Information on the beam chopper on the Wendelstein 7-X alkali beam diagnostic

S. Zoletnik

21 September, 2024

Chopper configuration

The beam modulation system of the alkali beam diagnostic (ABES) has two deflection plate pairs. One deflects the beam tororidall, one poloidally. The toroidal deflection is used for chopping (removing the beam from the observation volume), while the poloidal for deflection. For each plate pair two voltage states are defined and the control system switches between these two plates periodically during a discharge. The beam modulation status is described by the “chopper” and “deflection” states, each can be 0 or 1. The timing of these periods is defined in a chopper file whose contents are copied into the “Chopper” section of the shot XML configuration file. At present the deflection is not used as the beam is widened during deflection.

Two basic modes of the beam modulation operation are used:

- **Camera mode** means the chopper and deflection states change synchronized to the CMOS camera frames. The CXRS spectrometer camera also exposes synchronized to the CMOS camera. The timing description tells the system what state (chopper, deflection) should be set for consecutive frames. In most cases the chopper state is periodically changing between 0 and 1 and the deflection is always 0.
- **Timed mode** means the beam modulation is not synchronized to the CMOS camera but phases with different states and length are defined. This is typically used for fast chopping in the 10-100 kHz range.

Timing of the beam modulation is always based on the internal clock of APDCAM. This means, that when external clock is used in the APDCAM there may be a time drift between the chopper and the beam signals. The shift is maximum 100 microseconds per second. Therefore, for fast chopper operation internal clock must be used.

Using the chopper information in FLAP

The flap_w7x_abes module offers the possibility to read the chopper configuration and process the beam signals by taking into account the chopper timing. The following code reads the time periods when the beam is on into a FLAP data object.

```
d_beam_on=flap.get_data('W7X_ABES',
                             exp_id=exp_id,
                             name='Chopper_time',
                             coordinates = {'Time': timerange},
                             options={'State':{'Chop': 0, 'Defl': 0},
                                      'Start':1000,
                                      ’End’:-1000
                                      }
                            )
```

“Chopper time” is the requested signal name, the “coordinates” argument defines the time range for which the data is needed. The “State“ keyword in the “options” argument defines which chopper state is requested. In the example the chopper and deflection is off, that is the beam is on. The “Start” and “End” keywords enable to shift the beginning and end of the chopper time intervals relative to the chopper switching time. The value is in microseconds. When APDCAM was run on external clock 1000 microsecond is a typical time to use to compensate for the time shift. For fast chopping measurements 0 and -3 is usually a good value.

The data object returned by the above call contains no data but only “Time” and “Sample” coordinates which can be used for selecting time intervals for processing. In the spectral analsys, filtering and similar procedures a time interval can be set. It can be specified as

```
Intervals={‘Time’:d_beam_on}
```

or

```
Intervals={‘Sample’:d_beam_on}
```

The first example defines the time intervals in Time coordinates the second in Samples. The two are equivalent.

The timing of the chopper relative to the beam signal can be checked using the test_chopper_timing program in the flap_w7x_abes module. Example plots for a camera chopping and a fast chopping case are shown in Figure 1.

```
def test_chopper_timing(exp_id=None, timerange=None,signal='ABES-15',
                        resample=1e3,x_axis='Time',start_shift=0,end_shift=0):
    """
    Test the chopper timing visually. Plots a signal and the chopper on and off
    periods.
    Also calculates the mean signal in on/off intervals and plots them.

    Parameters
    ----------
    exp_id : string
        The experiment ID.
    timerange : 2-element list, optional
        The time range. The default is None, which means all data.
    signal : string, optional
        The signal name. The default is 'ABES-15'.
    resample : float, optional
        The resampling during data read. The default is 1e3. If None no resampling
        is done.
    x_axis : string, optional
        'Time' or 'Sample'. The default is 'Time'.
    start_shift : float
        The shift of the chopper start points [microsec]
    end_shift : float
        The shift of the chopper end points [microsec]

    Returns
    -------
    None.

    """
```

The chopper description can also be used for slicing the signal(s) according to the chopper with the “slice_data” FLAP moethod:

```
d.slice_data(slicing={‘Time’:d_beam_on})
```

This procedure adds a new dimension to the data which is looping through the samples in the chopper periods. Two “Time” and “Sample” are cut into two coordinates each, one along the samples in a chopper interval and one through the chopper intervals. The mean of the data in the chopper intervals can also be calculated using the “summing” argment:

```
d_on = d.slice_data(slicing={'Time':d_beam_on},
                    summing={'Rel. Sample in int(Time)':'Mean'},
                    options={'Regenerate':True}
                    )
```

In the above example mean calculation is done through the samples in the interval, that is through the coordinates “`Rel. Sample in int(Time)`”. This is one of the new coordinates generated by slicing. The “Regenate” option tells the program to reconstruct the Time coordinate from the two new coordinates from the slicing. This way the mean signal in the chopper intervals can be calculated as a function of time. The example below (from tests/chopper_test_plot.py in the flap_w7x_abes module) shows this procedure, and the resulting plots.

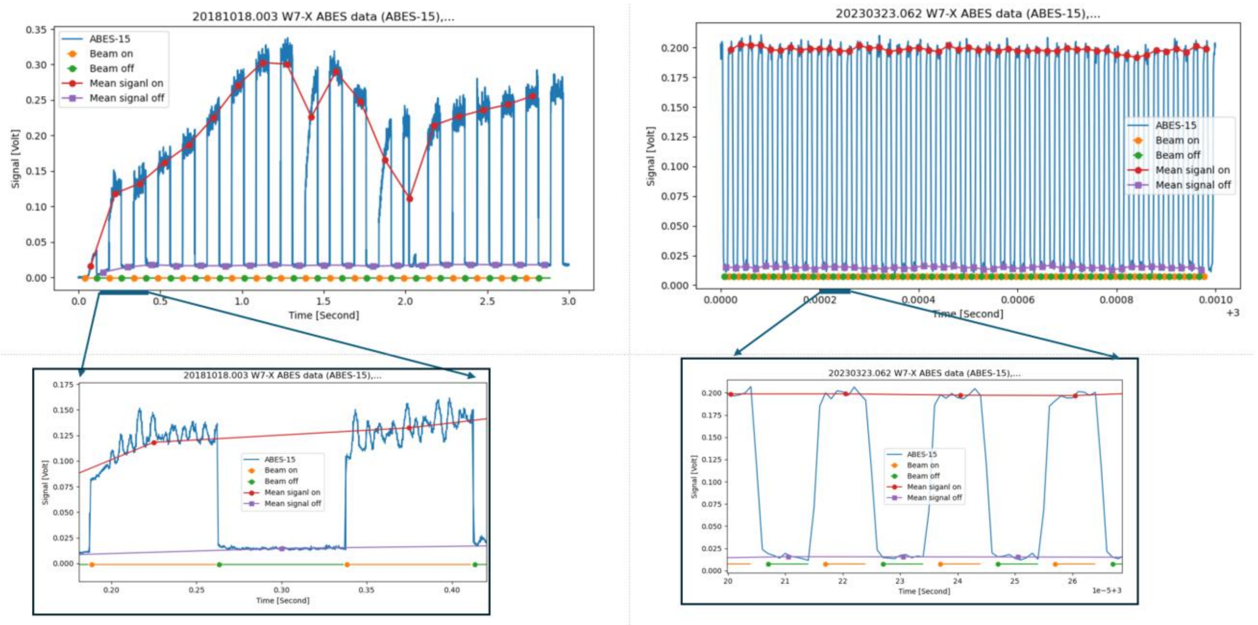

Figure 1. Example camera chopping (left) and fast chopping signal and processing using the chopper data object.

If backgorund correction is desired an interpolation is needed to bring the beam-on and beam-off signals to the same timescale. This can be done with the slice_data method:

```
        d_off = d_off.slice_data(slicing={'Time':d_on},
                                 options={'Interpolation':'Linear'}
                                 )
```

This call instructs FLAP to do interpolation of the data in d_off to the time coordinates of d_on.