# pycalibration Documentation

***Release 1.1.0***

**Jerome Douay**

**Aug 03, 2022**

# CONTENTS


**1 Contents** **3**
- 1.1 pycalibration . . . 3
- 1.2 License . . . 3
- 1.3 Contributors . . . 3
- 1.4 Changelog . . . 4
- 1.5 pycalibration . . . 6

**2 Indices and tables** **15**

**Python Module Index** **17**

**Index** **19**

This is the documentation of **pyCalibration**.

CHAPTER

ONE

# CONTENTS

## 1.1 pycalibration

python calibration support

Python library to supply basic common knowledge functionalities to support calibration

## 1.2 License



## 1.3 Contributors

- Jerome Douay <jerome@far-out.biz>

## 1.4 Changelog

### 1.4.1 Version 1.1.0

- MDF 3.0 time stamp correction
- small code error correction

### 1.4.2 Version 1.0.0

- large modification to code
- almost all classes with test procedure
- removes dcm as the code is broken, will be inserted again later on.

### 1.4.3 Version 0.8.0

- updates the vehiccle class and correct some of the code errors
- introduces the watchdog. This allow to have a monitoring system of a directory and detect new files
- extract introduces the interpolation of points from measurement files to correct the different time frame problems
- Extract no longer accept add mutliple channels to avoid issue with size and interpolation. Only single channel add.

### 1.4.4 Version 0.7.0

Inserted multiple new objects

- DCM
- Value
- Map
- Vehicle
- Gearbox
- Engine

See the documentation on features inserted to this release

### 1.4.5 Version 0.6.2

Function

- process class show a progress bar to indicate processing progress

Extract

- Extract inherit MDF class. MDF should no longer be used directly, always use Extract to get data
- removed add multiple chanels as the code was unstable
- add_channel now support a paraemter to interpolate or not the value ( not recommended for digitial signals )

MDF

- Still in the package and documentation.
- Allow interpolation of channels

### 1.4.6 Version 0.6.1

- corrects error in MDF get data, allowing multiple singals for one rename. The first defined singal is used, order matter !

### 1.4.7 Version 0.6.0

MDF

- data returned contain the measurement time in the time column. Indes is et to a datetime using the measurement time and the timestamp from the file

Function

- introduction of the function class
- base class for the development of functions in other packages
- process and lab method available

### 1.4.8 Version 0.5.0

Shift class inserted in the library. Documentation and test procedure available

### 1.4.9 Version 0.4.0

- Trigger class
- Trigger Documentation
- Trigger Test

### 1.4.10 Version 0.4.0

- MDF - Removed the set_rename and set_renames from the MDF to Extract - Documentation improved
- Extract - class introduced in this release - allow single file and multiple files - add signal ( single and multiple ) - set rename ( single and multiple ) - get data
- Documentation - documentation update and made easier to read

### 1.4.11 Version 0.3.0

- MDF - Add_signals to add singlas from a list - set_rename to allow renaming a channel after it has been inserted - set_renames allow to rename multiple channel using a list - disabled get_all as MDFReader can only process by channel group, work around to be done later on
- Documentation - documentation generation code completed - using the read the doc template - basic documentation, need more work
- Tests - MDF test procedure checking for most of the features. - 78% tested, need to improve to get 100 %

### 1.4.12 Version 0.1

- Initial commit, throw a lot a few code from my head into files.

## 1.5 pycalibration

### 1.5.1 pycalibration package

#### Submodules

#### pycalibration.curve module

**class** pycalibration.curve.**Curve**(*x*, *y*)

Bases: object

Curve allow the defintion and extroplation of data of a curve

This class is usually returned by the DCM class after importing a file.

**insert**(*x*, *y*)

Insert a point in the curve

**Parameters**

- **x** – x value of the point
- **y** – y value of the point

**Returns**
None

**y**(*x*)

return the y value of a curve given the x value.

Values are interpolated between the points given

### pycalibration.engine module

**class** pycalibration.engine.**Engine**(*n=[600, 1000, 2000, 4000, 6000]*, *t=[100, 500, 1000, 800, 100]*)

Bases: object

Class engine offers support for engine calculation and estimation.

**npower**(*power*)

**power**(*speed*, *limit=100*)

Return the Power at a given engine speed and a percentage torque :param speed: engine speed in rpm :param limit: max percentage torque :return: engie power in watts

**torque**(*speed*, *limit=100*)

Return the torque at an engine speed with a max percentage torque ( 100% is full torque) :param speed: engine speed in rpm :param limit: maximum percent torque from curve :return: engine torque in Nm

### pycalibration.extract module

**class** pycalibration.extract.**Extract**

Bases: *MDF*

Extract class extract channels from single or multiple files

**add_directory**(*pathname*)

Add a directory recursively to the files to be processed. Files recognize are mdf and mf4 exensions

**Parameters**
**path** – path to be added

**Returns**
none

**add_file**(*filename*)

Add single file to the list of files to be processed

**Parameters**
**file** – file name path to the file

**Returns**
none

**get**()

Read the MDF files and retrieved the requested data.

**Returns**
list of pandas dataframe contaiing the datas.

### pycalibration.function module

class pycalibration.function.**Function**

Bases: *Extract*

Base class to define function in module

Based on Extract, it supoprt all the extract method. It is necessary to set the files or directory to process before calling the process method.

Evaluate method should be overwritten for data evaluation related to the function beeing developed.

**evaluate**(*data*)

Using the data retrieved from the measurement file, generate calibration This method should be over writen by the derivative class and returns what ever the evaluation is producing.

**Returns**
should return the evaluation data

**lab**()

Write the labels and parameters in a lab file

**Returns**
None

**process**()

Retrieve the necessary information from the measurment files.

**Returns**
list containing files processed results

### pycalibration.gearbox module

class pycalibration.gearbox.**Gearbox**(*n=[1, 2, 3, 4, 5]*, *r=[10, 9, 8, 7, 6]*)

Bases: object

Gearbox class Class provide interface to get the gear ratio on a given gear

**ratio**(*gear*)

Return the gear ratio for a given gear :param gear: gear number :return: gear ratio

### pycalibration.map module

class pycalibration.map.**Map**(*x*, *y*, *z*)

Bases: object

**x**(*y*, *z*)

**y**(*x*, *z*)

**z**(*x*, *y*)

### pycalibration.mdf module

class pycalibration.mdf.**MDF**(*filename=None*)

Bases: object

MDF class to handle MDF read operation

**add_channel**(*channel*, *rename=''*, *inter=False*)

Set a channel to be retrieved from the MDF. If a rename name is supplied, the channels will be reneamed. If more than one channel as the same rename name, all channels will be checked until one available is found. Interpolation should not be used on digitial signal. The interpolation is linear and should be used on non digitial signals to improve accuracy lf signal in measurement with multiple time raster.

**Parameters**

- **channel** – channel name
- **rename** – name to be renamed to
- **inter** – Set to True to interpolate missing values, default False.

**Returns**
None

**get_channel**(*channel*)

Get the data designated by the channel name

**Parameters**
**channel** – channel name

**Returns**
pandas dataframe containing the data

**get_data**()

Read the MDF file and retrieved the requested data

**Parameters**
**filename** – filename ( with full path ) of the MDF file to open

**Returns**
pandas dataframe containing the datas. The time offset for the channels is set to the column offset. The dataframe indes is based on the file timestamp with the measurement time offset. This allows datetime operation on the dataframe.

**set_file**(*filename*)

### pycalibration.resistance module

class pycalibration.resistance.**Resistance**(*cx=0*, *fr=0*, *ro=0*)

Bases: object

This class calculates the vehicle driving resistance

**fair**(*v*)

Return the air resisting force :param v: vehicle speed in m/s :return: air resisting force in N

**fpitch**(*m*, *grade*)

Return the pitch resisting force at a specified pitch :param m: total vehicle mass in kg :param pitch: slope in percentage :return: pitch resisting force in N

**frolling**(*m*, *grade*)

Return the rolling resisting force :param m: total vehicle mass in kg :param pitch: slope in percentage :return: rolling resisting force in N

**resistance**(*v*, *m*, *grade=0*)

Return the sum of air, pitch and rolling resisting force :param v: vehicle speed in m/s :param m: total vehicle mass in kg :param grade: slope in percentage :return: total resisting force in N

### pycalibration.shift module

**class** pycalibration.shift.**Shift**

Bases: object

Generate Table bevore and after shifting All the columns supplied in the data for process will be used. At the beginning of the shift, the columns will be added '_pre', at the end of the shift '_post'.

**process**(*data*)

Process the data and return the table containing the shifts.

**Parameters**
**data** – pandas Dataframe containing the data, inclusive the channel to be used to detect the shift

**Returns**
Pandas dataframe containing the pre and post shifts data.

**set_post**(*post*, *up=False*)

Set the post channel used for trigger

**Parameters**

- **pre** – Channel name to be used as trigger
- **up** – Determine the signal direction ( up/down, up by default )

**set_pre**(*pre*, *up=True*)

Set the pre channel used for trigger

**Parameters**

- **pre** – Channel name to be used as trigger
- **up** – Determine the signal direction ( up/down, up by default )

### pycalibration.trigger module

**class** pycalibration.trigger.**Trigger**

Bases: object

Generate Table at the time of the event. The event shall be generated from a digital signal

**process**(*data*)

Process the data and returns the events in a pandas dataframe

**Parameters**
**data** – data to be analysed

**Returns**
pandas dataframe with all the signals and events

**set_trigger**(*name*, *up=True*)

Set the name of the trigger signal

**Parameters**

- **name** – name of the signal to be used as trigger ( digital )
- **up** – True if the event is rising

**Returns**
None

### pycalibration.value module

**class** pycalibration.value.**Value**(*x*, *y=None*, *z=None*)

Bases: object

Value class represent the informations stored in a parameter. Each value can be a single value, a curve or a map. All points are interpolated

**x**(*pos*)

Return the X value at a position ( interpolated from the array) :param pos: position in the array :return: X value

**y**(*x*)

Return the Y value at a X position ( interpolated from the array) :param pos: position in the array :return: Y value

**z**(*x*, *y*)

Return the Z value at a X,Y position ( interpolated from the array) :param pos: position in the array :return: Z value

### pycalibration.vbo module

**class** pycalibration.vbo.**VBO**

Bases: object

**read**(*filename*)

### pycalibration.vehicle module

**class** pycalibration.vehicle.**Vehicle**

Bases: object

class to represent a vehicle and basic calulation for the vehicle configuration

**energy**(*n*, *gear*, *mass*)

**engine_to_speed**(*engine*, *gear*)

Returns the vehicle speed from the engine speed and the common powertrain ratio :param engine: engine speed in 1/min :param gear: gear number :return: vehicle speed in m/s

**force_to_torque**(*f*, *gear*)

Returns the engine torque from the traction force and the common powertrain ratio :param f: traction force in N :param ratio: common powertrain ratio :return: engine torque in Nm

**fresist**(*speed*, *mass*, *grade=0*)

Returns the resistance force by entering the vehicle speed and the ratio (default ratio = 1) :param speed: vehicle speed in m/s :param mass: total vehicle mass in kg :param grade: road grade :return: resistance force in N

**ratio**(*gear*)

return the gear ration from the given gear

**Parameters**
**gear** – gear number

**Returns**
trans ratio (gear and rear axle ratio)

**speed_to_engine**(*speed*, *gear*)

Returns the engine speed from the vehicle speed and the common powertrain ratio :param speed: vehicle speed in m/s :param gear: gear number :return: engine speed in 1/min

**torque**(*n*, *pedal=100*)

By default, this function returns the torque at an engine speed with a max percentage torque (100% is full torque) :param n: engine speed in rpm :param pedal: percentage value of the accelerator pedal; default 100% means full torque :return: engine torque in Nm

**torque_to_force**(*t*, *gear*)

Returns the traction force from the engine torque and the common powertrain ratio :param t: engine torque in Nm :param ratio: common powertrain ratio :return: traction force in N

### pycalibration.watchdog module

**class** pycalibration.watchdog.**Watchdog**(*queue*, *path='.'*)

Bases: FileSystemEventHandler

**on_created**(*event*)

Called when a file or directory is created.

**Parameters**
**event** (DirCreatedEvent or FileCreatedEvent) – Event representing file/directory creation.

**on_moved**(*event*)

Called when a file or a directory is moved or renamed.

**Parameters**
**event** (DirMovedEvent or FileMovedEvent) – Event representing file/directory movement.

**process**(*path*)

**start**()

**stop**()

### Module contents

CHAPTER

TWO

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## p

pycalibration, 13
pycalibration.curve, 6
pycalibration.engine, 7
pycalibration.extract, 7
pycalibration.function, 8
pycalibration.gearbox, 8
pycalibration.map, 8
pycalibration.mdf, 9
pycalibration.resistance, 9
pycalibration.shift, 10
pycalibration.trigger, 10
pycalibration.value, 11
pycalibration.vbo, 11
pycalibration.vehicle, 11
pycalibration.watchdog, 12

# INDEX


## A

add_channel() (*pycalibration.mdf.MDF method*), 9
add_directory() (*pycalibration.extract.Extract method*), 7
add_file() (*pycalibration.extract.Extract method*), 7

## C

Curve (*class in pycalibration.curve*), 6

## E

energy() (*pycalibration.vehicle.Vehicle method*), 11
Engine (*class in pycalibration.engine*), 7
engine_to_speed() (*pycalibration.vehicle.Vehicle method*), 11
evaluate() (*pycalibration.function.Function method*), 8
Extract (*class in pycalibration.extract*), 7

## F

fair() (*pycalibration.resistance.Resistance method*), 9
force_to_torque() (*pycalibration.vehicle.Vehicle method*), 11
fpitch() (*pycalibration.resistance.Resistance method*), 9
fresist() (*pycalibration.vehicle.Vehicle method*), 11
frolling() (*pycalibration.resistance.Resistance method*), 9
Function (*class in pycalibration.function*), 8

## G

Gearbox (*class in pycalibration.gearbox*), 8
get() (*pycalibration.extract.Extract method*), 7
get_channel() (*pycalibration.mdf.MDF method*), 9
get_data() (*pycalibration.mdf.MDF method*), 9

## I

insert() (*pycalibration.curve.Curve method*), 6

## L

lab() (*pycalibration.function.Function method*), 8

## M

Map (*class in pycalibration.map*), 8
MDF (*class in pycalibration.mdf*), 9
module
    pycalibration, 13
    pycalibration.curve, 6
    pycalibration.engine, 7
    pycalibration.extract, 7
    pycalibration.function, 8
    pycalibration.gearbox, 8
    pycalibration.map, 8
    pycalibration.mdf, 9
    pycalibration.resistance, 9
    pycalibration.shift, 10
    pycalibration.trigger, 10
    pycalibration.value, 11
    pycalibration.vbo, 11
    pycalibration.vehicle, 11
    pycalibration.watchdog, 12

## N

npower() (*pycalibration.engine.Engine method*), 7

## O

on_created() (*pycalibration.watchdog.Watchdog method*), 12
on_moved() (*pycalibration.watchdog.Watchdog method*), 12

## P

power() (*pycalibration.engine.Engine method*), 7
process() (*pycalibration.function.Function method*), 8
process() (*pycalibration.shift.Shift method*), 10
process() (*pycalibration.trigger.Trigger method*), 10
process() (*pycalibration.watchdog.Watchdog method*), 12
pycalibration
    module, 13
pycalibration.curve
    module, 6
pycalibration.engine
    module, 7
pycalibration.extract
    module, 7

pycalibration.function
    module, 8
pycalibration.gearbox
    module, 8
pycalibration.map
    module, 8
pycalibration.mdf
    module, 9
pycalibration.resistance
    module, 9
pycalibration.shift
    module, 10
pycalibration.trigger
    module, 10
pycalibration.value
    module, 11
pycalibration.vbo
    module, 11
pycalibration.vehicle
    module, 11
pycalibration.watchdog
    module, 12

## R

ratio() (*pycalibration.gearbox.Gearbox method*), 8
ratio() (*pycalibration.vehicle.Vehicle method*), 12
read() (*pycalibration.vbo.VBO method*), 11
Resistance (*class in pycalibration.resistance*), 9
resistance() (*pycalibration.resistance.Resistance method*), 10

## S

set_file() (*pycalibration.mdf.MDF method*), 9
set_post() (*pycalibration.shift.Shift method*), 10
set_pre() (*pycalibration.shift.Shift method*), 10
set_trigger() (*pycalibration.trigger.Trigger method*), 10
Shift (*class in pycalibration.shift*), 10
speed_to_engine() (*pycalibration.vehicle.Vehicle method*), 12
start() (*pycalibration.watchdog.Watchdog method*), 12
stop() (*pycalibration.watchdog.Watchdog method*), 12

## T

torque() (*pycalibration.engine.Engine method*), 7
torque() (*pycalibration.vehicle.Vehicle method*), 12
torque_to_force() (*pycalibration.vehicle.Vehicle method*), 12
Trigger (*class in pycalibration.trigger*), 10

## V

Value (*class in pycalibration.value*), 11
VBO (*class in pycalibration.vbo*), 11
Vehicle (*class in pycalibration.vehicle*), 11

## W

Watchdog (*class in pycalibration.watchdog*), 12

## X

x() (*pycalibration.map.Map method*), 8
x() (*pycalibration.value.Value method*), 11

## Y

y() (*pycalibration.curve.Curve method*), 6
y() (*pycalibration.map.Map method*), 8
y() (*pycalibration.value.Value method*), 11

## Z

z() (*pycalibration.map.Map method*), 8
z() (*pycalibration.value.Value method*), 11