AIOP Photon Source Data Analysis

Will Imoehl

May 2025

Contents

1 Introduction/Motivation

The AIOP project aims to automate the coherent peak ”nudging” procedure by using an ML algorithm to continuously adjust the pitch and/or yaw of the diamond. In GlueX, the polarization of the photon peaks near the coherent edge, which is where the steep dropoff in energy occurs. By automating the energy nudging procedure, we ensure a more consistent coherent edge position over time, which will reduce the run-to-run spread in the beam energies. When integrating over the full run period, this will result in a sharper coherent peak edge, as well as a higher level of polarization. This improvement will reduce systematic uncertainties in the polarization, which could make our physics results more sensitive.

Would be good to have a viz of a good coherent peak, and what it looks like if you do time-weighted avg of a couple good ones and a bad one. Maybe with polarization too.

The stated goal of this project is to keep the coherent peak energy within a ±10 MeV window of the desired set-point energy over the full run period. While the primary handle that Hall D has on the coherent edge position is the pitch and yaw of the diamond, we know that there are other variables that can also affect the photon beam energy. These include changes in the position or energy of the electron beam, thermal effects, vibrations on the goniometer on which the diamond is mounted, and diamond degradation over time [1]. Patrick is focusing on studying diamond degradation over time and the effect of the x and y position of the electron beam, while my studies are focused on how nudge sequences change the coherent edge position in existing data. This includes measuring how much the energy changes per nudge for each run period, as well as understanding the impact of backlash in the goniometer.

'HD:coda:daq:run_number'	'HD:coda:daq:status'	'HD:CBREM:PLANE'	'HD:GONI:RADIATOR_ID'
'HD:GONI:X'	'HD:GONI:X.RBV'	'HD:GONI:Y'	'HD:GONI:Y.RBV'
'HD:GONI:PITCH'	'HD:GONI:PITCH.RBV'	'HD:GONI:YAW'	'HD:GONI:YAW.RBV'
'HD:GONI:ROLL'	'HD:GONI:ROLL.RBV'	'HD:CBREM:REQ_EDGE'	'HD:GONI:RADIATOR_NAME'
'HD:CBREM:EDGE'	'HD:CBREM:WIDTH'	'AC:inner:position:x'	'AC:inner:position:y'
'IPM5C11B.XPOS'	'IPM5C11B.YPOS'	'IBCAD00CRCUR6'	'HALLD:p'

Table 1: Table summarizing all the EPICS variables we get from the archive.

2 Data Collection and Processing

This section describes the procedures I use to obtain, clean, and merge the data.

2.1 Obtaining Data From myget

The myget [3] package can retrieve archived data from Mya. Specifically, myget will report each time a variable changes in the Mya archive. We use myget [3] to obtain the full history for each variable listed in Tab. 1. An example myget command showing how we obtain the run number for Spring 2020 is shown below.

```
myget -m history -c 'HD:coda:daq:run_number' -b '2020-01-11 00:00:00' -e '2020-09-21 12:00:00' | tee run_number.txt
```

This procedure is carried out for the Spring 2020, Spring 2023, and Spring 2025 run periods. A list of the exact commands used are given in Appendix B

Most variables are self explanatory. The bottom row has the x and y position, current, and energy of the electron beam. HD:CBREM:WIDTH is the measured uncertainty in the coherent edge position. A link to the spreadsheet showing the EPICS variables and their definitions is given as Ref. [4].

2.2 Conversion to .csv files

The output from myget is initially saved as a series of 24 .txt files for each run period. The code in

```
cleanAndConvertData_txt_to_csv.py
```

converts each of these .txt files into .csv files. In other words, it will take, for example

```
ac_x.txt -> ac_x.csv
```

During this conversion process, the following data cleaning steps are performed:

1. any rows with a "<< Network disconnection >>" message are removed.
2. Time stamps are only recorded every second. Sometimes, there are multiple recorded values in less than a second, especially for the readback values. In these cases, we only take the first row with the given time stamp.
3. Time stamps are reformatted to look like "[YEAR]-[MONTH]-[DAY]T[HOUR]:[MINUTE][SECOND]", which can automatically be converted into a datetime object in Python.

2.3 Merging multiple .csv files into a master .csv

The code for this section is in

```
merge_all_csv.ipynb
```

Table 2: Table showing a simplified example of the initial DataFrame after all outer joins are complete. The variable values are made up for this example.

DateTime	Run	DAQ:STATUS	GONI:YAW	GONI:YAW.RBV	EBEAM:CURRENT
2021-09-17T01:11:12	NaN	0	0.80	0.80	150
2021-09-17T01:12:12	NaN	NaN	0.90	0.82	149
2021-09-17T01:12:13	NaN	NaN	NaN	0.84	NaN
2021-09-17T01:12:13	NaN	NaN	NaN	0.86	150
2021-09-17T01:12:13	NaN	NaN	NaN	0.88	NaN
2021-09-17T01:12:13	NaN	NaN	NaN	0.90	151
2021-09-17T01:15:00	70356	1	NaN	NaN	150
2021-09-17T01:15:05	NaN	NaN	NaN	NaN	152
2021-09-17T01:15:37	NaN	NaN	NaN	NaN	NaN
2021-09-17T01:16:00	NaN	2	NaN	NaN	150

2.3.1 Details on Merging and Handling NaN's

The previous step gives us 24 .csv files, one for each variable of interest. We read each of these files into a separate pandas DataFrame, and then perform an outer join on the Date/Time column. If a variable does not have a value at a certain time, the missing value is filled with a NaN during the joining procedure. We show the number of non-null values for each variable and each run period in Fig. 1. Measured beamline variables have the most non-null values, while set-point variables have substantially fewer recorded values.

To visualize how the data may look after the joining process, Tab. 2 shows a simplified, hypothetical table that could result from the merge process.

There are three types of variables here:

1. Run number: the run number only changes once the DAQ reaches the prestart phase (status 1). The remaining values will be NaN, since myget only records values when they change.

2. Desired/setpoint variable values: These only have values if the set point changed. Therefore, a NaN value means we can forward fill the last non-NaN value.

3. Measured/Readback variable values: these have fewer missing values. If there is a NaN at a given time, it means there was either a "<<Network disconnection >>" or that the variable is the same as the last reading. In either case, it is reasonable to forward fill.

To deal with missing/NaN values, we use forward-fill. This should give the same behavior as mySampler, which is what Jiawei was using for his analysis.

After the forward-fill, there are two issues. The first few rows of the table may have NaN's that still exist, because there is no value to forward fill here. To deal with these, we back-fill after the initial forward fill. A better solution may just be to drop these initial rows, since the set points were necessarily different before this.

The second issue comes with the run number. After the forward fill, run number 70356 will extend from when run 70356 prestarted all the way to the next run that has prestarted. This means that all nudges before the run prestarts will be assigned to the previous run. To fix this, we reassign run numbers such that the run number follows the DAQ:STATUS sequence $0 \rightarrow 1 \rightarrow 2$, instead of the original sequence of $1 \rightarrow 2 \rightarrow 0$.

After these steps are carried out, the hypothetical table above is transformed to the one shown in Table 3.

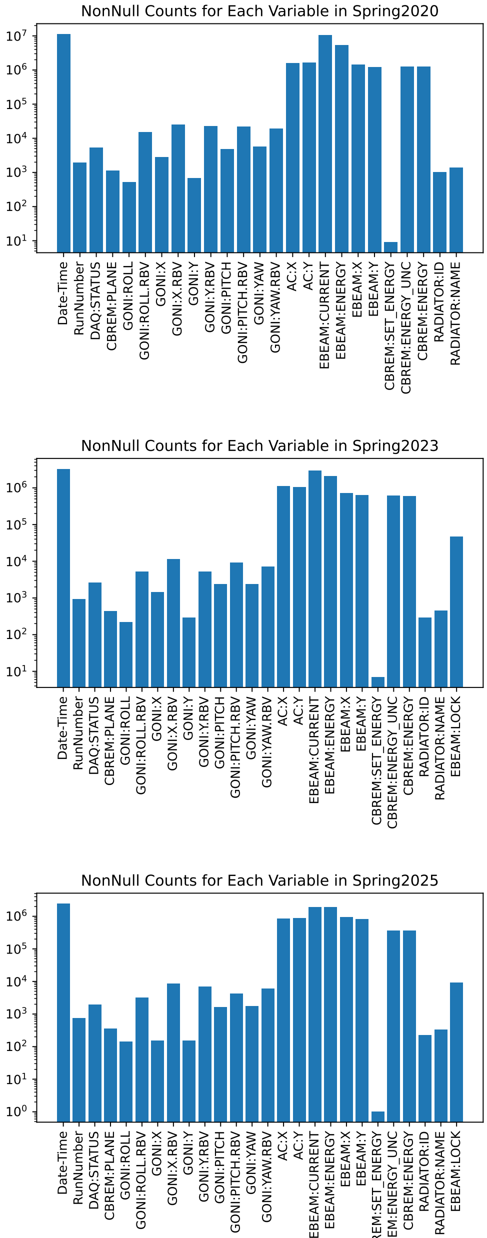

Figure 1: Figure showing the number of not NaN values for each variable for Spring 20 (top), Spring 23 (middle), and Spring 25 (bottom). Measured values have the highest counts, while setpoint values are relatively sparse.

Table 3: Table showing a simplified example of the final merged DataFrame after the outer joins and cleanup steps are completed. The variable values are made up for this example.

DateTime	Run	DAQ:STATUS	GONI:YAW	GONI:YAW.RBV	EBEAM:CURRENT
2021-09-17T01:11:12	70356	0	0.80	0.80	150
2021-09-17T01:12:12	70356	0	0.90	0.82	149
2021-09-17T01:12:13	70356	0	0.90	0.84	149
2021-09-17T01:12:13	70356	0	0.90	0.86	150
2021-09-17T01:12:13	70356	0	0.90	0.88	150
2021-09-17T01:12:13	70356	0	0.90	0.90	151
2021-09-17T01:15:00	70356	1	0.90	0.90	150
2021-09-17T01:15:05	70356	1	0.90	0.90	152
2021-09-17T01:15:37	70356	1	0.90	0.90	152
2021-09-17T01:16:00	70356	2	0.90	0.90	150

2.3.2 Casting Variable Types

The variables in the DataFrame are all automatically converted to floats during the merging process. To reduce the size of the data a bit, we convert 'RunNumber', 'DAQ:STATUS', 'CBREM:PLANE', and 'RADIATOR:ID' to integers.

2.4 Filtering Data

Originally, we filtered out runs that were not labeled as production runs, but this causes us to lose a handle on the full goniometer history, which is important for flagging goniometer backlash.

For Spring 2020, the coherent edge fitting script was broken for the first portion of the run period. We remove all runs before 72094, which is when the script was fixed.

2.5 Defining New Variables

In this subsection, we define new variables in the DataFrame to make visualizing the nudges easier.

The first variable we create is 'goodRun'. This is True if the run number is in the production set for the given run period, and False otherwise. Since production runs are not defined for Spring 23 and 25 yet, this variable is always True for those two run periods.

We also want to be able to visualize variables as a function of time, so we create a new column that converts the Date-Time string into a timestamp using the Python datetime package:

```
from datetime import datetime
time_convert = lambda x: x.timestamp()
df_good['TimeStamp'] = pd.to_datetime(df_good['Date-Time']).apply(time_convert)
```

Before defining variables related to the nudges, we output initial merged .csv files, in case the kernel crashes during the execution of the next steps. Crashes are much less frequent with the most recent version of the code. The output files are:

```
csv_data/merged/Spring20_initialMerged.csv
csv_data/merged/Spring23_initialMerged.csv
csv_data/merged/Spring25_initialMerged.csv
```

2.5.1 Variables Describing Nudges

The next step is to create variables related to changes in the pitch and yaw set points. We defined a nudge to be any change in the set point for the pitch and yaw. There are two types of variables related to nudges: run-wide properties and nudge-specific properties. For instance, run-wide variables include:

- RunHasNudge - True if there is at least one nudge in the run.
- NudgeDuringRun - True if there was a nudge while DAQ was on (status 2)
- TotalNudgeSize_thisRun - the total (quadrature sum) of the change in pitch and yaw from the start to the end of the nudge sequence
- NudgeSequence - string corresponding to the sequence of up ('u') and down ('d') nudges, for instance 'uuudduu'
- NudgeSequenceReduced_Correct - simplified nudge sequence properly accounting for the effect of backlash
- NudgeSequenceReduced_Naive - simplified nudge sequence ignoring the effect of backlash
- TotalNudges_thisRun - the total number of nudges that occur during the run
- RunHasAbnormalNudge - boolean - some nudges are abnormally large $> 1^{\circ}$. These are likely the result of an orientation change.
- RunHasBacklash - True if the goniometer reverses motion in pitch or yaw during this run.

Nudge-specific variables include:

- NudgeOccurred - True only for rows in which the pitch and yaw set points changed.
- BacklashNudge - True if the current nudge involves reversing the direction of the goniometer motor in either pitch or yaw. It is also true until the backlash has cleared, which takes 2 nudges for 0/90, and 3 for 45/135.
- NudgeNumber - a unique integer for each nudge in the run.
- NudgePitchSize - signed change in the pitch set point, implemented using pandas.diff()
- NudgeYawSize - signed change in the yaw set point, implemented using pandas.diff()
- PitchMotionSize - signed change in the pitch readback value, indicating active goniometer motion
- YawMotionSize - signed change in the yaw readback value, indicating active goniometer motion.
- MotionDone - True only if the readback values have stabilized
- NudgeEnSize_10s - rough estimate the new energy 10 seconds after the nudge, but very sensitive to noise
- CBREM:ENERGY_LAG_ADJUSTED - the coherent edge energy shifted 5 seconds earlier in time, to account for the roughly 5 second delay from the fitting script.

Other variables include:

- ConfigChange_inProgress - True if switching radiator, orientation, or PARA/PERP Not currently implemented

In order for a nudge to be defined, we check to make sure the CBREM:PLANE variable is either 1 or 2. When this variable is 0, it indicates switching between PARA and PERP configurations.

After this process is complete, the resulting DataFrames are saved to

```
csv_data/merged/Spring2020Merged_goodRuns.csv
csv_data/merged/Spring2023Merged_goodRuns.csv
csv_data/merged/Spring2025Merged_goodRuns.csv
```

We investigate correlations between the electron and photon beam parameters in nonudge-study.ipynb. In add_beam_up_time_combined.ipynb, we create a simplified data frame that excludes all nudge info from the above .csv files, and also add in a variable for the length of time since the last beam drop. We define a beam drop as any time the electron beam current drops below 30 nA. These are saved to

```
csv_data/merged/Spring2020_with_up_time.csv
csv_data/merged/Spring2023_with_up_time.csv
csv_data/merged/Spring2025_with_up_time.csv
```

This whole nudge defining process does not check for amorphous runs, so weird behavior can happen if the run is amorphous. Working on incorporating Radiator Name to check if the radiator is amorphous.

2.6 Sorting Runs

We sort runs into several categories. The following list outlines the classes and what we use them for.

- No nudge events are used to look at correlations between the electron beam parameters and the coherent edge energy. The code is in nonudge-study_combined.ipynb
- Single nudge events are used to visualize nudge sequences and to estimate the energy change per nudge. The code is in singlenudge-study.ipynb
- Multiple nudge events are used to get a more reliable estimate of the energy change per nudge. The code is in multinudge_combined.ipynb.
- Backlash runs are used to estimate the effect of backlash in the goniometer. The code is in backlash_study_combined.ipynb

Fig. 2 shows the percentage of runs that fall into each category.

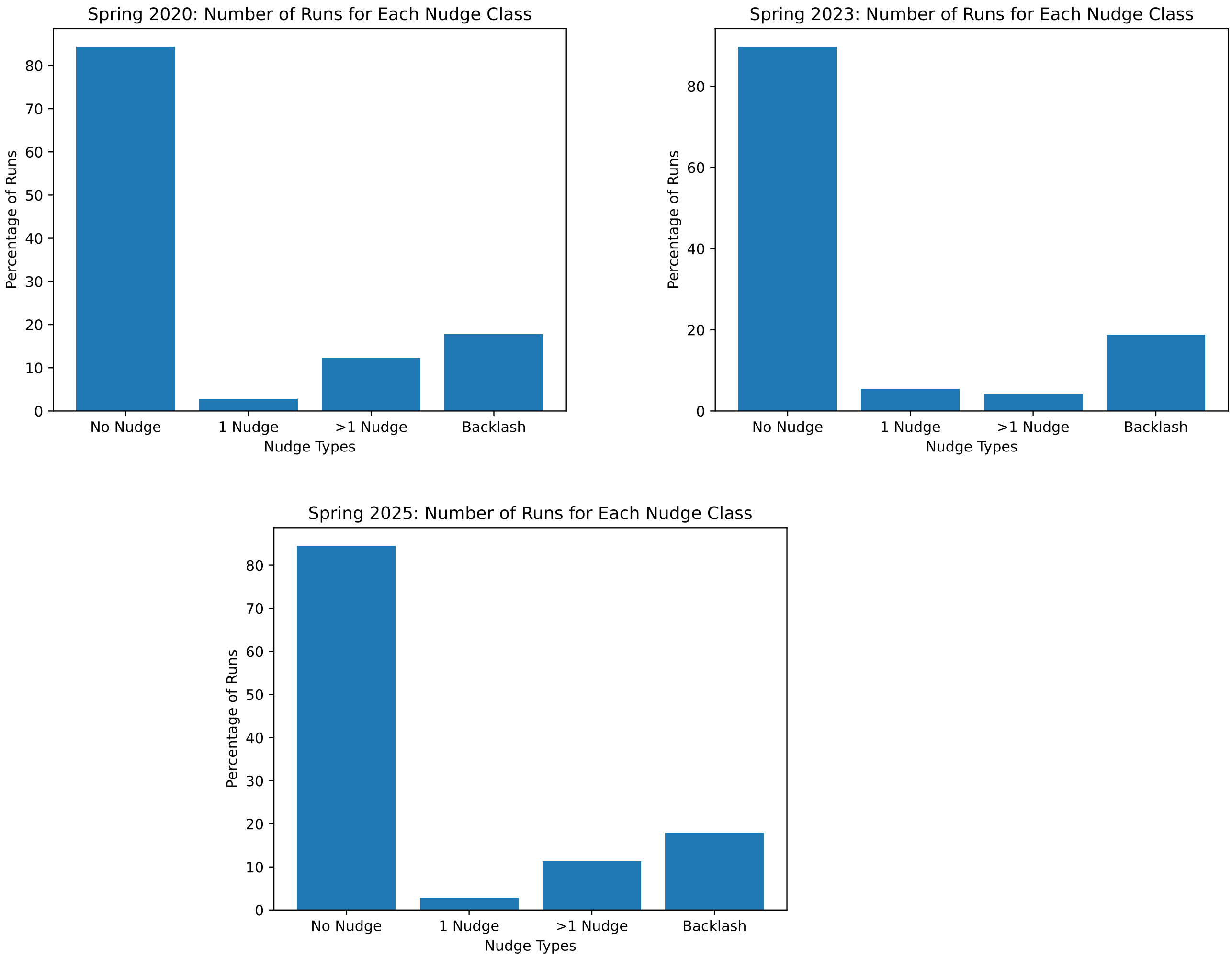

Figure 2: Percentage of runs with no nudges, 1 nudge, more than 1 nudge, or backlash for each run period.

3 Variables Impacting Coherent Edge Energy

We know that nudging the pitch and yaw values will change the energy of the coherent edge, but there are several other variables that can impact it too. The full list of variables we expect to have an impact on the coherent edge energy are:

- Pitch
- Yaw
- Electron beam X position
- Electron beam Y position
- Electron beam current
- Electron beam energy

We look at the impact of changing pitch and yaw in Section 4. In this section, we investigate the correlations between the coherent edge energy and the electron beam positions (x,y), the electron beam current, and the electron beam energy. The code to produce the plots here is in nonudge-study_combined.ipynb.

3.1 Procedures

We remove runs that have nudges, which gives us a clean data sample where pitch and yaw are stationary. This means that only the electron beam parameters should change the coherent edge energy.

This analysis has a few potential pitfalls. The pitfalls and the efforts we make to mitigate their effect are listed below.

1. There is a delay between when the electron beam conditions change to when the coherent edge updates. This delay occurs because there is a fitting script that needs to be rerun after the tagger scalers are updated. Fig. 3 shows the energy change versus the number of seconds since the last beam drop. We see that it takes roughly 4-5 seconds for new values of energy to appear. To try to reduce the effect of this time delay, we will use a coherent edge energy adjusted for time lag, which is shifted 5 seconds from the original time.

2. Electron beam conditions can change more rapidly than the coherent edge can. Rapid changes are unlikely to be reflected in the coherent edge position.

3. Anytime the beam drops, all variables will go to zero, regardless of whether they are correlated with each other. We will mitigate this effect by requiring the electron beam to have been up for at least 30 seconds, which removes the drops and the potential unstable beam periods directly following a beam drop.

4. Forward-filling missing values could "create" correlations that are not really there. It could also overemphasize the data points that occur before the beam drops, since they will be carried forward until the beam is restored.

5. The correlation matrices/correlation coefficients assume a linear relationship. There is no reason to suspect a strictly linear relationship here.

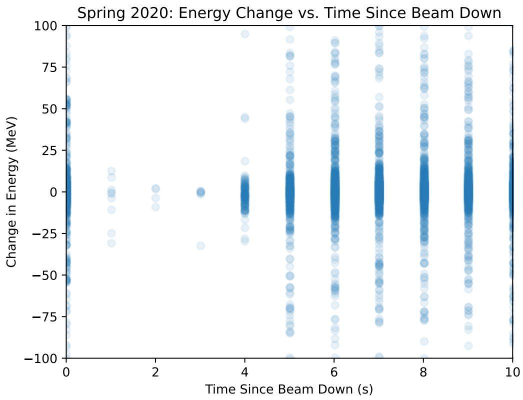

Figure 3: A plot showing the difference in coherent edge energy as a function of time since the last electron beam drop. There are very few points before 4 seconds, indicating that it takes a minimum of 4 seconds for the fit script to give a new energy value.

3.2 A Naive Look at Correlations

In this section, we ignore all the warnings given above and look at the correlation map for the electron beam position, current, and energy related to the coherent edge position adjusted for time lag. We do this separately for each orientation and run period. We require the beam to have been "up" for at least 30 seconds to ensure stable beam conditions exist. The results are shown in Fig 4 for all three run periods. Even for the same orientations, we do not see the same correlation values across run periods. This is likely due to the fact that these plots are integrating over large time periods and beam conditions. A better way to look at these correlations is to focus on a smaller time period, like a single run, and look how each of the electron beam variables influences the coherent edge position.

3.3 Extreme Variations of Each Variable

In this section, we try to find individual runs that have a large variance in each electron beam variable. We start by focusing on the electron beam position and then move onto the current and electron beam energy variables. When possible, we show examples from multiple run periods and diamond orientations.

3.3.1 Electron Beam Position

Generally, there is a beam lock that constrains the position of the electron beam to a narrow window in x and y. In this section, we look for runs where the beam lock appears to be inactive. These runs generally have larger changes in beam position, which will make it easier to see correlations between the electron beam x and y positions and the coherent edge position. We show the most dramatic example of large changes in beam position in Fig. 5. We see the x position of the electron beam change by up to 4 mm, which results in coherent edge energies between 8500 and 8600 MeV. We can see that there is not a strictly linear relationship between E_γ and the electron x position, but in the typical range around -0.5 mm, the behavior looks approximately linear.

Figure 4: Correlation coefficients for four electron beam variables versus the time lag-adjusted coherent edge energy for Spring 2020.

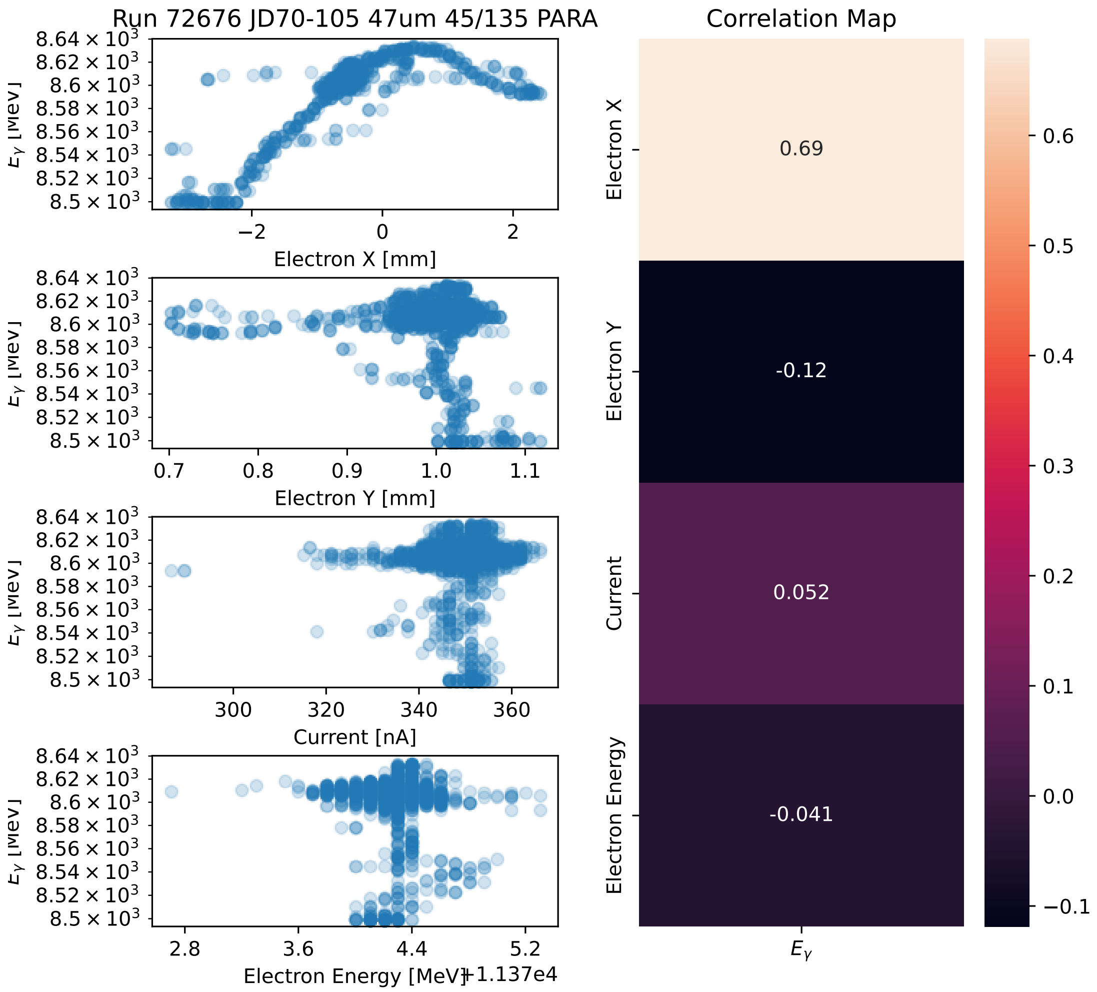

Figure 5: (left) Plots of the time lag-adjusted coherent edge energy as a function of electron beam x position (top), y position (2nd row), beam current (3rd row), and electron beam energy (final row). The right column shows the correlations between each variable and the coherent peak energy.

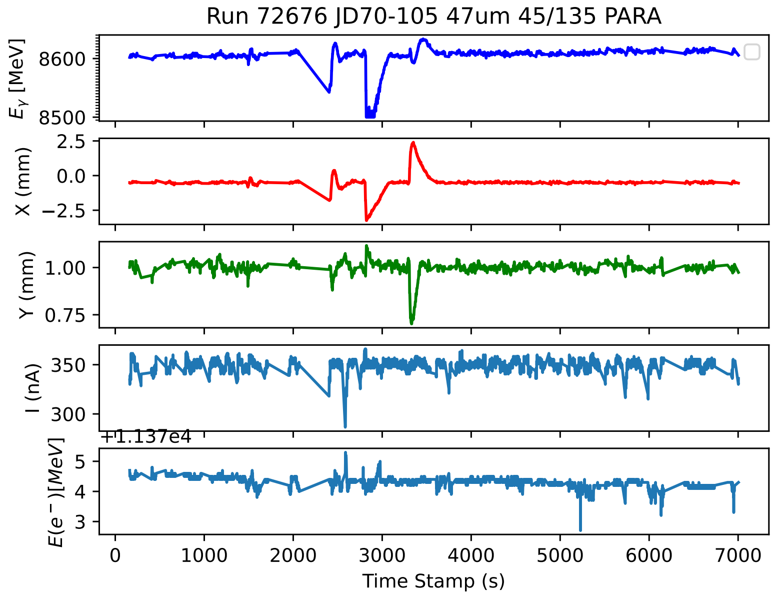

Figure 6: Time series of Run 72676, which has large fluctuations in both the x and y positions of the electron beam.

We also look at the time series progression for this run in Fig. 6. We see that jumps in the beam x position seem to precede jumps in energy by a few seconds, which we know is roughly the time it takes the coherent edge position to update.

Next, we look for a run that has a large variance in y. The correlation plots are shown in Fig. 7. We see both x and y have large correlations with the coherent peak energy in this case. The corresponding time series plot is shown in Fig. 8.

3.3.2 Electron Beam Current

Next, we look for a run with large variances in electron beam current. The variances are best visualized in a time series, as shown in Fig. 9. There are very clear oscillations of the beam current as a function of time. The corresponding correlation plots are shown in Fig. 10. One interesting thing to note is that the correlation with the electron current is relatively small, but correlations with electron beam y position and electron energy are large.

3.3.3 Electron Beam Energy

Changes in the electron beam energy will obviously have an impact on the coherent edge position. We show a dramatic example of this in Fig. 11. The coherent edge position slowly drifts over time by 1 MeV, and appears to induce a roughly 10 MeV change in the coherent edge position. The corresponding correlation plots are shown in Fig. 12.

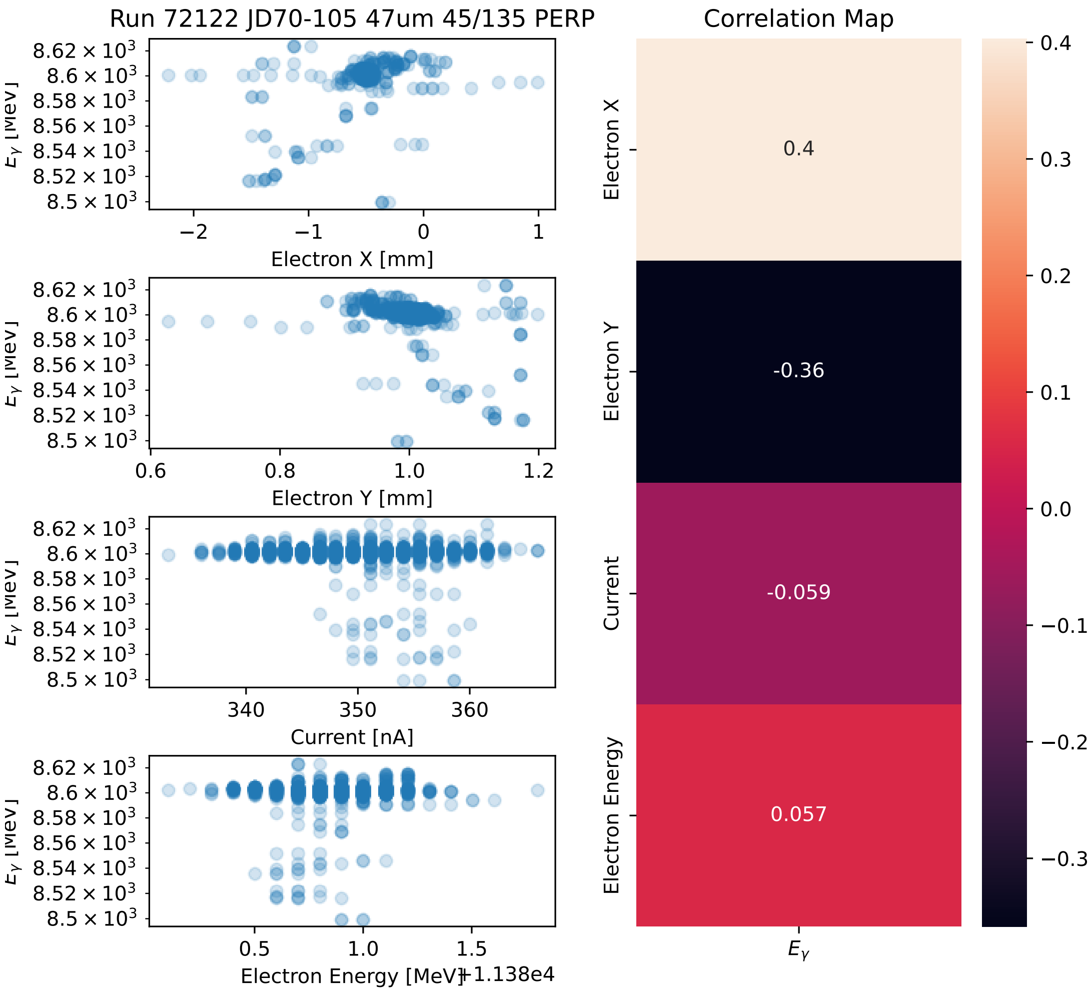

Figure 7: Plots of time lag adjusted coherent edge versus electron beam variables (left) and the corresponding correlation coefficients (right) for Run 72122, which has large variations in the electron y position.

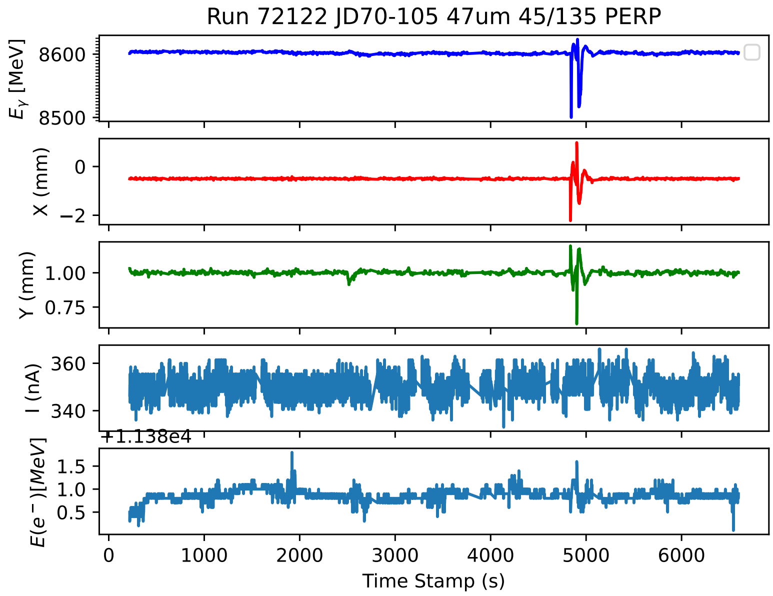

Figure 8: Time series plots of electron beam and coherent edge position for Run 72122, which has large y fluctuations.

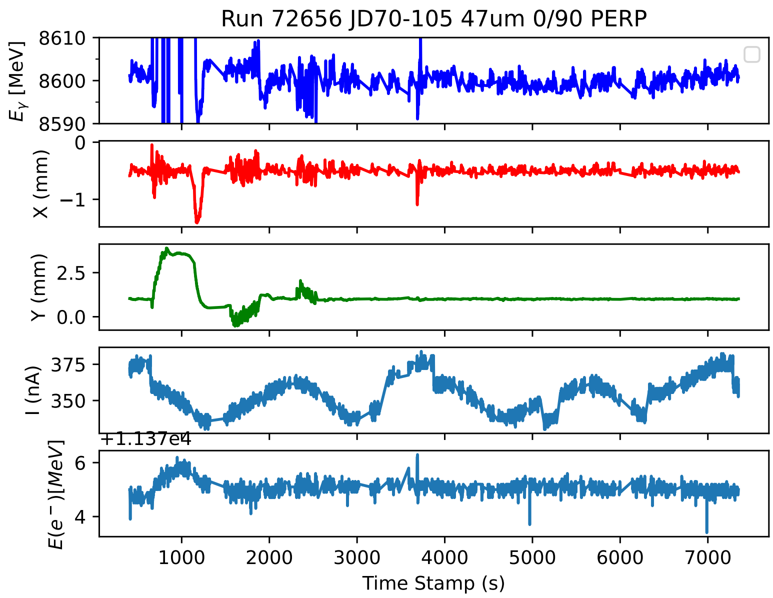

Figure 9: Time series plot of Run 72656, where the electron beam current undergoes oscillations.

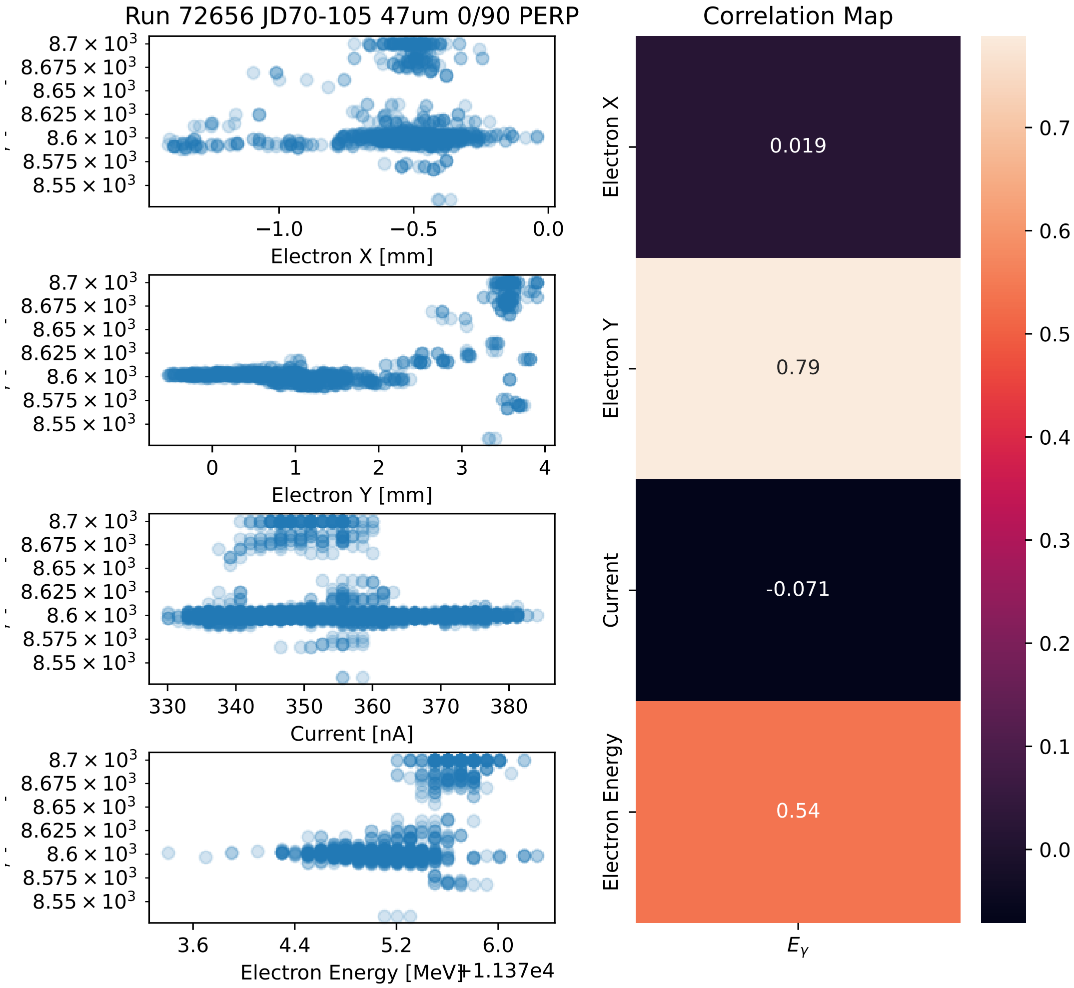

Figure 10: Correlation plots for Run 72656, which has a large periodic fluctuation in beam current.

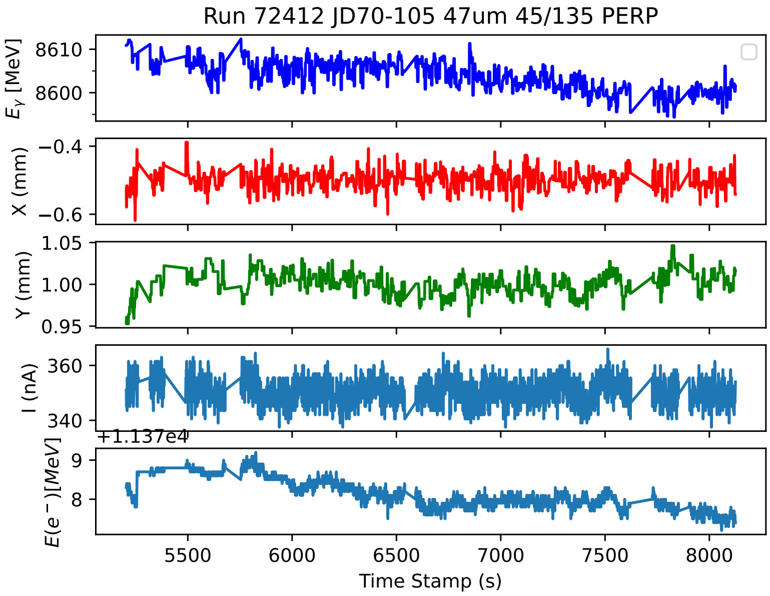

Figure 11: Time series plot for a run where the electron beam energy changes roughly 1 MeV, which appears to induce a change in the coherent edge position of around 10 MeV.

3.4 Summary of Studies

Here we only looked at a tiny fraction of the available runs, but we see clear examples where the electron beam x and y position impact the coherent edge position. We also saw that the electron beam energy change can have a large impact on the coherent edge energy. Things are less clear for the electron beam current. The correlation maps for each orientation show no significant correlation, with the largest correlations on the order of 0.2 for certain orientations.

The conclusion of these studies is that x and y appear to be the most important variables in determining the beam energy. If the electron beam energy is not stable, it also needs to be taken into account. Including the beam current in the model may be useful, but so far there is no "smoking gun" plot that shows that it can impact the coherent edge position. Of course, using the beam current as a proxy variable for whether or not there is beam could be useful, but that is likely redundant with whether the x and y position values are non-zero.

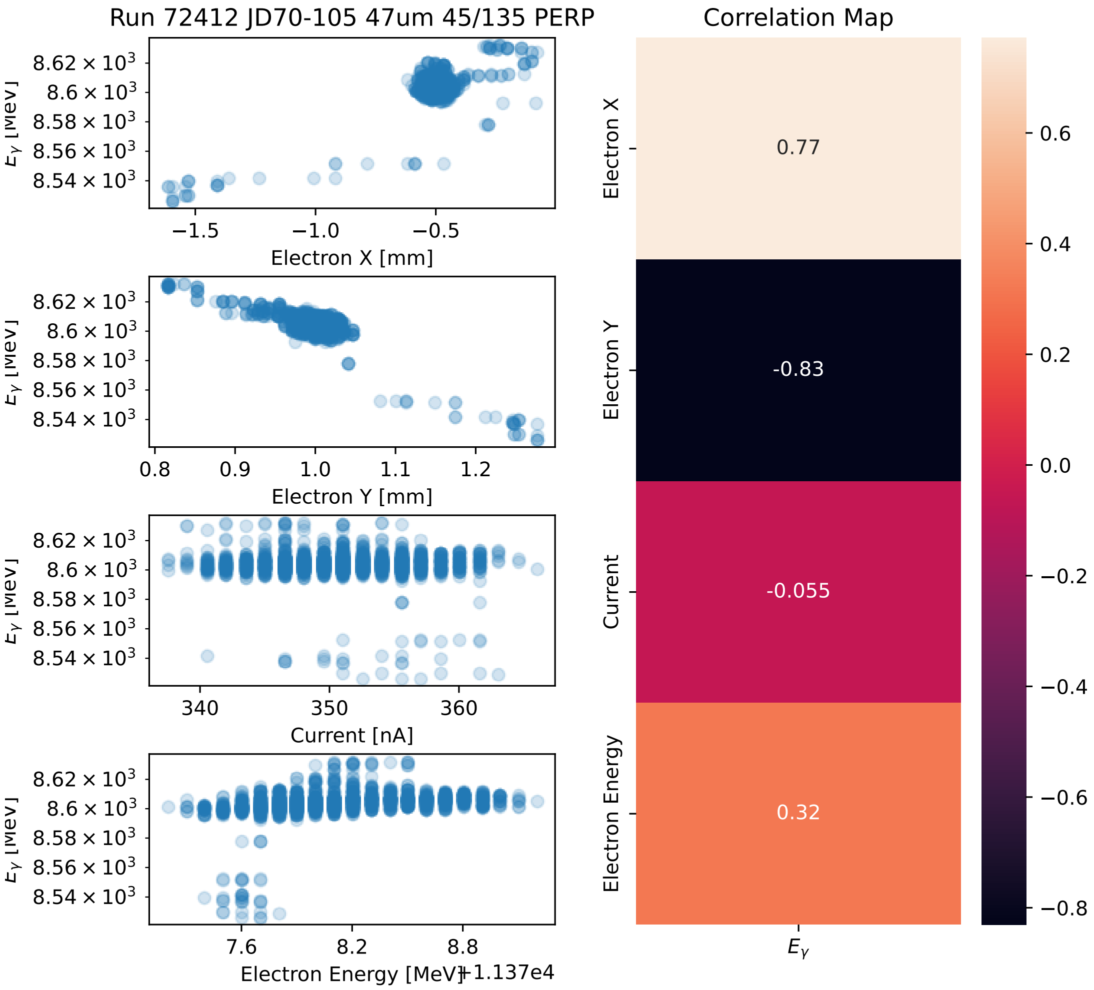

Figure 12: Correlation plots for Run 72412, where the electron beam energy appears to change 1 MeV over the course of the run.

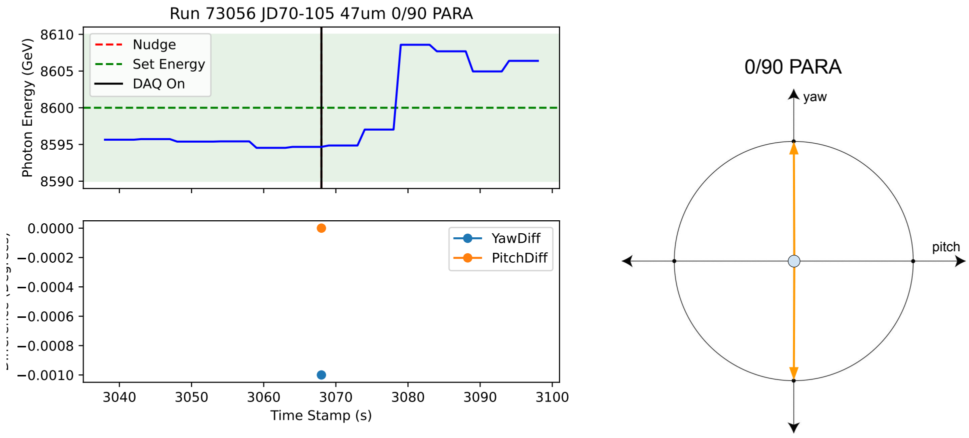

Figure 13: Left: example nudge for the 0/90 PARA configuration. Right: diagram showing nudges only occur in the ± yaw direction for 0/90 PARA.

4 Visualizing and Understanding the Nudges

The code for this section is in singlenudge-study.ipynb

In this section, we visualize how the photon beam coherent edge energy changes in response to nudges. We used production runs from Spring 2020, Spring 2023, and Spring 2025. For Spring 2020, we only used runs after the coherent edge-fitting scripts were fixed.

We start by looking at runs that have single-nudge events. This allows us to identify

1. the time it takes the beam energy to change after a nudge.
2. how much the energy changes for a single nudge.

For each of these single nudge events, we produce a plot that includes the measured photon energy as a function of time (blue), the desired photon energy (dashed green), and the acceptable beam energy range (±10 MeV of desired, green shaded region). For each instant that a nudge occurs, we include a vertical red dashed line. On the lower portion of the plot, we also include how large the change in pitch and yaw was for each nudge. The solid black line indicates the time when the DAQ was turned on for the run.

Nudges can be small (1 millidegree) or large (10 millidegrees). In addition, since we only record values every second, you can also have integer multiples of these step sizes if somebody hits the nudge button multiple times in one second. The direction in which the pitch and yaw change for each nudge depends on the orientation. We include diagrams of the nudge direction along with example plots of nudge events. These are shown for 0/90 PARA in Fig. 13, for 0/90 PERP in Fig. 14, for 45/135 PARA in Fig. 15, and for 45/135 PERP in Fig. 16. We see that it takes approximately 15 seconds for the beam energy to change, and the energy change is on the order of 5 MeV for a small nudge.

To get a better idea of the energy change per nudge, we look at the change in the coherent edge position 10 seconds after the nudge occurred. A plot showing the energy change is given in Fig. 17. The statistics are relatively small, so it is difficult to draw any firm conclusions using single nudge events. Furthermore, this energy change calculation is very sensitive to noise and beam drops. In the next subsection, we look at how the energy change behaves as a function of the number of small nudges, which is a more robust way to estimate the energy change per nudge.

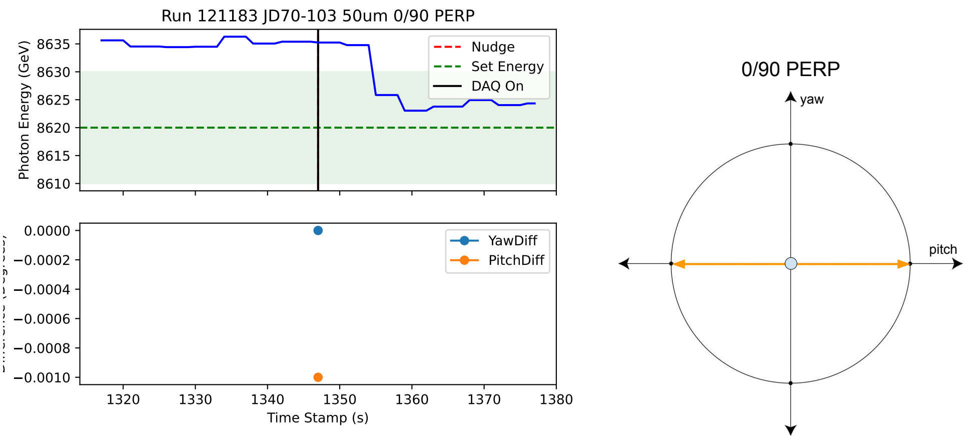

Figure 14: Left: example nudge for the 0/90 PERP configuration. Right: diagram showing nudges only occur in the ± pitch direction for 0/90 PERP.

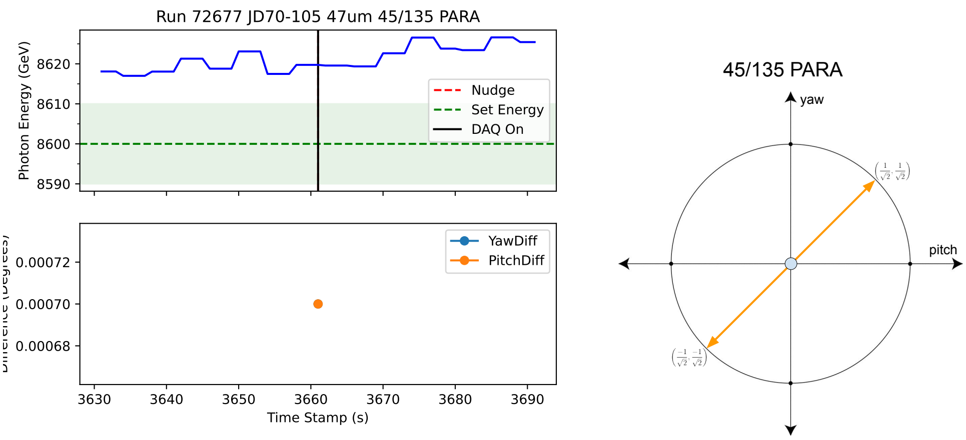

Figure 15: Left: example nudge for the 45/135 PARA configuration. Right: diagram showing nudges change yaw and pitch equally for 45/135 PARA.

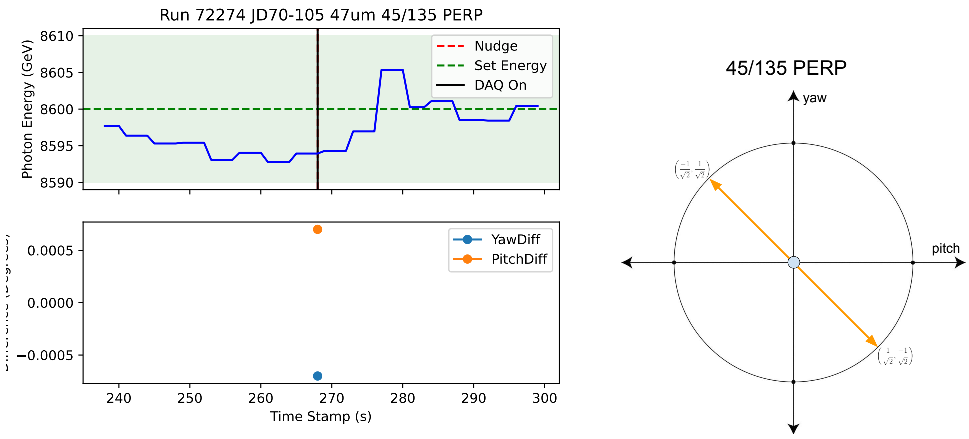

Figure 16: Left: example nudge for the 45/135 PERP configuration. Right: diagram showing nudges change pitch and yaw in opposite directions for 45/135 PERP.

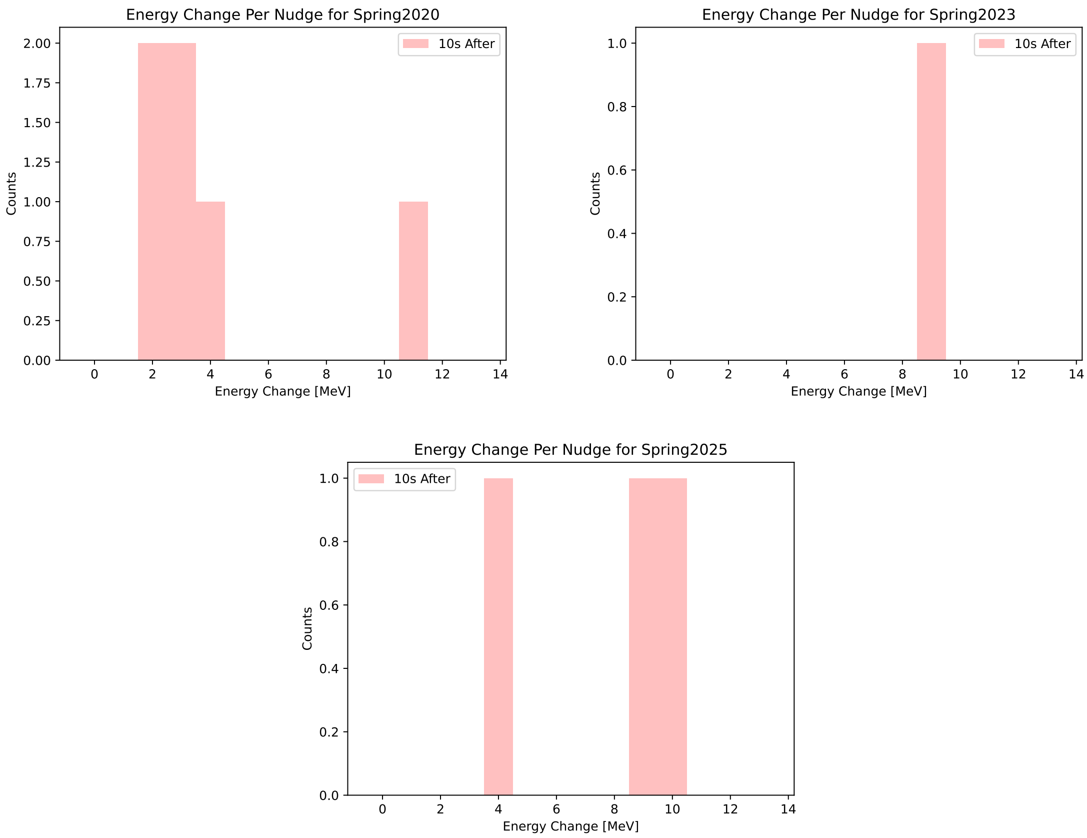

Figure 17: Histogram showing typical energy changes 10 seconds after a nudge occurred for Spring 2020 (top), Spring 2023 (middle), and Spring 2025 (bottom).

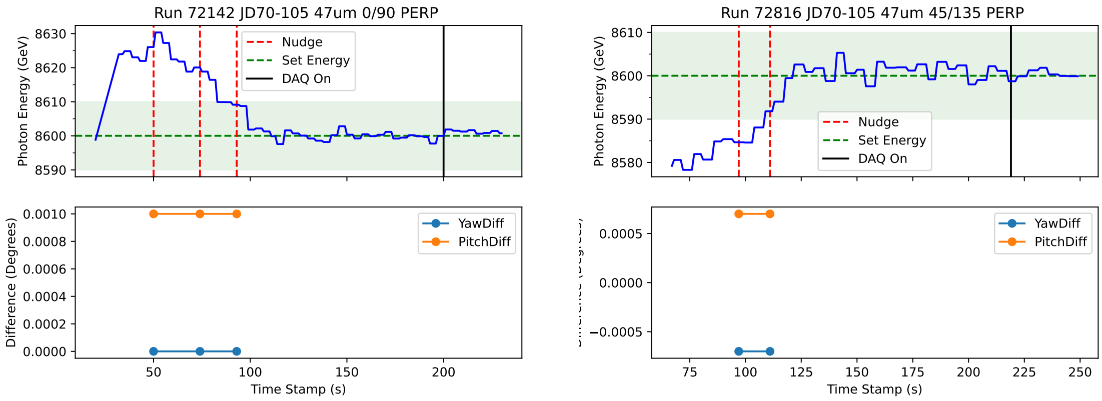

Figure 18: Clean examples of Spring 2020 runs with multiple nudges for the 0/90 (left) and 45/135 (right) orientations.

4.1 Runs With At Least One Nudge

The estimated energy change per nudge in Fig. 17 is not very reliable. To get a more robust estimate, we can plot the energy change versus the number of nudges for runs with at least one nudge and no backlash. To improve data quality, I also looked through the plots of all the runs and picked the ones where the beam energy was most stable, ie there were no beam drops in the time range of interest. Examples of clean runs with at least one nudge are shown in Fig. 18 for Spring 2020, Fig. 19 for Spring 2023, and Fig. 20 for Spring 2025.

The plots for Spring 2020 show the run in the 0/90 orientation has 3 nudges and an energy change of roughly 30 MeV, so we can estimate roughly 10 MeV as the energy change for a single nudge. In the Spring 2020 45/135 configuration, there are 2 nudges and an energy change of only 15 MeV. This gives an estimated energy change per nudge of 7.5 MeV for the 45/135 orientation.

To investigate this further, we measure the difference in the energy of the coherent edge before and after the nudge sequence is finished. The initial energy is taken as the energy when the first nudge occurred. To get a reliable estimate for the ending energy, we look at the time window between 15 and 30 seconds after the final nudge, and take the average energy in that time.

We look at the change in energy as a function of the number of nudges for all three run periods in Fig. 21. Based on our earlier observation that 0/90 and 45/135 appear to have different energy changes per nudge, we split this plot by orientation as well. We fit the corresponding scatter plots with a linear regression model, where we force the y-intercept to be zero, since 0 nudges must cause 0 change in energy. For the 0/90 orientation, we find a slopes between 9 and 11 MeV per nudge across the three run periods, consistent with the results above. For the 45/135 orientation, we get slopes ranging from 6 to 8.6 MeV per nudge, consistent with our estimate above. For the 45/135 orientation in Spring 2023, there is a point at (0,12) on the graph. This is caused by a weird bug where only the change in one direction was recorded, so the total nudge size is less than 1 millidegrees.

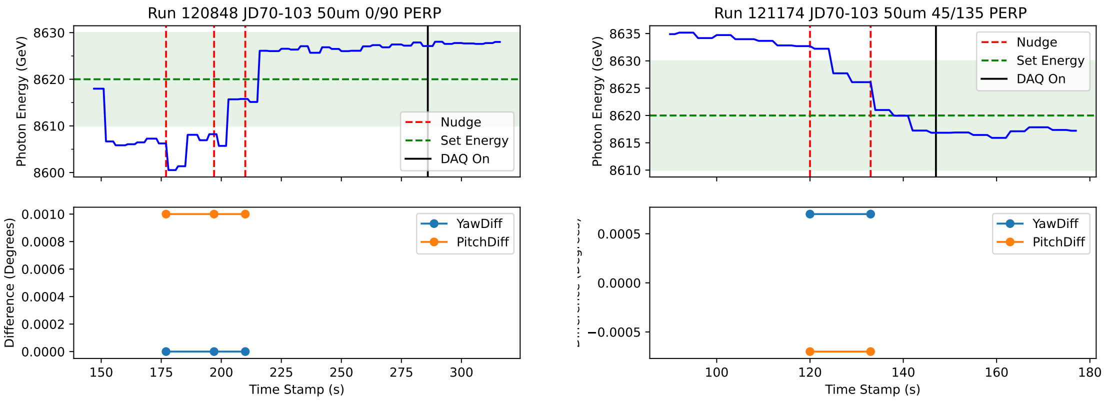

Figure 19: Clean examples of Spring 2023 runs with multiple nudges for the 0/90 (left) and 45/135 (right) orientations.

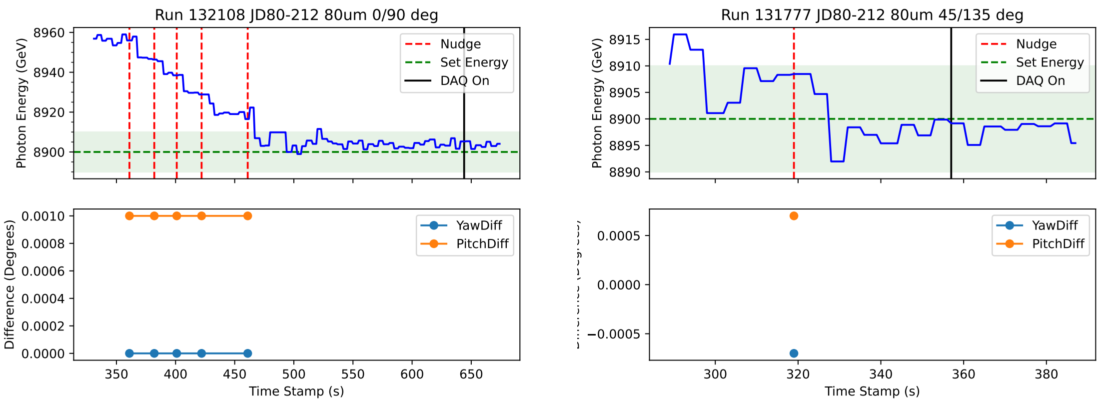

Figure 20: Clean examples of Spring 2025 runs with at least one nudge for the 0/90 (left) and 45/135 (right) orientations.

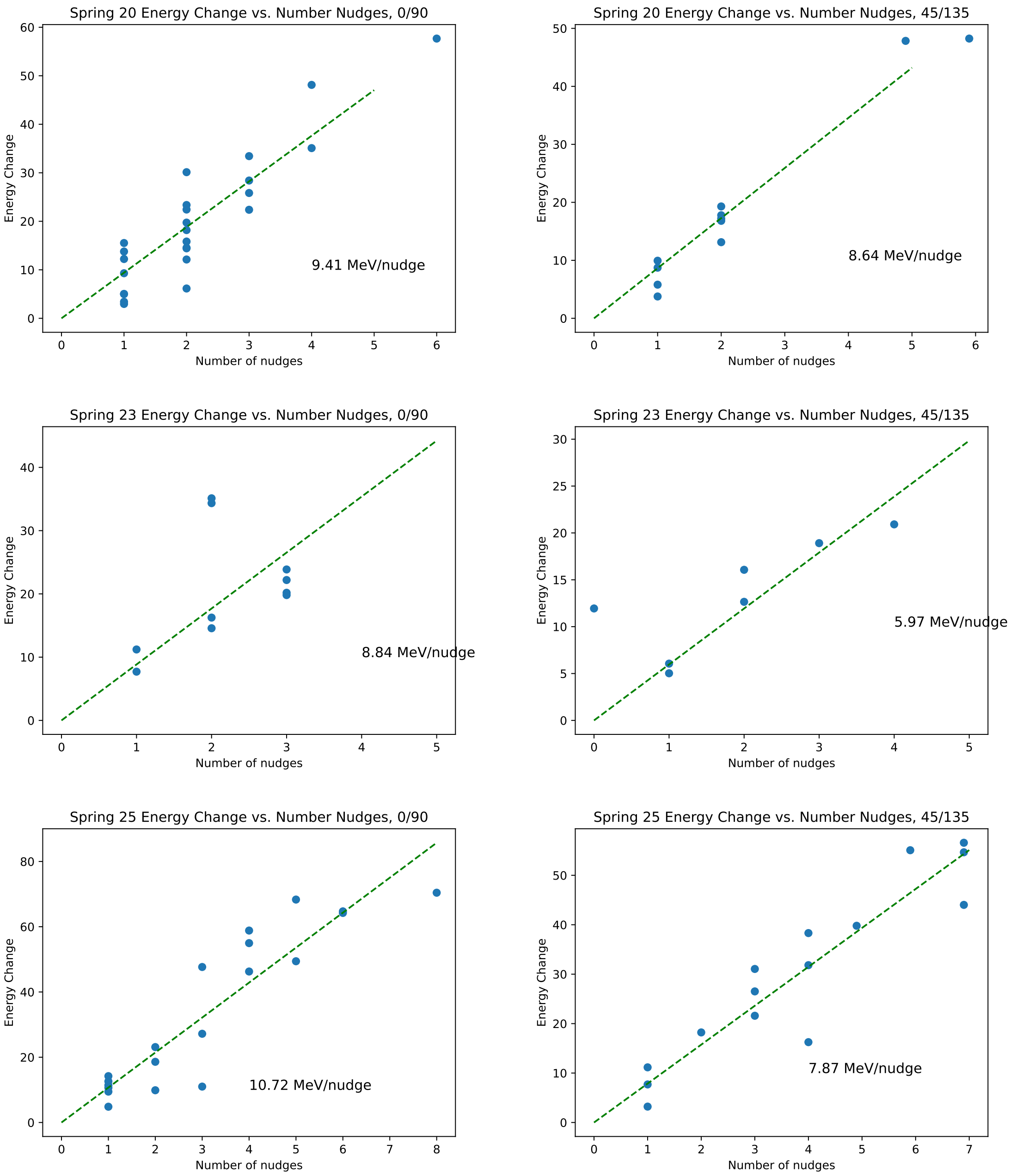

Figure 21: Measured energy changes as a function of the number of nudges in the run.

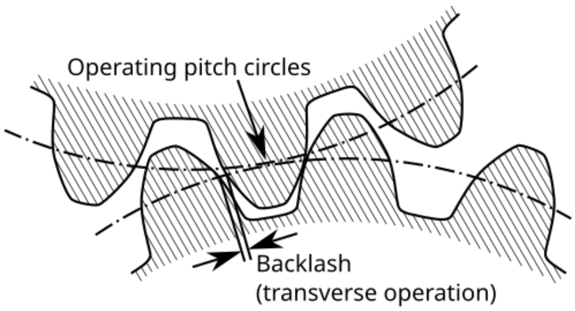

Figure 22: Diagram showing how backlash can appears in a system of gears.

5 Backlash

5.1 Background

5.1.1 Definition

Backlash is a common effect in mechanical systems. As an example, if two gear wheels are fit together, there is likely a small gap between the gears, in other words, they can "wiggle" a bit without putting appreciable force into the system. The amount of free movement allowed is the backlash of the system. The figure I stole off of Wikipedia [5] in Fig. 22 shows the backlash of a system of two gears.

5.1.2 Initial Goniometer Studies

Backlash can impact the goniometer when the motor direction changes, because some of the motion will be lost to backlash. Goniometer studies indicated that there is approximately 2.1 millidegrees of backlash in the angular stages of the goniometer [7]. This corresponds to 2 small nudges if only pitch or yaw are changing, or 3 small nudges if both pitch and yaw change. The expected effects of this are described in the next subsection (so they are easier to find from the table of contents).

5.2 Expected Effects of Backlash

If the motor motion reverses, there is around 2.1 millidegrees of backlash. This means for small nudges

- For the 0/90 orientation, only pitch or yaw changes. Therefore, it takes 2 small nudges to cancel out most of the backlash, and the 3rd small nudge will 10% less effective.
- For 45/135, both pitch and yaw change. There are 2.1 millidegrees of backlash in both directions, and step sizes are $(1/\sqrt{2} \approx 0.7)$ millidegrees in each direction. Therefore, it takes 3 small nudges to cancel out the effect of backlash.

Large nudges will change the coherent edge in the correct direction, but will have a reduced effect:

- For 0/90, the large nudge will be 21% less effective due to backlash.
- For 45/135, the large nudge will be about 30% less effective due to backlash.

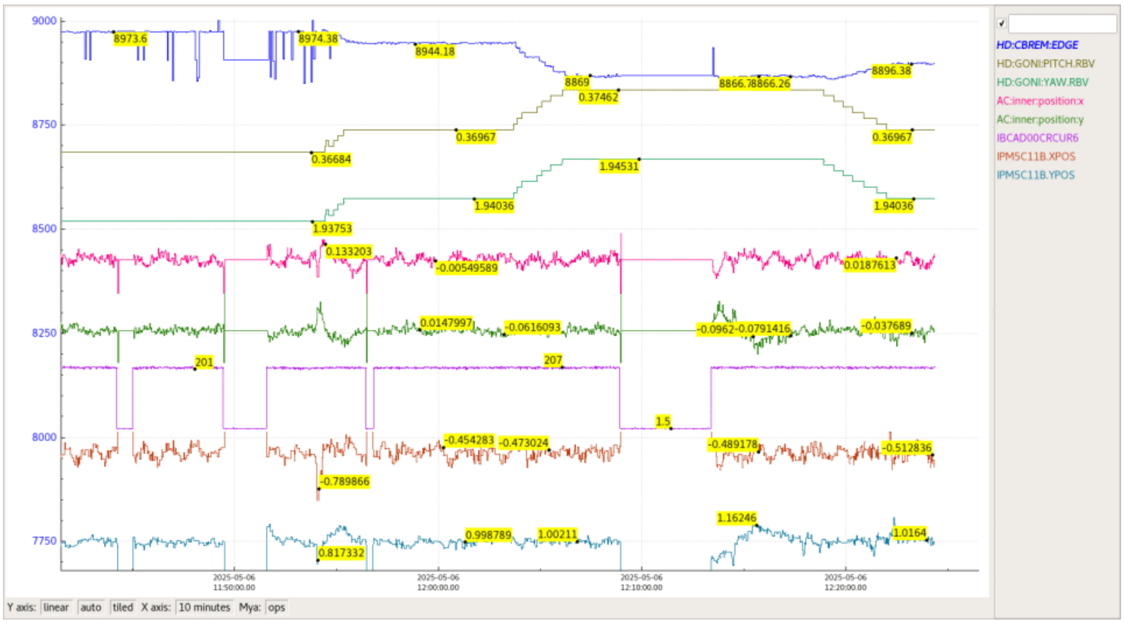

Figure 23: Plot of measured beam energy, pitch, yaw, etc.

5.3 Backlash Observed in 2025 Running

Hovanes noticed an example of backlash during the 2025 data taking, in the logbook entry cited here [6]. In the logbook entry, Hovanes estimates 12 MeV/millidegree as the change in the coherent edge given a nominal beam energy of 8.9 GeV. He assumes roughly 2 millidegrees of backlash, which gives an expected change in energy of around 25 MeV. His calculation is correct assuming only pitch or yaw is changing, but since this is the 45/135 configuration, both pitch and yaw are changing. In this case, the backlash effect is equivalent to three small nudges, so the effect should be closer to 36-40 MeV. The observed energy change was 48 MeV, so the scale is approximately correct, but the measured effect is still bigger than expected. A plot showing the measured beam energy, as well as pitch and yaw readback values is shown in Fig. 23. We see that the coherent edge energy is different for the same readback values in pitch and yaw, which is indicative of backlash in the system.

5.4 Backlash in Spring 2020 Nudges

In the good runs of Spring 2020 (after the energy fit script was fixed), we find several instances of backlash. We flag a run as having backlash if the goniometer motor reverses direction in either pitch or yaw. The plots in this section follow the same style as those in the previous section, except the backlash nudges are indicated by gray lines. The gray lines have an additional gray-shaded region covering the 15 seconds after the backlash nudge. For effective nudges, the coherent edge energy will change within this 15 second window, but for backlash nudges, we expect no change in energy during this time frame.

5.4.1 Clearest Backlash Example

In Fig. 24, we show the clearest examples of backlash events for the 0/90 orientation. In this example, we see 3 downward nudges followed by 3 upward nudges. We denote this nudge sequence as

$$ddduuu = 0, \tag{1}$$

where the 0 indicates the final pitch and yaw positions will be identical to the starting values. We can see the energy changes from around 8620 MeV to 8600 MeV, which is what we expect if there is backlash. Based on the studies above, we expect the first two nudges to be ineffective, so the "effective" nudge sequence can be written as

$$ddd\textcolor{red}{uu}u = dddu = dd \tag{2}$$

so it is equivalent to just two downward nudges. Estimating the energy change per nudge, we see a 20 MeV change in energy for two "effective" nudges, which gives an energy change per nudge of 10 MeV, consistent with the energy change per nudge measured earlier.

5.4.2 Sequential Backlash in Same Run

We look at a run that has multiple instances of backlash in Fig. 25. Note that the upward nudge is 3 small nudges in rapid succession. This gives us a nudge sequence of

$$ddduuudddd = dddd, \tag{3}$$

so we would expect a roughly 40 MeV change in energy if there was no backlash. Accounting for backlash, we get the "effective" nudge sequence of

$$ddd\textcolor{red}{uu}u\textcolor{red}{dd}dd = dddudd = dddd, \tag{4}$$

which is identical to what we would expect in the absence of backlash. In this case, the backlash effect is "cleared" in both directions, so the effect cancels out.

5.4.3 Additional Backlash Examples

We show an additional backlash plot for the 0/90 orientation in Fig. 26. We see that the first two downward nudges appear to be ineffective, while the third results in a change in the coherent peak energy. This is exactly the behavior we expect for the 0/90 configuration. The actual nudge sequence of

$$dduuu = u \tag{5}$$

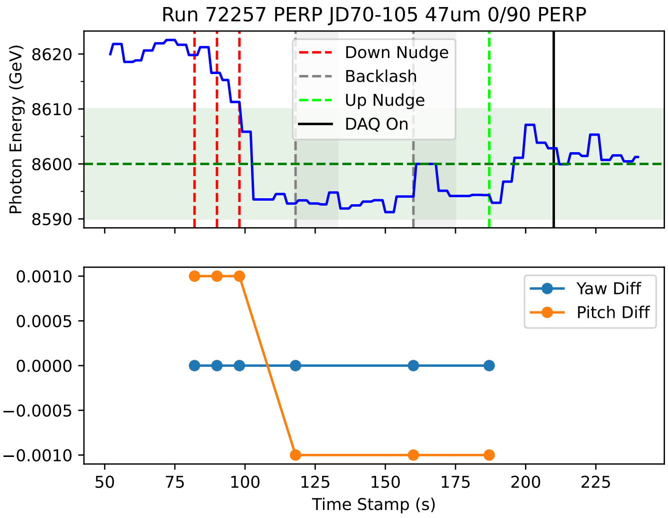

Figure 24: The clearest example of backlash in Spring 2020. Three nudges down followed by three nudges up should result in no energy change, but the energy clearly changes. The dashed red lines indicate downward nudges, the green line is an upward nudge, and the gray lines indicate backlash nudges. The 15 seconds after a backlash nudge are shaded in gray, indicating that no coherent edge energy change should occur in this time frame.

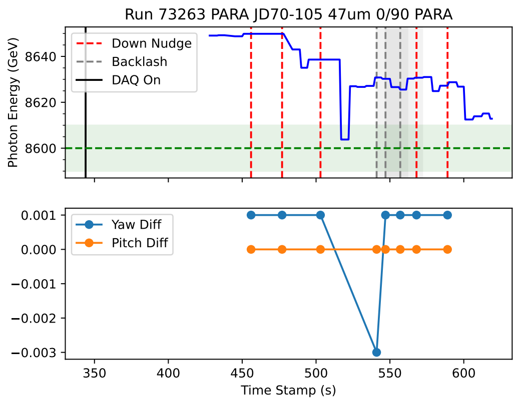

Figure 25: An example where backlash occurs in both directions, effectively canceling out the backlash effect.

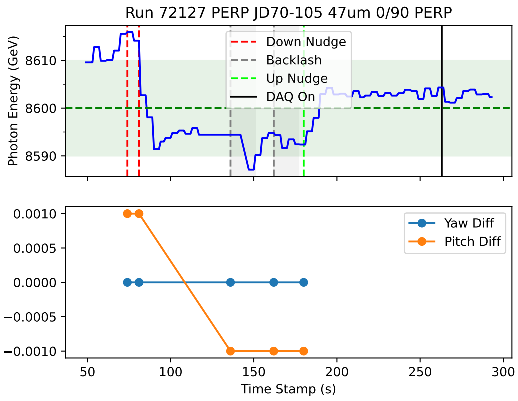

Figure 26: Cleanest additional example of backlash for the 0/90 orientation in the Spring 2020 data.

gives an expected energy change of 10 MeV up. Accounting for the backlash, the ”effective” nudge sequence is

$$dd\textcolor{red}{uu}u = d \tag{6}$$

which is a 10 MeV downward shift. A 10 MeV downard shift is observed in this case.

We show two examples of backlash events for the 45/135 orientation in Fig. 27. The plot on the left shows an unstable coherent edge position, regardless of if a nudge occurred, so it is harder to interpret than the cases above. We see the final three nudges only have a small change on the coherent peak energy, but this run likely has unstable x or y parameters, which make backlash harder to observe and interpret.

The second plot in Fig. 27 shows much more stable coherent edge energies. We see that the two nudges that occur are backlash nudges, and they appear to have no effect on the coherent edge energy, as we would expect in this case.

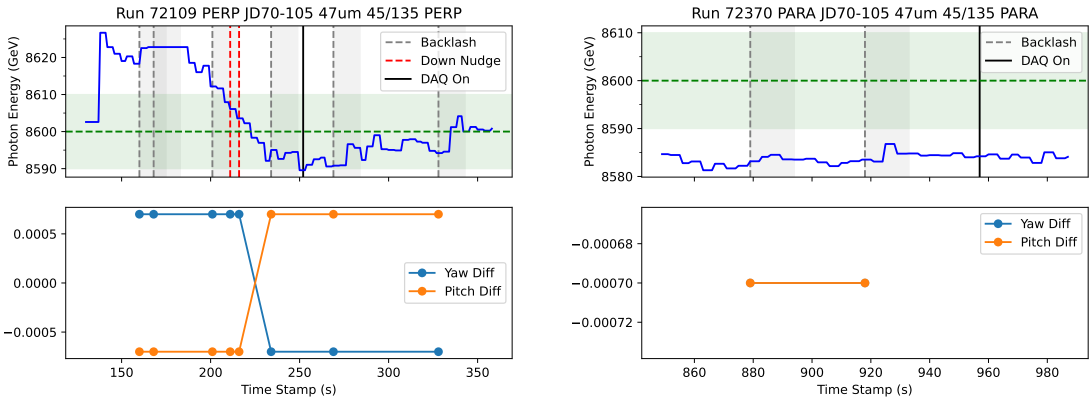

Figure 27: The cleanest examples of backlash for the 45/135 orientation. The bottom right plot isn't clean, but has an interesting change in beam energy that occurs mid-run.

References

[1] AIOP Project Proposal.

[2] mySampler User's Guide. https://halldweb.jlab.org/wiki/images/2/23/MySampler.user.pdf

[3] myget User's Guide https://halldweb.jlab.org/wiki/images/e/e3/Myget.pdf

[4] EPICS Variables and their definitions

`https://docs.google.com/spreadsheets/d/1C6lJ3S_5pj_ariDfMVNg5n1iyFbjDV4QYBcpYHo6-90/edit?gid=0#gid=0`

[5] Wikipedia https://en.wikipedia.org/wiki/Backlash_(engineering)

[6] Logbook entry of backlash https://logbooks.jlab.org/entry/4370351

[7] Yi's Study of Backlash https://halldweb.jlab.org/wiki/images/5/56/Goniometer_tests-11-2013.pdf

A To Do, Random Facts

A.1 To Do

- Investigate why some nudges are more than 1 degree. Likely due to changes in orientation.
- Consider dropping initial rows of NaN's instead of doing backfill during the merge process.

A.2 Random Facts

By comparing the measured roll values to the notes in RCDB, I was able to map the set roll values to the 0/90, 45/135 orientations. I found that for Spring 2020

- Roll of $-10.5^\circ \rightarrow 0/90$
- Roll of $34.5^\circ \rightarrow 45/135$
- CBREM:PLANE of 1 is PARA
- CBREM:PLANE of 2 is PERP
- CBREM:PLANE of 0 indicates switching configs

B myGet Commands For Each Run Period

B.1 Spring 2020

```
myget -m history -c 'HD:coda:daq:run_number' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/run_number.txt
myget -m history -c 'HD:CBREM:PLANE' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/cbrem_plane.txt
myget -m history -c 'HD:coda:daq:status' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/daq_status.txt
myget -m history -c 'HD:GONI:X' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_x.txt
myget -m history -c 'HD:GONI:X.RBV' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_x_rbv.txt
myget -m history -c 'HD:GONI:Y' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_y.txt
myget -m history -c 'HD:GONI:Y.RBV' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_y_rbv.txt
myget -m history -c 'HD:GONI:PITCH' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_pitch.txt
myget -m history -c 'HD:GONI:PITCH.RBV' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_pitch_rbv.txt
myget -m history -c 'HD:GONI:YAW' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_yaw.txt
myget -m history -c 'HD:GONI:YAW.RBV' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_yaw_rbv.txt
myget -m history -c 'HD:GONI:ROLL' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_roll.txt
myget -m history -c 'HD:GONI:ROLL.RBV' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/goni_roll_rbv.txt
myget -m history -c 'HallDACOrb:Suspended' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/beam_lock.txt
myget -m history -c 'HALLD:p' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/ebeam_energy.txt
myget -m history -c 'HD:CBREM:EDGE' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/photonbeam_energy.txt
myget -m history -c 'HD:CBREM:REQ_EDGE' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/photonbeam_energy_desired.txt
myget -m history -c 'HD:CBREM:WIDTH' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/photonbeam_energy_uncertainty.txt
myget -m history -c 'AC:inner:position:x' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/ac_x.txt
myget -m history -c 'AC:inner:position:y' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/ac_y.txt
myget -m history -c 'IPM5C11B.XPOS' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/ebeam_x.txt
myget -m history -c 'IPM5C11B.YPOS' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/ebeam_y.txt
myget -m history -c 'IBCAD00CRCUR6' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/ebeam_current.txt
myget -m history -c 'HD:GONI:RADIATOR_NAME' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/radiator_name.txt
myget -m history -c 'HD:GONI:RADIATOR_ID' -b '2020-01-10 00:00:00' -e '2020-09-21 12:00:00' | tee Spring20/radiator_id.txt
```

B.2 Spring 2023

```
myget -m history -c 'HD:coda:daq:run_number' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/run_number.txt
myget -m history -c 'HD:CBREM:PLANE' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/cbrem_plane.txt
myget -m history -c 'HD:coda:daq:status' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/daq_status.txt
myget -m history -c 'HD:GONI:X' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_x.txt
myget -m history -c 'HD:GONI:X.RBV' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_x_rbv.txt
myget -m history -c 'HD:GONI:Y' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_y.txt
myget -m history -c 'HD:GONI:Y.RBV' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_y_rbv.txt
myget -m history -c 'HD:GONI:PITCH' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_pitch.txt
myget -m history -c 'HD:GONI:PITCH.RBV' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_pitch_rbv.txt
myget -m history -c 'HD:GONI:YAW' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_yaw.txt
myget -m history -c 'HD:GONI:YAW.RBV' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_yaw_rbv.txt
myget -m history -c 'HD:GONI:ROLL' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_roll.txt
myget -m history -c 'HD:GONI:ROLL.RBV' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/goni_roll_rbv.txt
myget -m history -c 'HallDACOrb:Suspended' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/beam_lock.txt
myget -m history -c 'HALLD:p' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/ebeam_energy.txt
myget -m history -c 'HD:CBREM:EDGE' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/photonbeam_energy.txt
myget -m history -c 'HD:CBREM:REQ_EDGE' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/photonbeam_energy_desired.txt
myget -m history -c 'HD:CBREM:WIDTH' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/photonbeam_energy_uncertainty.txt
myget -m history -c 'AC:inner:position:x' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/ac_x.txt
myget -m history -c 'AC:inner:position:y' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/ac_y.txt
myget -m history -c 'IPM5C11B.XPOS' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/ebeam_x.txt
myget -m history -c 'IPM5C11B.YPOS' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/ebeam_y.txt
myget -m history -c 'IBCAD00CRCUR6' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/ebeam_current.txt
myget -m history -c 'HD:GONI:RADIATOR_NAME' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/radiator_name.txt
myget -m history -c 'HD:GONI:RADIATOR_ID' -b '2023-01-21 00:00:00' -e '2023-03-19 22:00:00' | tee Spring23/radiator_id.txt
```

B.3 Spring 2025

```
myget -c 'HD:coda:daq:run_number' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/run_number.txt
myget -c 'HD:CBREM:PLANE' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/cbrem_plane.txt
myget -c 'HD:coda:daq:status' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/daq_status.txt
myget -c 'HD:GONI:X' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/goni_x.txt
myget -c 'HD:GONI:X.RBV' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/goni_x_rbv.txt
myget -c 'HD:GONI:Y' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/goni_y.txt
myget -c 'HD:GONI:Y.RBV' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/goni_y_rbv.txt
myget -c 'HD:GONI:PITCH' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/goni_pitch.txt
myget -c 'HD:GONI:PITCH.RBV' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/goni_pitch_rbv.txt
```

```
myget -c 'HD:GONI:YAW' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/goni_yaw.txt
myget -c 'HD:GONI:YAW.RBV' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00'| tee Spring25/goni_yaw_rbv.txt
myget -c 'HD:GONI:ROLL' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/goni_roll.txt
myget -c 'HD:GONI:ROLL.RBV' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/goni_roll_rbv.txt
myget -c 'HallDACOrb:Suspended' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/beam_lock.txt
myget -c 'HALLD:p' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/ebeam_energy.txt
myget -c 'HD:CBREM:EDGE' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/photonbeam_energy.txt
myget -c 'HD:CBREM:REQ_EDGE' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/photonbeam_energy_desired.txt
myget -c 'HD:CBREM:WIDTH' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/photonbeam_energy_uncertainty.txt
myget -c 'AC:inner:position:x' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/ac_x.txt
myget -c 'AC:inner:position:y' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/ac_y.txt
myget -c 'IPM5C11B.XPOS' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/ebeam_x.txt
myget -c 'IPM5C11B.YPOS' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/ebeam_y.txt
myget -c 'IBCAD00CRCUR6' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/ebeam_current.txt
myget -c 'HD:GONI:RADIATOR_NAME' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/radiator_name.txt
myget -c 'HD:GONI:RADIATOR_ID' -b '2025-05-03 20:00:00' -e '2025-06-15 23:59:00' | tee Spring25/radiator_id.txt
```

C Logging Into Gluons

Logging into the gluons is relatively simple. First, login to login.jlab.org as normal. Then, instead of logging into ifarm, you can ssh into one of the gluons. gluon150-155 are for general use.