given product like e.g. maximum speed or maximum acceleration of a car derived from the velocity signal
- Possibility of the treatment of unlabeled data → unsupervised learning
- Possibility of integration of additional data sources (e.g. interpretation of images)
- Visualization of final results

# Interpretation of the signals - example

Calculation of the derivative of the signal based on the difference scheme (forward, backward, central)



$$N_{\text{forward}} = \frac{f(x+h) - f(x)}{h}$$

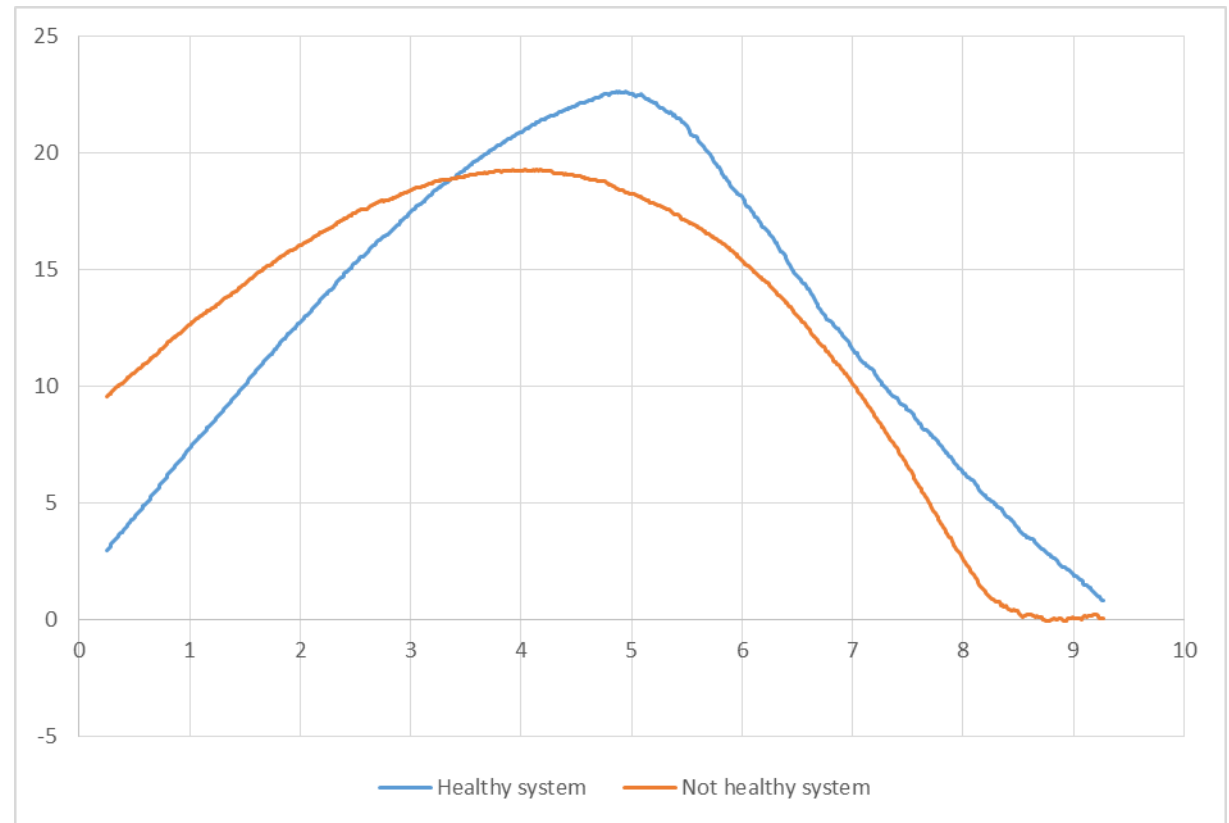
$$N_{\text{backward}} = \frac{f(x) - f(x-h)}{h}$$

$$N_{\text{central}} = \frac{f(x+h) - f(x-h)}{2h}$$

# Interpretation of the signals - example

The differentiation curve has to be smoothed first in order to reach explicit attributes for the description of the shift. For the smoothing of the curve, the moving average can be applied:

$$\bar{p}_S(t) = \frac{1}{n} \sum_{i=0}^{n-1} p(t - i)$$



# Machine learning – state of the art

- **Supervised learning:** The underlying connection between input data and the upfront known target variable is established using training data
- **Unsupervised learning:** Used on unlabeled datasets with no knowledge about the required target variables. Hidden structures within the data are revealed
- **Reinforcement learning:** A sequence of actions gets explored by an intelligent agent that either rewards or punishes certain behavior. The aim is to maximize the cumulative gained reward
- The **C4.5** algorithm is a decision tree classifier that belongs to the class of supervised learning algorithms

$$I(S) = -\log_2 \left( \frac{\text{freq}(C_j, S)}{|S|} \right)$$

$$\text{Info}(S) = -\sum_{j=1}^k \frac{\text{freq}(C_j, S)}{|S|} \times \log_2 \left( \frac{\text{freq}(C_j, S)}{|S|} \right)$$

$$\text{info}_x(TT) = \sum_{i=1}^n \frac{|TT_i|}{|TT|} \times \text{info}(TT_i)$$

$$\text{gain}(X) = \text{info}(TT) - \text{info}_x(TT)$$

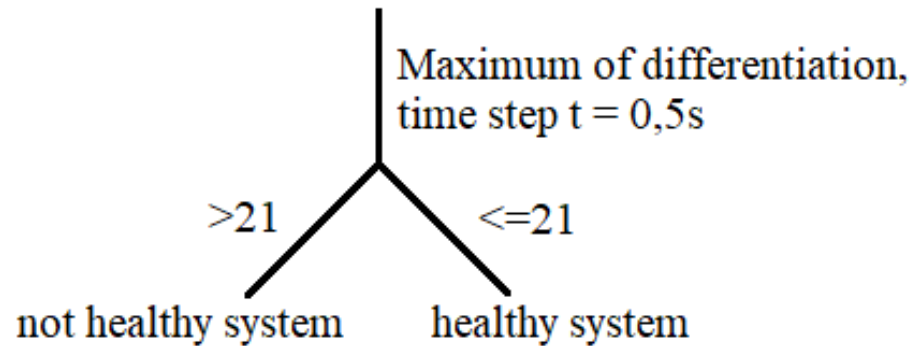


# Machine learning – example

Training data established with the discussed example:

Max differentiate t=0.5s	Max differentiate t=1s	t_max of max diff t=0.5s	t_max of max diff t=1s	Tagret variable
19,37002604	19,28925506	4,13	4,03	Healthy system
23,01009687	22,65621314	5,36	4,89	Not healthy system

A typical result of the machine learning analysis in form of a decision tree and based on the discussed example performed with the C4.5 algorithm



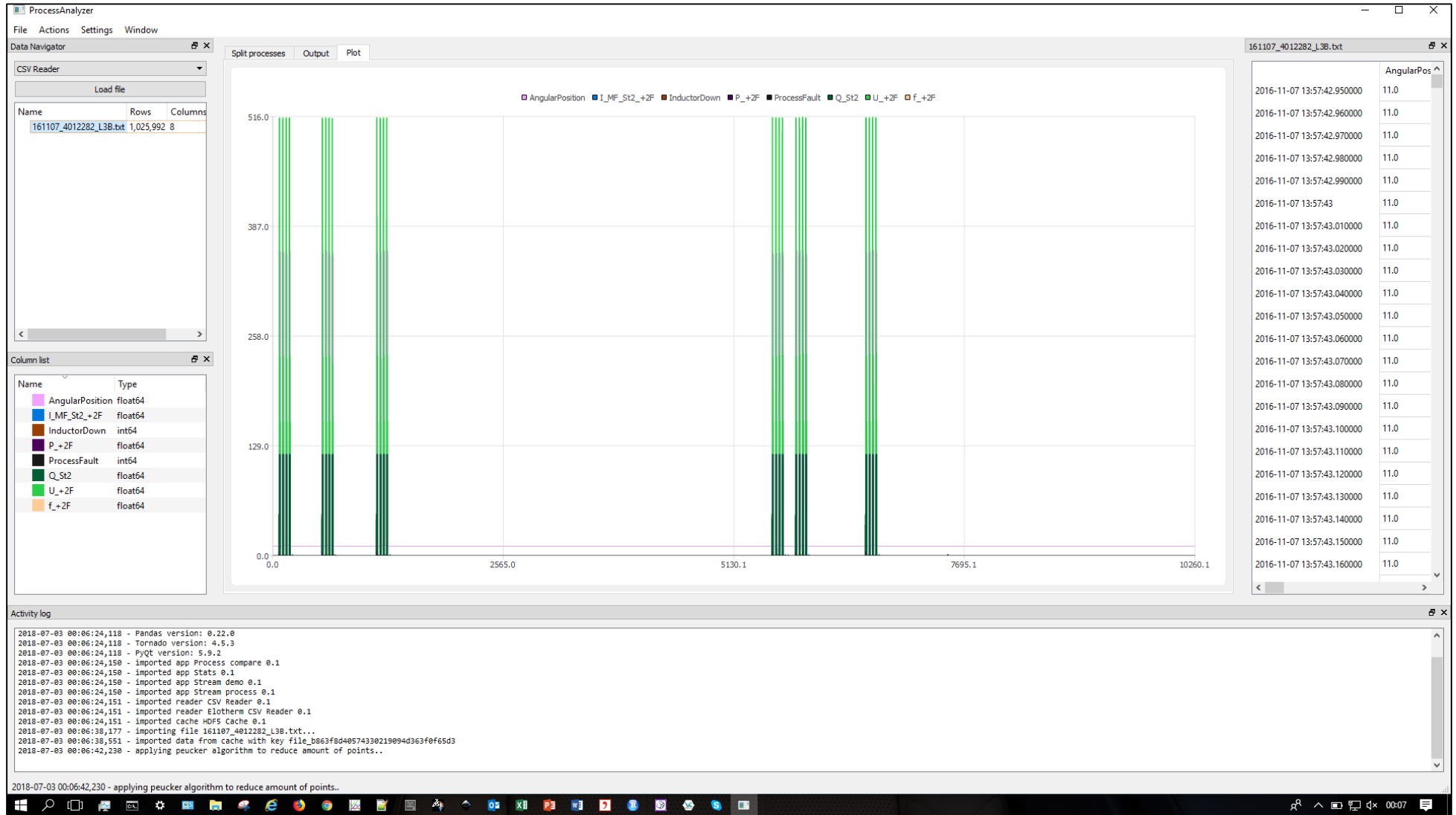
# Software package for condition monitoring

- Data import / export and data manipulation:
  - Handling of various data formats (Excel, .csv, hdf5, MySQL, logger-files)
- Data manipulation:
  - Handling of NA-Values, proper identification of data types, manual treatment of data (with given formulas), resampling, unit conversion
- Visualisation of data (test series):
  - Handling of big-data with resampling (e.g. based on Peucker-algorithm)
- Splitting of time series into separate processes based on given rules:
  - Out of signals, the processes are extrapolated for the further statistical comparison

# Software package for condition monitoring

- Determination of statistical values:
  - E.g. mean, median, quantiles, fast Fourier transform, rainflow counts
- Statistical tests:
  - Hypothesis test (e.g. Levene, F-/t-, Mann–Whitney U), trend analysis, curve fitting
- Detection of anomalies in the data sets (processes)
  - Based on machine learning algorithms (neuronal networks, inductive learning)
- Python based solution (various packages included: e.g. scipy, numpy, bokeh, pandas)

# Software package for condition monitoring



# Under development

The screenshot displays the ProcessAnalyzer application interface. The main window shows a decision tree for damage classification based on various engine and battery parameters. The tree starts with the root node 'Min engine rpm <= 799'. If 'Yes', it leads to 'Damage (482)'. If 'No', it leads to 'Median coolant temperature > 41'. This node branches into 'Yes' (leading to 'Max intake air manifold pressure <= 214') and 'No' (leading to 'Min engine rpm > 800'). The 'Max intake air manifold pressure <= 214' node branches into 'Yes' (leading to 'Mean engine rpm derived <= 3.88544') and 'No' (leading to 'Min engine rpm > 801'). The 'Mean engine rpm derived <= 3.88544' node branches into 'Yes' (leading to 'Min battery voltage > 12.2') and 'No' (leading to 'Min engine rpm > 809'). The 'Min battery voltage > 12.2' node branches into 'No' (leading to 'No damage (39189)') and 'Yes' (leading to 'Max acceleration > 7.460226'). The 'Max acceleration > 7.460226' node branches into 'No' (leading to 'No damage (714)') and 'Yes' (leading to 'Min engine rpm > 801'). The 'Min engine rpm > 809' node branches into 'Yes' (leading to 'Damage (3)') and 'No' (leading to 'No damage (21)'). The 'Min engine rpm > 801' node branches into 'Yes' (leading to 'Damage (4)') and 'No' (leading to 'No damage (22)'). The 'Min engine rpm > 801' node (under 'Max acceleration > 7.460226') branches into 'Yes' (leading to 'Damage (3)') and 'No' (leading to 'No damage (21)'). The 'Min engine rpm > 801' node (under 'Min battery voltage > 12.2') branches into 'Yes' (leading to 'Damage (5)') and 'No' (leading to 'No damage (21)'). The 'Min engine rpm > 800' node branches into 'Yes' (leading to 'Damage (5)') and 'No' (leading to 'No damage (22)').

The Data Navigator on the left shows a CSV file '161107\_4012282\_L38.txt' with 1,025,992 rows and 8 columns. The Column list shows variables: AngularPosition (float64), l\_MF\_St2\_+2F (float64), InductorDown (int64), P\_+2F (float64), ProcessFault (int64), Q\_St2 (float64), U\_+2F (float64), and f\_+2F (float64).

The Activity log at the bottom shows the following entries:

```
2018-07-03 00:06:124,118 - Pandas version: 0.22.0
2018-07-03 00:06:124,118 - Tornado version: 4.5.3
2018-07-03 00:06:124,118 - PyQt version: 5.9.2
2018-07-03 00:06:124,150 - imported app Process compare 0.1
2018-07-03 00:06:124,150 - imported app Stats 0.1
2018-07-03 00:06:124,150 - imported app Stream demo 0.1
2018-07-03 00:06:124,150 - imported app Stream process 0.1
2018-07-03 00:06:124,151 - imported reader CSV Reader 0.1
2018-07-03 00:06:124,151 - imported reader Elotherm CSV Reader 0.1
2018-07-03 00:06:124,151 - imported cache HDF5 Cache 0.1
2018-07-03 00:06:138,177 - importing file 161107_4012282_L38.txt...
2018-07-03 00:06:138,551 - imported data from cache with key file_b863f8d40574330219094d363f0f65d3
2018-07-03 00:06:142,230 - applying peucker algorithm to reduce amount of points...
2018-07-03 00:06:42,230 - applying peucker algorithm to reduce amount of points...
```



**Thank you for the attention!**

**Questions?**