CESR BPM
Version 1.0
May 11, 2021

Toy Monte Carlo simulation of the CBPM system

A. Chapelain[1]

[1]Cornell Laboratory for Accelerator-based Sciences and Education

Contents

1 Introduction **2**

2 Toy Monte Carlo ingredients and configuration file **2**

2.1 Beam position 2

2.2 Button analog waveform 3

2.3 Timing offset 4

2.4 Timing jitter 4

2.5 Electronic noise 5

2.6 Beam position modulation and drift 5

3 How to run the code **6**

4 Button amplitude: sources of error **6**

4.1 Ideal button readings 7

4.2 Adding timing offset 7

4.3 Adding timing jitter 7

4.4 Adding electronic noise 7

5 Button amplitude: beam position drift and modulation **9**

5.1 Beam position drift 9

5.2 Beam position modulation 9

A Configuration file **10**

1 Introduction

This note presents the toy Monte Carlo simulation written in `Python3` of the `CBPM` system: its ingredients, how to use it and some examples. The toy Monte Carlo aims at reproducing and understanding the performance of the `CBPM` system, namely the single-shot (turn-by-turn) spatial resolution. Resolution refers in this context to the repeatability of the beam position measurement from one turn to the next. To put it more explicitly: if the beam were to be perfectly still at a given `CBPM` location, how much would the beam position measurement change turn-by-turn, i.e., what is the intrinsic precision of the `CBPM` system?

One `CBPM` module is made of four buttons. Each button generates an analog waveform when the beam (electric charges) passes by. The `CBPM` electronics collects one digitized sample of this waveform, the sample ideally aligned on the peak of the waveform. In the linear approximation, the horizontal (x) and vertical (y) positions of the beam are reconstructed using the following equations:

$$x = k_x \frac{(b_2 + b_3) - (b_0 + b_1)}{b_0 + b_1 + b_2 + b_3}, \quad k_x = 25.9 \text{ mm},$$
$$y = k_y \frac{(b_0 + b_3) - (b_1 + b_2)}{b_0 + b_1 + b_2 + b_3}, \quad k_y = 19.8 \text{ mm},$$

where b_0, b_1, b_2 and b_3 are the digitized amplitudes of the button signals. The factor k_x and k_y are the geometric factors that account for the shape of the beam pipe and can be changed to any value in the simulation. The button naming convention used in this note and in the code is shown in Fig. 1.

2 Toy Monte Carlo ingredients and configuration file

This section presents the ingredients of the toy Monte Carlo simulation. The various ingredients are set in a configuration file (see App. A for the configuration file in full). The relevant configuration parameters will be shown in the following sections assuming that three `CBPM`s are simulated:

```
1    # List of CBPM names to simulate
2    "cbpm": "12W2 12W3 12W",
```

2.1 Beam position

The starting point of the simulation is the position of the beam: its horizontal and vertical positions (x,y). The configuration parameters are:

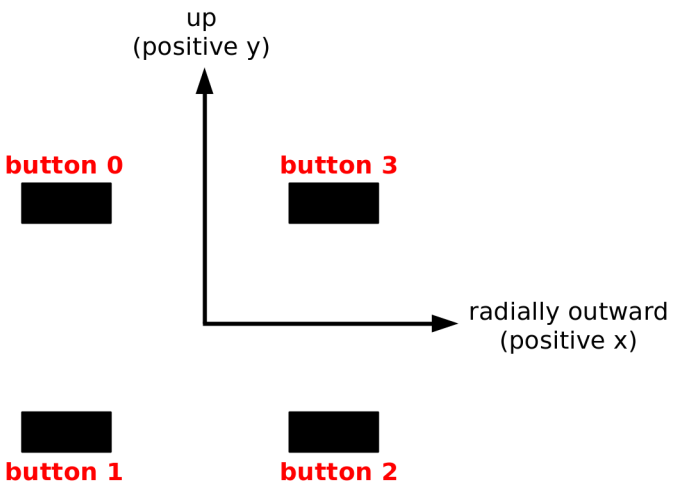


Figure 1: Naming convention of the four buttons.

```
    # Initial horizontal position at each CBPM [mm]
    "x_pos":[0,0,0],
    # Initial vertical position at each CBPM [mm]
    "y_pos":[0,0,0],
```

for the three `CBPM`s, with their units the millimeter. These are the initial positions for the first simulated turn. The center of the beam pipe corresponds to (x,y) = (0,0).

2.2 Button analog waveform

The next step consists in converting the beam position into a signal amplitude and an analog waveform for each of the four buttons for each `CBPM` module. The conversion is done in a linear fashion, i.e., excluding possible non-linear effects [1]:

$$
\begin{aligned}
A_{b_0} &= A_{(0,0)} \times (1 - \frac{\delta x}{k_x} + \frac{\delta y}{k_y}),\\
A_{b_1} &= A_{(0,0)} \times (1 - \frac{\delta x}{k_x} - \frac{\delta y}{k_y}),\\
A_{b_2} &= A_{(0,0)} \times (1 + \frac{\delta x}{k_x} - \frac{\delta y}{k_y}),\\
A_{b_3} &= A_{(0,0)} \times (1 + \frac{\delta x}{k_x} + \frac{\delta y}{k_y}),
\end{aligned}
$$

where $A_{(0,0)}$ is the reference signal amplitude for a beam perfectly centered on (x,y)=(0,0) as set in the configuration file:

[1] Non-linearities appear for a beam at large excursions

```
    # Reference amplitude of the button analog waveforms [ADU]
    "amplitude": [32768, 32768, 32768],
```

and δx and δy are the beam position displacements with respect to the center. The amplitudes A_{b_0}, A_{b_1}, A_{b_2} and A_{b_3} set the maximum amplitude of the simulated sine waveform for each button. The sine waveform has a 540 MHz frequency. It was shown analyzing button data (see [1]) that a 540 MHz sine waveform reproduces well within the region of interested ($\pm$100 ps of the peak) the typical shape of a button waveform.

2.3 Timing offset

The read-out channel associated with a button collects one digitized sample per analog waveform (per bunch passing by). The sample is ideally collected at the maximum of the waveform. However, the read-out channel can only align its sampling clock within steps of 10 ps. The algorithm responsible for aligning the sampling point on the maximum of the waveform is not perfect and it has been observed that the sampling points can be 2-3 10 ps units away from the maximum. If the four buttons of one `CBPM` module have different offset, the read-out amplitudes will be biased differently thus leading to an error in the position measurement.

The timing offset is implemented in the simulation as an offset in sampling the sine waveform. Without any offset, the sampling is performed at the maximum of the sine waveform. Any offset (positive of negative) would shift the sampling point about the maximum. The timing offset is set for each button of each `CBPM`. The relevant configuration parameters are:

```
    # Timing offset, set to true to apply
    "apply_timing_offset":true,
```

to apply the timing offset, and:

```
    # Timing offset [ps]
    "timing_offset":        [10,10,10,10,  10,10,10,10,  10,10,10,10],
```

to set the offset values in ps units.

2.4 Timing jitter

The sampling clock of a `CBPM` module has a jitter, meaning that the sampling point moves about its reference timing offset. The typical jitter was measured in the data to have a standard deviation of about 10 ps (see [2]). The sampling clock is common to the four buttons within a module and therefore the clock jitter can be expected to be correlated across the four buttons. The measurement of the correlation in the data was not conclusive. The simulation therefore has the possibility to correlate or not the clock jitter acros the four buttons. The relevant configuration parameters are:

```
    # Timing jitter, set to true to apply
    "apply_timing_jitter":true,
```

to apply the timing jitter. In the case of correlated jitter:

```
    # To correlate the timing jitter between the 4 buttons, set to true
    "timing_jitter_correlated": true,
```

to specify correlated jitter across the four buttons, and

```
    # Timing jitter common to all the buttons in one CBPM (correlated jitter)
     [ps]
    # Requires the 'timing_jitter_correlated' switch to be turned on
    "corr_timing_jitter": [1, 1, 1],
```

to set the amount of jitter per `CBPM`. In the case of un-correlated timing jitter:

```
    # Uncorrelated timing jitters for the individual buttons [ps]
    "uncorr_timing_jitter": [10,10,10,10, 10,10,10,10, 10,10,10,10],
```

to specify the jitter values per button per `CBPM`. The correlated jitter switch should be set to false:

```
    # To correlate the timing jitter between the 4 buttons, set to true
    "timing_jitter_correlated": false,
```

2.5 Electronic noise

The `CBPM` electronics has an intrinsic noise and added to it is whatever environmental noise their might exist. This noise was measured in data to be about 9 ADU (see [3]) and following a distribution resembling a Gaussian distribution. In the simulation, the noise is added randomly, drawing it from a Gaussian distribution, to the previously simulated button amplitude. The relevant configuration parameters are:

```
    # Electronic noise, set to true to apply
    "apply_noise": true,
```

to simulate the noise, and:

```
    # Electronic noise [ADU]
    "noise": [9,9,9,9, 9,9,9,9, 9,9,9,9]
```

to set its value per button per `CBPM`.

2.6 Beam position modulation and drift

The simulation allows for position drift(s) and modulation(s). The drift is specified in the configuration file in millimeter and its value represents the full drift over the total number of turns specified:

```
    # Horizontal drift over the total number of turns [mm]
    "x_drift":[0,0,0],
    # Vertical drift over the total number of turns [mm]
    "y_drift":[1.6384, 1.6384, 1.6384],
```

The modulation is simulated with two inputs: its standard deviation and its frequency. An arbitrary number of modulations can be simulated. In the following example, the same two vertical modulations are applied to the three simulated CBPMs, i.e., a 60 Hz modulation with a 1 mm standard deviation and a 20 kHz modulation with a 3 mm standard deviation:

```
    # Standard deviation(s) of the vertical modulation(s) [mm]
    "y_modulation_std":[0.2,0.2,0.2, 0.05, 0.05, 0.05],
    # Frequency(ies) of the vertical modulation(s) [Hz]
    "y_modulation_freq":[60,60,60, 0.75e3,0.75e3,0.75e3],
```

3 How to run the code

The CBPM toy Monte Carlo simulation code is available on GitHub at the following address:

```
https://github.com/atc93/CBPMsimulation
```

You can check out the code using Git:

```
$ git clone https://github.com/atc93/CBPMsimulation.git
```

or using SVN:

```
$ svn co https://github.com/atc93/CBPMsimulation.git
```

or simply download the code in a ZIP format from the website. To run the code with the example configuration file, simply run:

```
$ python cbpmsimulation.py config.json
```

This will create an output text file called toymc.dat. The format of the output file matches the format of an actual data file. The reason being that the same code can therefore analyze both data and simulation.

4 Button amplitude: sources of error

This section presents studies conducted to understand the button readings given the different sources of error. The sources of error will be added on top of each other as the sections go and are the ones defined in Sec. 2. The configuration file used in this section is shown in full in App A.

4.1 Ideal button readings

The ideal case corresponds to an error free system, meaning that the timing offset, timing jitter and electronic noise are not simulated. In this case, each button reads out only one amplitude value and the position of the beam is perfectly reconstructed: absolutely precise and accurate. Figure 2(a) shows a simulated button reading for this case.

4.2 Adding timing offset

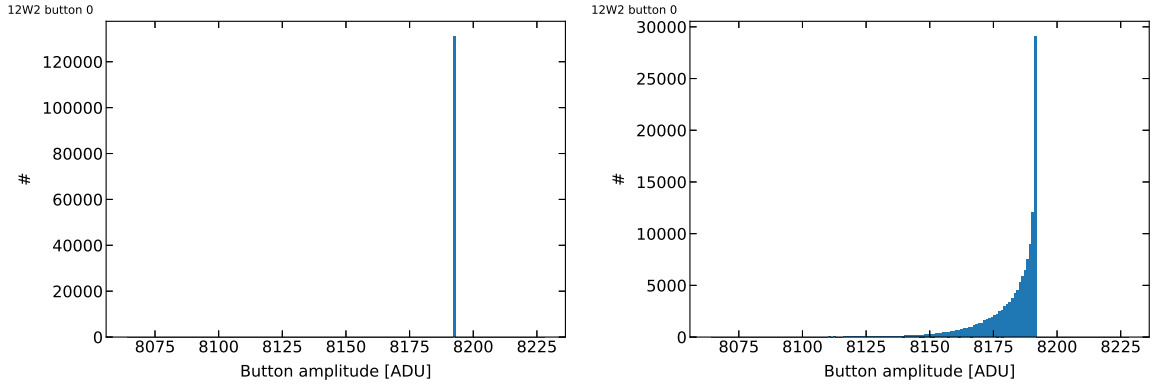
Introducing a timing offset leads to the beam position to be less accurate but still absolutely precise if the four button of one `CBPM` module have the same offset. Each button read-out will sample the sine waveform off the peak, always in the same fashion without varying (for fixed offset values). Figure 2(c) shows how the amplitude decreases from sampling off the peak. If the four buttons have different varying timing offset, the beam position will loose both accuracy and precision.

4.3 Adding timing jitter

Introducing a timing jitter leads to the sampling timing to move about the fixed timing offset on a turn basis, therefore the read-out amplitude changes on a turn basis. Figure 2(b) shows how the button reading is now a distribution. The asymmetry comes from the fact that there is a maximum amplitude corresponding to the top of the sine waveform, and a wide range of amplitudes sampling off the peak.

4.4 Adding electronic noise

The typical noise measured in the data is a randomly distributed noise according to a Gaussian distribution with a standard deviation of 9 ADU. Therefore, the noise broadens the distribution read-out by the buttons. The broadening is relatively more significant for lower average amplitude (given the noise is fixed). Figure 2(d) shows the broadening of the button reading. The shape of the distribution results from the convolution of the distribution from Sec. 4.3 with a Gaussian distribution coming from the noise.



(a) No error.

(b) Timing offset + jitter

(c) Timing offset.

(d) Timing offset + jitter + noise.

Figure 2: Simulated button amplitude for different error scenarios: (a) no error, (c) timing offset added, (d) timing jitter added and (d) electronic noise added.

5 Button amplitude: beam position drift and modulation

5.1 Beam position drift

As explained in Sec. 2.6, a drift of the beam position can be simulated in both the vertical and horizontal directions. Figure 3 shows a simulated example with the relevant configuration parameters defined in Sec. 2.6. The drift is only in the vertical direction.

5.2 Beam position modulation

As explained in Sec. 2.6, a modulation of the beam position can be simulated in both the vertical and horizontal directions. Figure 4 shows a simulated example with the relevant configuration parameters defined in Sec. 2.6. The modulation is only in the vertical direction.

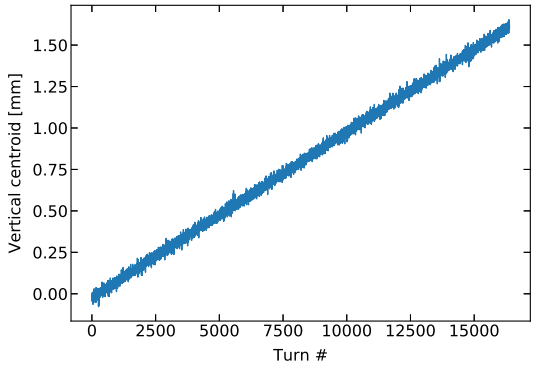


Figure 3: Simulated vertical position drift as a function of turn number. The drift is set to 0.1 μs per turn.

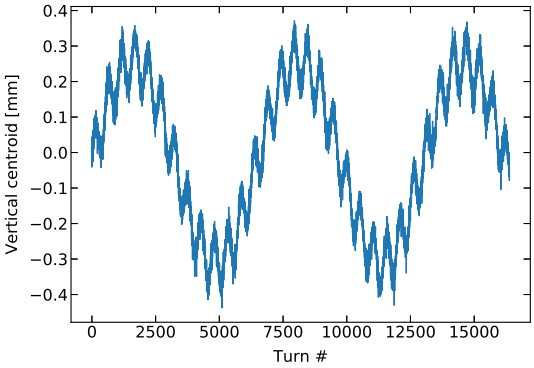


Figure 4: Simulated vertical position modulation as a function of turn number.

References

[1] Slides 16, 17 and 18 of CBPM meeting presentation (October 30, 2019): link

[2] Slides 13 and 14 of CBPM meeting presentation (October 30, 2019): link

[3] Slides 3 and 4 of CBPM meeting presentation (October 30, 2019): link

A Configuration file

```
{
        ##== General paramters ==##

        # Name of the output data file
        "output_file_name":"toymc.dat",
        # Print-out verbose
        "verbose":0,
        # Number of simulated turns
        "n_turns":131072,
        # Frequency of the sine waveform used as button response
        "frequency": 540e6,
        # Horizontal geometrical factor
        "kx":25.9,
        # Vertical geometrical factor
        "ky":19.8,
        # List of CBPM names to simulate
        "cbpm": "12W2 12W3 12W",

        ##== Various switches ==##

        # Electronic noise, set to true to apply
        "apply_noise": true,
        # Timing jitter, set to true to apply
        "apply_timing_jitter":true,
        # Timing offset, set to true to apply
        "apply_timing_offset":true,
        # To correlate the timing jitter between the 4 buttons, set to true
        "timing_jitter_correlated": false,

        ##== Per CBPM configuration ==##

        # Initial horizontal position at each CBPM [mm]
        "x_pos":[0, 0, 0],
        # Initial vertical position at each CBPM [mm]
        "y_pos":[-0.02522, -0.40095, 0.70323],
        # Horizontal drift over the total number of turns [mm]
```

```
        "x_drift":[0, 0, 0],
        # Vertical drift over the total number of turns [mm]
        "y_drift":[1.6384, 1.6384, 1.6384],
        # Standard deviation(s) of the horizontal modulation(s) [mm]
        "x_modulation_std":[0,0,0, 0,0,0],
        # Standard deviation(s) of the vertical modulation(s) [mm]
        "y_modulation_std":[0.2,0.2,0.2, 0.05, 0.05, 0.05],
        # Frequency(ies) of the horizontal modulation(s) [Hz]
        "x_modulation_freq":[0,0,0, 0,0,0],
        # Frequency(ies) of the vertical modulation(s) [Hz]
        "y_modulation_freq":[60,60,60, 0.75e3,0.75e3,0.75e3],
        # Reference amplitude of the button analog waveforms [ADU]
        "amplitude": [16384, 16384, 16384],
        # Timing jitter common to all the buttons in one CBPM (correlated
    jitter) [ps]
        # Requires the 'timing_jitter_correlated' switch to be turned on
        "corr_timing_jitter": [10, 10, 10],

        ##== Per button configuration ==###

        # Timing offset [ps]
        "timing_offset":        [10,10,10,10, 10,10,10,10, 10,10,10,10],
        # Uncorrelated timing jitters for the individual buttons [ps]
        "uncorr_timing_jitter": [10,10,10,10, 10,10,10,10, 10,10,10,10],
        # Electronic noise [ADU]
        "noise": [9,9,9,9, 9,9,9,9,  9,9,9,9]
}
```