CWI User Manual

Reuben Zotz-wilson

May 10, 2018

Date updated:	22 July, 2016
Partners:	Thijs Boerrigter

1 Introduction

This manual describes the use of a package of functions which allow the processing of time series data for the purpose of coda wave analysis. The initial intention was to expedite the analysis of time lapse, ultrasonic data acquired during axial loading experiments. The processing is now generalised to allow in input of two basic data streams, time-series (TS) data and any number of complementary perturbation vectors.

2 Data Input

Two principal data types are taken as input, the time series (e.g. Ultrasonic time lapse traces) and the corresponding perturbation dataset (e.g. axial loading information). The following will detail the accepted input formats for these two principal data streams. The raw input data will be loaded into memory and then saved to disk in a HDF5 database.
Note: I need to generalise the input data further, and instead of searching for the format, simply ask the user to specify the input data format types.

2.1 Binary TS Data and CSV PV Data

The user must provide a folder containing binary file TS data only, along with a single csv file all PV data formats. (This import function needs to be generalised)

2.2 Database loading

After the first run of a particular dataset a folder `"TS_loc"_DB` will be created in the run folder. A HDF5 database `TS_rawh5` will be created containing a single table titled `TS_full`. Due to the shape of such time series information, typically square instead of tall this database is saved in "fixed" format and therefore can not be queried. Another database `DB_tblh5` will be also created at this point in the same folder, containing the remaining imported data, this dataset can be queried.
.

`TS_rawh5`	hdf5 database containing all TS data within the input folder
`TS_full`	Tabel name stored within `TS_rawh5`
`DB_tblh5`	hdf5 database containing all input PV and TShdrs data found

If the user input parameter `loadDB` is set to `True` and existing `DB_tblh5` and `DB_tblh5` databases are found within the folder `TS_loc_DB`, then no reload of the raw input data will be performed. Instead, the hdf5 databases will be loaded, after which all normal processing is possible. Note, it is preferable to avoid re-loading any initial data formats as this step is time consuming, particularly for large numbers of CSV files.

After the processing is finished, `run_datastore()` will be called. This function within the class `dataStore` will create the following databases within the folder `"TS_loc"_DB` .

`DB_tbl_processedh5`	hdf5 database containing the processed PV and CC and TShdrs
`TS_cuth5`	hdf5 database containing the TS data equal to the number of CC rows. The table name `TS_df` can be used to access the this table. Note: this databased is saved in "fixed" format and therefore cannot be queried

2.3 section

2.4 Time Series

The accepted formats for the time series (TS) data are ".CSV" and binary file formats. In both cases a single file is expected to contain a vector of amplitude with a constant sampling frequency. If multiple receivers are available for each measurement step stored within the same file, the user must select which receiver is to be analysed. File numbering should increase with the time of measurement, where the date and time of the file creation is used to match the TS data to its corresponding PV (perturbation vector - time column).

2.5 Perturbation Vectors

The perturbation vectors are input from a ".CSV" file, where the display of both a primary and secondary vector is possible. The primary data is intended to be plotted against the CWI CC's (Cross-correlation Coefficients) when the perturbation is in a continuous direction. In the case of a cyclic perturbation measurement numbers or measurement time should be input as the primary perturbation vector. The secondary perturbation vector is intended to be used for line of surface colouring, in order to display this data on the same plot as the CC's.

2.6 Paramater Setup File

Parameter input and overall control of the signal processing is handled by an input text file. There are two types of inputs, those which are mandatory such as the relative folder location for the TS data, and those which have a default value if not defined, such as the sample or run name. An example of the input syntax is given below:

```
% input file for python based CWI processing
FB_cut = 1 % sample points to remove from all TS traces
END_cut = 5000 % sample point after which all will be removed from all TS traces
STA_meas = 1 % start of TS measurements included in processing
END_meas = 10 % last measurement to be included in processing


% Cross-correlation parameters
ww = 400 % the window with in number of samples
```

```
ww_ol = 10 % the window with [%] overlap
CC_ref = 1 % If one is selected
```

2.7 User input variable description List

`import_dtype`	The date type to import, Either 'bin_par', 'Shell_format' or 'NoT-Shdrer_format'. Default is 'bin_par'
`TSloc`	location of time series data containing folder or single file
`PVloc`	location of pert. data containing folder or single file
`loadDB`	True or False, any existing raw database file will be loaded in place of the raw input data formats.
`acqDate`	The date of acquisition, is required for some data types which are not time matched
`recNo`	The receiver number which will be selected for processing, if not given no TS splitting will be attempted
`sampNo`	The number of sample points in a single trace recording, only required as input if the rec
`SampFreq`	The number of samples recorded per second

Pree-processing inputs

`stress_strain`	If "True", an attempt will be made to calculate stress and strain from inputs L and D assuming a cylinder
`L`	The length of sample in mm
`D`	The diameter of the sample in mm
`Ax_Press_corr_ADT`	[micron/bar] correction for depletion strain calculation
`Rad_press_corr_ADT`	[micron/bar] correction for depletion strain calculation
`avg_Press_corr_RDT`	[micron/bar] correction for depletion strain calculation
`Temp_corr`	[micron/DegC] correction for depletion strain calculation
`Pax_friction`	correction for depletion strain calculation
`LVDT_zidx`	Index point in perturbation data frame where LDVT will be zeroed
`LLLength`	Length in sample points of the Linear Line, if given the search for the most linear line will run. Note: 'Stress [MPa]' and 'Strain Ax. [%]' must be in PV_df
`FB_cut`	Number of sample points to remove from the start of each TS
`END_cut`	Sample point after which all following will be removed
`STA_meas`	The first measurement to be included in the processing
`END_meas`	The last measurement to be included in the processing
`PV_STA`	The first PV measurement to include in display of information
`PV_END`	The last PV measurement to include in display of information
`TS_samp_dt`	The TS sampling dt in seconds. If given then time matching will be based on both the dt and the first .par file count vector.
`match_timeshift`	Time shift to the TS file acquisition time in seconds. Only used in conjunction with `TS_samp_dt`.
`TS_file_tmatch`	Match TS data based on time file last modified.

Cross-Correlation parameters

ww	The correlation window width in sample points. If wdwPos is given then ww_ol is irrelevant and ww must have the same number of entries as wdwPos
ww_ol	The percentage overlap of windows
wdwPos	The select positions of correlation windows in sample points
CC_type	Either 'fixed' or 'rolling' to define the CC type input
CC_ref	1,2,3...etc TS as fixed or rolling reference
Eng_ref	If True, and CC_type='rolling', energy comparison will be made with the first TS acquired. If False, comparisons will be made with same TS as CC.
taper	Either 'True' or 'False' to filter with a hann window each CC window. Note the default is set to False if taper not given in the input file.

Post Processing Paramerters

FBP	Request First Break Picking to be performed of each processed trace
ww_ol	The percentage overlap of windows
CC_type	Either 'fixed' or 'rolling' to define the CC type input

Display parameters

disp_DB	Toggle the request to plot processed data, if 'True' plot will be made, otherwise not.
PV1	Name of the perturbation information as in input file, x-axis
PV2	Name of the perturbation information as in input file, y-axis
PV1_lb	label for PV1, if not provided the column name is assigned
PV2_lb	label for PV1, if not provided the column name is assigned
TS_units	User defined TS display units, 'sec' or 'msec'

2.8 Module list

userInput	For all user defined parameters and additional setup data, read from a text file where comments after # are ignored.
data_import	Functions for the import of a variety of data types and formats, requires location of TS and PV data as input
pre_process	Various pre-processing operations performed on both TS and PV input data
cross_correlation	Grouping of functions which handle multi window correlation time series data, with focus on CWI and associated methods
postProcessing	Various post-processing operations performed on both TS and PV input data
dataStore	Databasing and loading of the TS and PV data, along with txt binary storage of parameters.
utilities	A collection of functions intended to perform useful operations for the expected user base.
dispCWI	Display of CWI derived data against TS and PV input data
runCWI	For the overarching control of steps userInput, data_import, pre_process, cross_correlation and dispCWI

3 Module description

3.1 Module: `dataStore`

3.1.1 Class: utilities

A collection of functions intended to perform useful operations for the expected user base. For example, if the user wishes to export all the standard process generated databases then the function `hdf_csv_dump` can be called.

`hdf_to_csv(DB_fdl):`
Simply provide a relative of absolute path to the folder containing the program generated hdf5 databases in the formate `'database_folder/'`. The following csv files will be saved within the given database folder:

- `PV_df.csv`: All the supplied perturbation data in a single csv file equal in dimensions to the number to TS measurements processed.
- `CC.csv`: All generated CC coefficients for all employed window positions within the coda.
- `TShdrs.csv`: All TS header information input included in a single CSV table.

An example of using this function would be while in the run folder type the following into the terminal.

```
Import dataStore as ds
ds.utilities.hdf_csv_dump('AV_run_b_DB/')
```

Please view the source code for further explanation.

`run_dataLoad(DB_fdl):`
Loads a previous processing session into memory ready for analysis. `Inputs`

- `DB_fdl`: input folder holding the expected databases in the form `'DB_fld/'`
- `PV_df`: Main database holding PV, and CC data
- `TS_DB`: Database of TS data
- `PV_df_full`: database including all PV data, empty if original PV and TS data was coincident already

An example of using this function would be while in the run folder type the following into the terminal.

```
Import dataStore as ds
ds.utilities.run_dataLoad('TS_Data_DB/')
```

3.2 Module: disCWI

A collection of classes intended to the display of CWI type data.

3.2.1 Class: `dispUtilities`:

A collection of functions for various specific display requirements
multi_wdw(param, DB, TS, lag, PV, multi_plot = None, PV1 = None, PV2 = None):
Intended to display multiple correlation windows on a single 2D plot, with the average of all CC trend lines bold. Comparison with two PV is possible. The parameter multi_plot can be used to select either multiple 'CC', 'CC_lag', or 'RelAmp', default set to 'CC' if none are given.
—Inputs—
param: output param file summarising the call, the window width, PV1, PV2 are read from this file
DB: DataFrame containing all PV and CC data
TS: The time series, pandas series trace indexed in time [sec]
lag: The value of lag to compare to
PV: DataFrame of PV only data. If provided then PV1 and PV2 will be taken from here, not DB.
multi_plot: Either 'CC', 'CC_lag' or 'RelAmp', default 'CC'.
PV1 and PV2: name of x y2 columns to plot against, if given

```
import dispCWI as disp
hdl = disp.dispUtilities.multi_wdw(param, PV_df, TS_DB[3], 3, PV_df_full)
```

The output image from the function displays multiple overlapping windows along with the average of all for a single lag value.

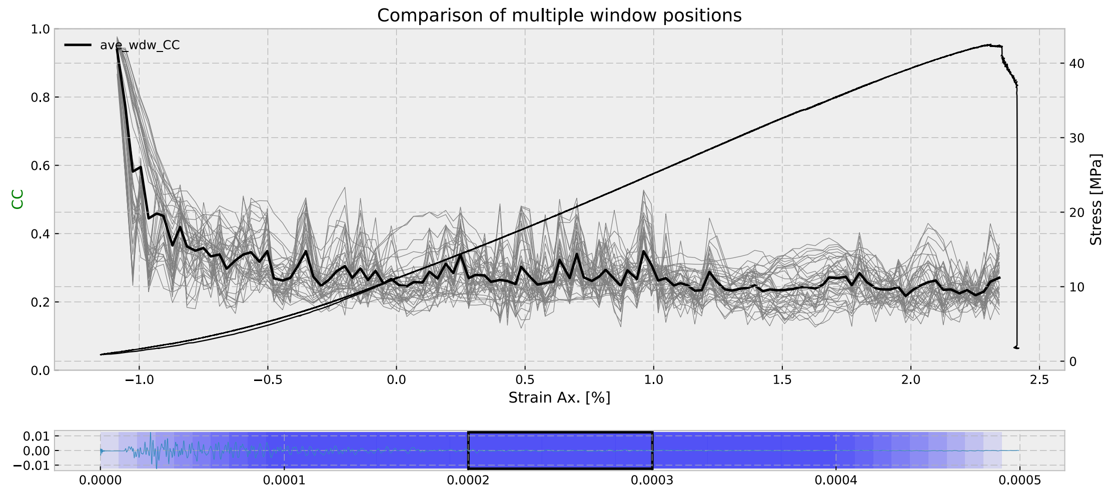

Figure 1: The output image from the function displays multiple overlapping windows along with the average of all for a single lag value.

dispCWI_pres(param, DB, TS, ccPlot_list, PV_full=None, PV1=None, PV2=None, PV3=None, PV4 = None, plot_dic={}):
The flexible display of multiple input vectors
—Inputs— param: file summarising the processing run
DB: Database of all PV and CC data to be plotted

TS: Single TS in order to display the window positions
PV_full Non filtered PV_data, used for plot if given
ccPlot_list List of tuples in CC data to plot
PV1=None X-axis
PV2=None Y2 right axis
PV3=None Y3 right axis
plot_dic Dictionary to flexibility in ploting

```
plot_dic = {'PV3_lb': 'Stress[MPa]', 'PV2_lb': 'Strain', 'CC_range': [0.95, 1],
            'X_axis_range': None, 'PV2_range': None,
            'PV3_range': None, 'unit_m_4': 1000, 'elapsed_hours': True}
```

```
import dispCWI as disp
hdl = disp.dispUtilities.dispCWI_pres(param, PV_df, TScut_DB[3], ccPlot_list,
                                       None, PV1, PV2, PV3, PV4, plot_dic)
```

The output image from the function displays multiple overlapping windows along with the average of all for a single lag value.

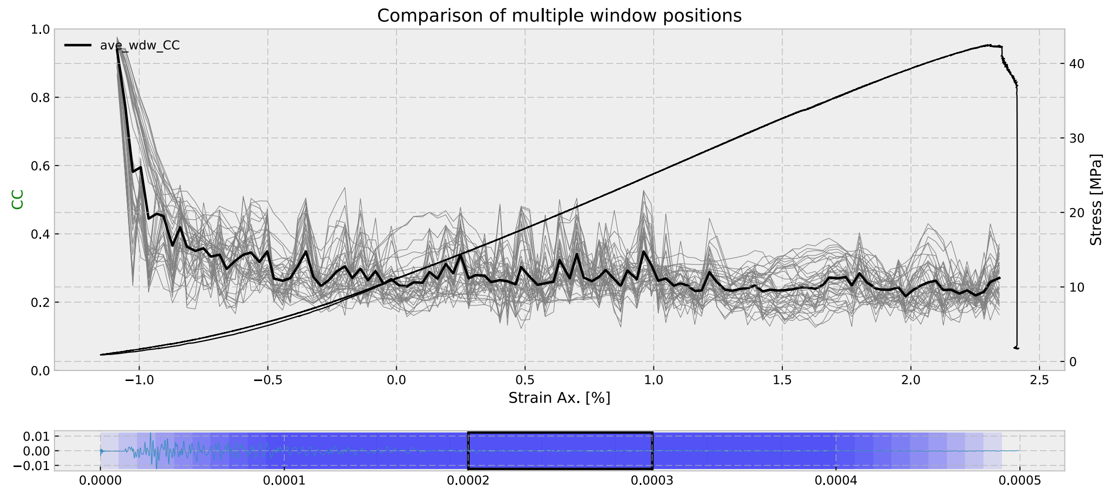

Figure 2: The output image from the function displays multiple overlapping windows along with the average of all for a single lag value.

3.3 Module: pre_process

A collection of pre-processing utility functions.

3.3.1 Class: `utilities`:

detect_most_linear(x, y, m):
Finds the most linear portion of a line via the use of a covariance matrix
Inputs:
x: x-axis array of values
y: y-axis array of values
m: Length of most linear portion of line (x, y)

Outputs:
L_end: `x_end, y_end` values at the end of the line
idx: Indices of the most linear portion of the line

3.4 Module: postProcess

A collection of functions intended for post processing of CWI type data structures. For example, the calculation of Time Of Flight (TOF) velocity analysis can be added to a database through the use of the function `TS_FBP`.

3.4.1 Class: `postProcess`:

Calls a sequence of functions which define the standard post processing.

postProcess_run(self): Perform expected Post Processing. This function will perform First Break Picking, if the parameter FBP is given and set to True in the user input file.

TS_DB(self): Convert TS matrix into a DataFrame with axis index in time.

_DatetimeIndex(self): Check if the word Date is in the PV1 user defined parameter provided in the user input file. If so perform processing to provide pd.DatetimeIndex as the axis.

_stress_strainP(self): Perform stress strain calculation based on user defined inputs. Checks to see if input parameter *stress_strain* is given and is 'True' (string not boolian). Note this function assumes that the data frame contains the following columns:

- `LVDT1 and LVDT2`
- `Force(kN)`

PV_CC_join(self): Combine both PV and CC data into a single dataset based on.
measurement number

3.4.2 Class: `post_utilities`:

`TS_FBP`: This function takes as input a pandas database of time series increasing columnweise, and performs analysis of the first break of energy on each trace. A window width of noise is defined, which is used to determine the expected standard deviation and mean of the noise for each trace. The function returns there outputs:

- `idx_break`: The index of the first break detection point for each TS

In order to check the first break picking, one can run the following.

```
import postProcess as pp
wdw_noise = 0.015 # in seconds
threshold = 0.5
verbose = True # request an interactive plot of the TS
mpd = 200
thdFactor = 10 # Factor added to stdThd*noiseStd for low noise environments
pp.post_utilities.TS_FBP(TS_DB, wdw_noise, threshold, thdFactor, mpd, verbose)
```

For further diagnostics one can run a trace by trace detection of peaks as follows:

```
import postProcess as pp
pp.post_utilities.detect_peaks(TScut_DB[7], mph=0.05, show=True)
```

References