Finding Energy Deviation (δ) from Spectrometer BPM Residuals: My Approach and Step-by-Step Implementation

Dawit Yerdea

August 18, 2025

1 Background

This project builds on a technical note prepared by my mentor, Zack Buschmann at SLAC, which outlines the methodology for deriving beam energy deviation from the residual orbit at a spectrometer BPM. In the first phase of the project, I developed and tested an orbit fitting code to model beam trajectories, and I validated that approach by presenting the results through 2D plots.

The next phase of the work focuses on extending this model to derive and implement an equation for calculating the relative energy deviation (δ) by using spectrometer BPM residuals. This includes testing of the method, designing the performance reference point for each measurement, and evaluating the average execution time. The ultimate objective is to integrate this work into a real-time feedback system, allowing energy deviation to be measured and corrected during beamline operation.

2 Model

First, using the technical note as a reference, I wrote a code for orbit fitting. In this phase, I will focus on the horizontal plane and use the transport model. The initial conditions are defined as ($\boldsymbol{x_0}, \boldsymbol{x_0'}, \delta_0$) at a reference location s_0,

$$\boldsymbol{x}_0 = \begin{pmatrix} x_0 \\ x_0' \\ \delta_0 \end{pmatrix}, \qquad \boldsymbol{b} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix},$$

where $\boldsymbol{x}_0$ are BPM(Beam Position Monitoring) sensor readings and $\boldsymbol{b}$ is the column vector of initial beam position and slope . Under the no-coupling assumption, each downstream BPM satisfies

$$\boldsymbol{x} - 0 \;=\; R_{11}\, x_0 \;+\; R_{12}\, x_0' \;+\; R_{16}\, \delta_0, \tag{1}$$

and, compactly,

$$\boldsymbol{b} \;=\; M\, \boldsymbol{x}_0, \qquad \boldsymbol{x}_0 \;=\; (M^{\mathsf{T}} M)^{-1} M^{\mathsf{T}} \boldsymbol{b}. \tag{2}$$

we denote the horizontal dispersion by $\eta(s_0) = R_{16}$ with units of m, thus, the dispersive orbit contribution at BPM is given by $\eta(s_0)\, \delta_0$.

3 Deriving δ_0 from a Spectrometer BPM

To determine the energy deviation, we use a BPM spectrometer located where the dispersion is large $\eta = R_{16}$. The initial conditions from upstream BPMs are fitted first, which allows us to

predict the expected betatron orbit at the spectrometer. The actual BPM measurement x_{meas} is then compared with this fitted prediction x_{fit}.

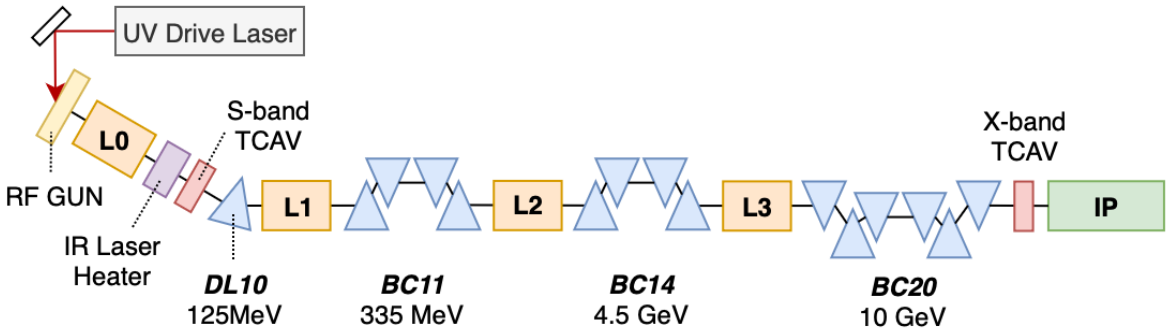

Figure 1: Beamline layout at FACET-II (courtesy of SLAC National Accelerator Laboratory, FACET-II).

The BPM regions DL10, BC11, and BC14 are chosen as shown above in Figure 1. We chose these regions because they are in spectrometer sections with high dispersion occurring, they include several BPMs for redundancy and more reliable fitting. They also make it possible to verify energy deviation at multiple points along the machine. These features make them ideal locations for residual-based energy measurements.

The energy deviation δ_0 can be extracted by comparing the BPM position with the fitted orbit. These x_0, x_0' have been fitted from upstream BPMs through eqn(2).

$$x_{fit} = R_{11}\, x_0 \; + \; R_{12}\, x_0'.$$

$$\delta_0 \; = \; \frac{\Delta x}{\eta} \; = \; \frac{x_{\text{meas}} - x_{fit}}{R_{16}}. \tag{3}$$

where η is the dispersion previously defined in eqn 2.

4 Concept and Implementation of functions

In this section, I will translate the main ideas into a practical numerical workflow. I will define the key functions and explain why each function is needed, and then walk through their step-by-step use. The steps as follow; loading orbit data, fitting the upstream model, computing spectrometer residuals, extracting delta from dispersion, converting it to energy offset and percentage error, and validating the results with plots and basic performance checks.

Concept

1. Load BPM readings data from `.mat` files. (note : since the beam-line operation has paused for a moment, I am getting the.mat file from the past run in FACET-II archive. When the beam-line operation resumes, I hope to plug live data to test the code).

2. Fit the orbit to estimate initial conditions (x_0, x_0', y_0, y_0') using upstream BPMs and the model M(Matrix).

3. Predict the betatron position at a spectrometer BPM, subtract from the measured value to form a residual .

4. Compute δ_0 through Eq. (3) using the model dispersion $\eta = R_{16}$ at that BPM.

5. Convert δ_0 to energy offset $\Delta E = E_{\text{design}}\delta_0$. $E_{\text{design}} = 13.5$ GeV, and % error $= 100|\delta_0|$.

6. Collect results from multiple files and BPMs, save the outputs into CSV logs, and run timing test to check performance.

4.1 Steps and Roles of each functions

`def load_orbit(filepath):-` Use orbit module to ingest saved file given by filepath .

- **Inputs:** `filepath` to `.mat`.
- **Outputs:** BPM names, longitudinal positions $\boldsymbol{s}$, and raw arrays $x_{\text{raw}}, y_{\text{raw}}$ (in m).
- **Steps:** open file $\rightarrow$ analize BPM table $\rightarrow$ convert units $\rightarrow$ return to orbit.

`def orbit_fit(f2m, bpmnames, x_raw, y_raw):-` fits BPM data using linear maps, returns orbit at single position s_0

- **Inputs:** machine model `f2m`, BPM list, raw x, y arrays.
- **Outputs:** fitted orbits $x_{\text{fit}}, y_{\text{fit}}$ at all BPMs, fitted orbit parameters $(x_0, x_0', y_0, y_0', \delta_0)$.
- **Steps:** build M from `f2m` $\rightarrow$ solve $(M^{\mathsf{T}}M)\boldsymbol{x}_0 = M^{\mathsf{T}}\boldsymbol{b}$ $\rightarrow$ compute fitted x_{fit} and y_{fit}.

`def plot_orbit(s_positions, x_raw, y_raw, x_fit, y_fit):` Visualization of fits and dispersive regions x_{raw} vs. s and x_{fit} vs. s; similarly for y.

`def measure_delta(f2m, bpm0, x0, spec_bpm, x_meas, E_design=13500, eta_min=1e-9, delta_warn=0`
Calculates energy deviation at a given spectrometer BPM.

- **Inputs:** `f2m` model, reference BPM index/name `bpm0`, fitted (x_0, x_0'), spectrometer BPM name, measured x_{meas}, design energy E_{design} (MeV), guard thresholds. also compute $\Delta E = E_{\text{design}}\delta_0$ (MeV), % error $= 100|\delta_0|$.
- **Output:** δ, ΔE, residual, x_{fit}, η, %error

`def compute_energy_offset(delta0, beam_energy_eV):` - Convert δ_0 to absolute energy offset.

- **Inputs:** δ_0, nominal energy in eV.
- **Output:** $\Delta E = \delta_0 \times E_{\text{nominal}}$ (same units as input).

`def process_energy_errors():` Batch process many files and/or spectrometer BPMs and show log results.

- **Steps:**
  1. enumerate `.mat` files $\rightarrow$ `load_orbit` each.
  2. `orbit_fit` to get (x_0, x_0') and x_{fit}.
  3. for each chosen spectrometer BPM, read x_{meas} and call `measure_delta`.
  4. log $\{\text{file}, \text{BPM}, \delta_0, \Delta E, \%\text{error}\}$ to CSV.

`def performance_test(num_trials=1000):` Currently under development.

4.2 Step-by-Step Methodology

Here's the way I've been running things so far, keeping each step adjustable so that I can debug them along the way.

1. **Sanity check:** Run `test()` to confirm file paths, unit conversions, and that a sample file loads properly.
2. **For each file:**
   (a) Load orbit data: $(\boldsymbol{s}, x_{\text{raw}}, y_{\text{raw}}, \text{BPM names}) \leftarrow$ `load_orbit(filepath)`.
   (b) Fit upstream BPMs: $(x_{\text{fit}}, y_{\text{fit}}, x_0, x_0') \leftarrow$ `orbit_fit(f2m,` BPM names$, x_{\text{raw}}, y_{\text{raw}})$.
   (c) For each spectrometer BPM, record the measured value: $x_{\text{meas}} \leftarrow$ position at that BPM.
   (d) Estimate energy deviation: $(\delta_0, \Delta E, \%\text{error}) \leftarrow$ `measure_delta(f2m, bpm0, spec_bpm,` $x_{\text{meas}}, E_{\text{design}})$.
   (e) Use `compute_energy_offset` to check units. I make sure the energy is first handled in eV before converting back to GeV or MeV.
   (f) Add results to CSV and store residuals for later regression tests.
   (g) Plot comparison with `plot_orbit`$(\boldsymbol{s}, x_{\text{raw}}, y_{\text{raw}}, x_{\text{fit}}, y_{\text{fit}})$.
3. **Batch mode:** Run `process_energy_errors()` across all orbit files and all spectrometer sections (DL10, BC11, BC14, ...).

So far, I've stopped here. Performance testing is something that i am working on currently. Finally, even though it is not possible to include all of the plots and CSV outputs in this report, I have attached two representative figures to demonstrate the type of results produced by the analysis. These figures provide an example of the orbit fitting plots as well as the numerical table output, which show how the measured data compare with the fitted model.

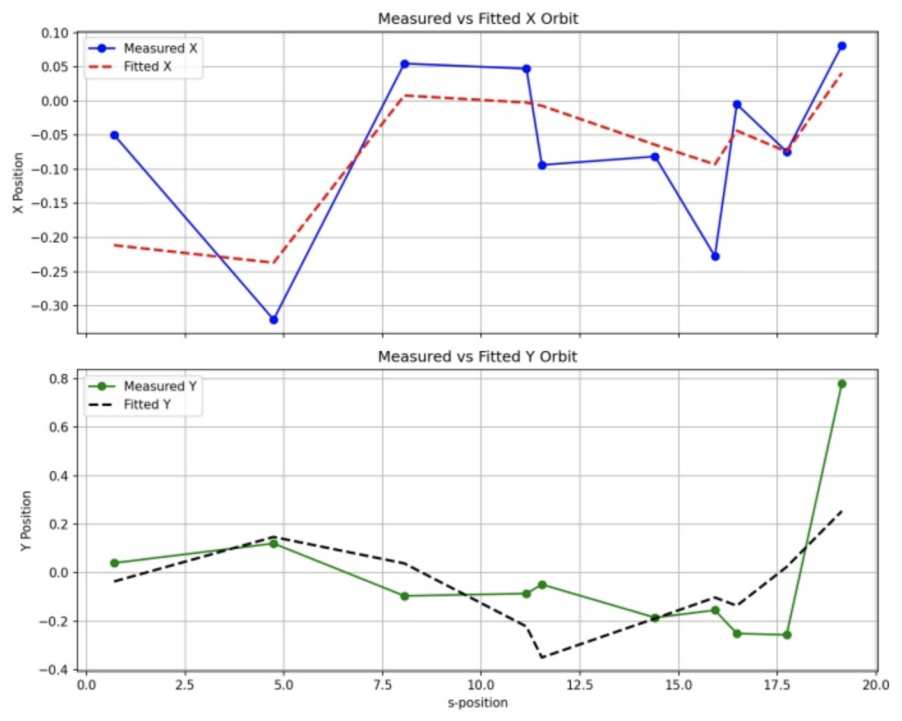

Figure 2: The figure illustrates how the measured orbit (dots) is compared with the fitted model (line), this verifies the correctness of the upstream fitting procedure.

	Filename	Region	BPM	x_meas (m)
0	05-04-2025_good_for_users.mat.mat	DL10	BPM10731	-0.074144
1	05-04-2025_start_of_owl.mat.mat	DL10	BPM10731	0.021940
2	05-09-2025_E300_laser_aligned.mat.mat	DL10	BPM10731	-0.082511

	x_model (m)	eta_x (m)	delta	delta_E (MeV)	% Error
0	-0.074144	0.01	-2.568304e-12	-3.467211e-08	-2.568304e-10
1	0.021940	0.01	-3.618296e-12	-4.884699e-08	-3.618296e-10
2	-0.082511	0.01	-1.914011e-13	-2.583915e-09	-1.914011e-11

Figure 3: Example output from the orbit-fitting and energy-deviation pipeline. This table demonstrates how the workflow extracts δ, ΔE, and percentage error from measured BPM data.

Disclaimer

I put this summary together using notes and conversations with Zack Buschmann but the way It's written and explained reflects my own take and understanding.