

Technical Write-up

Overview

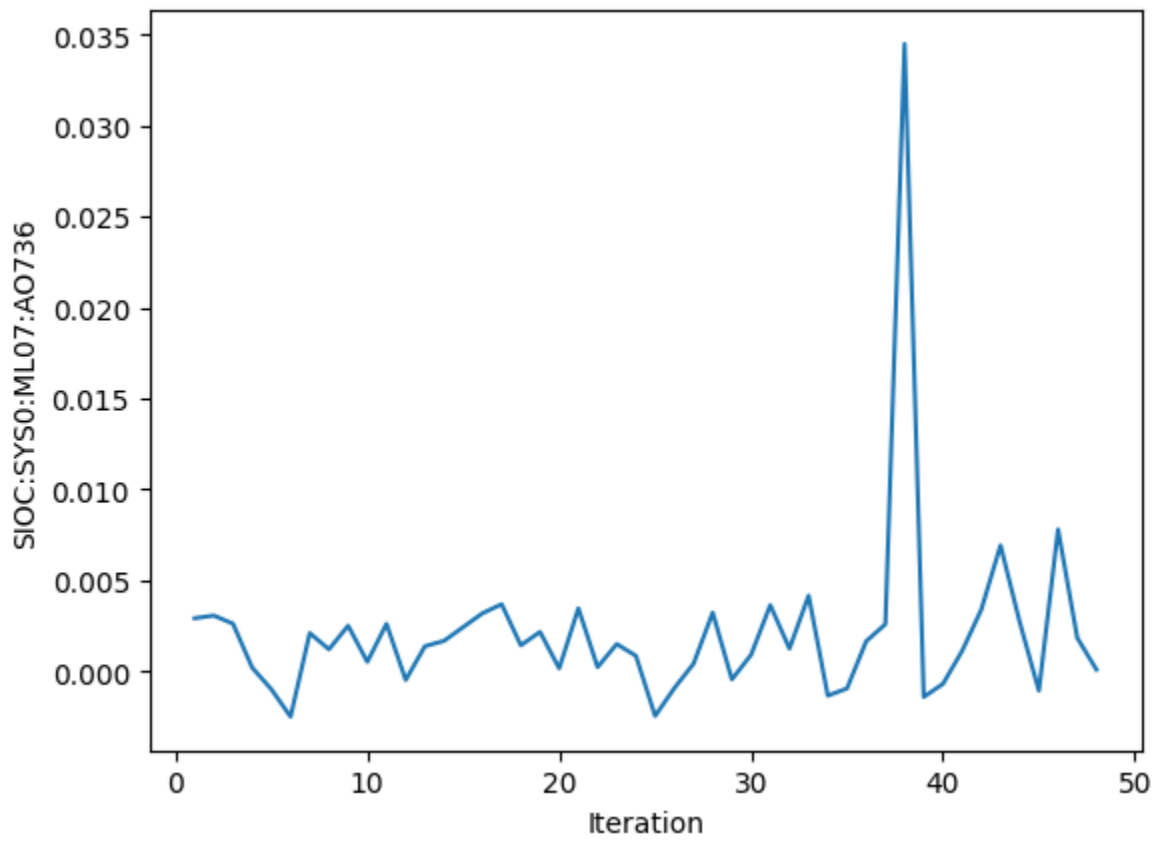
In this internship, the basic goals for me were to get set up and familiarized with the software that was used, to access and use the two-bunch data in the LCLS-archive, and to run a simple Gaussian process on the data to optimize the gas detector output. Some areas of this took longer than others and there were definitely issues getting things to work on my end. Many of these issues were from my unfamiliarity with the processes, but some of it was also the way things were recorded or other errors that are too difficult for me to resolve, which I will elaborate on in the Recommendations section.

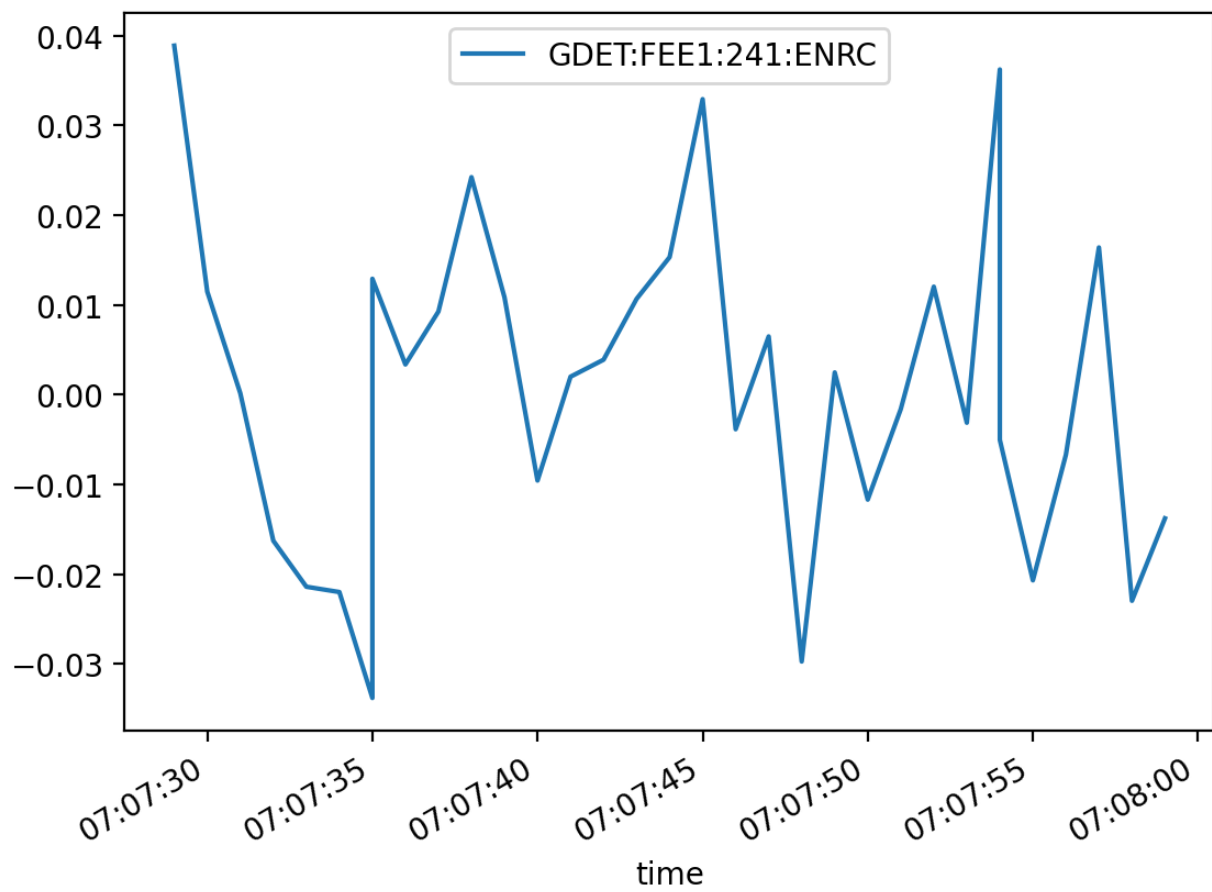
Details

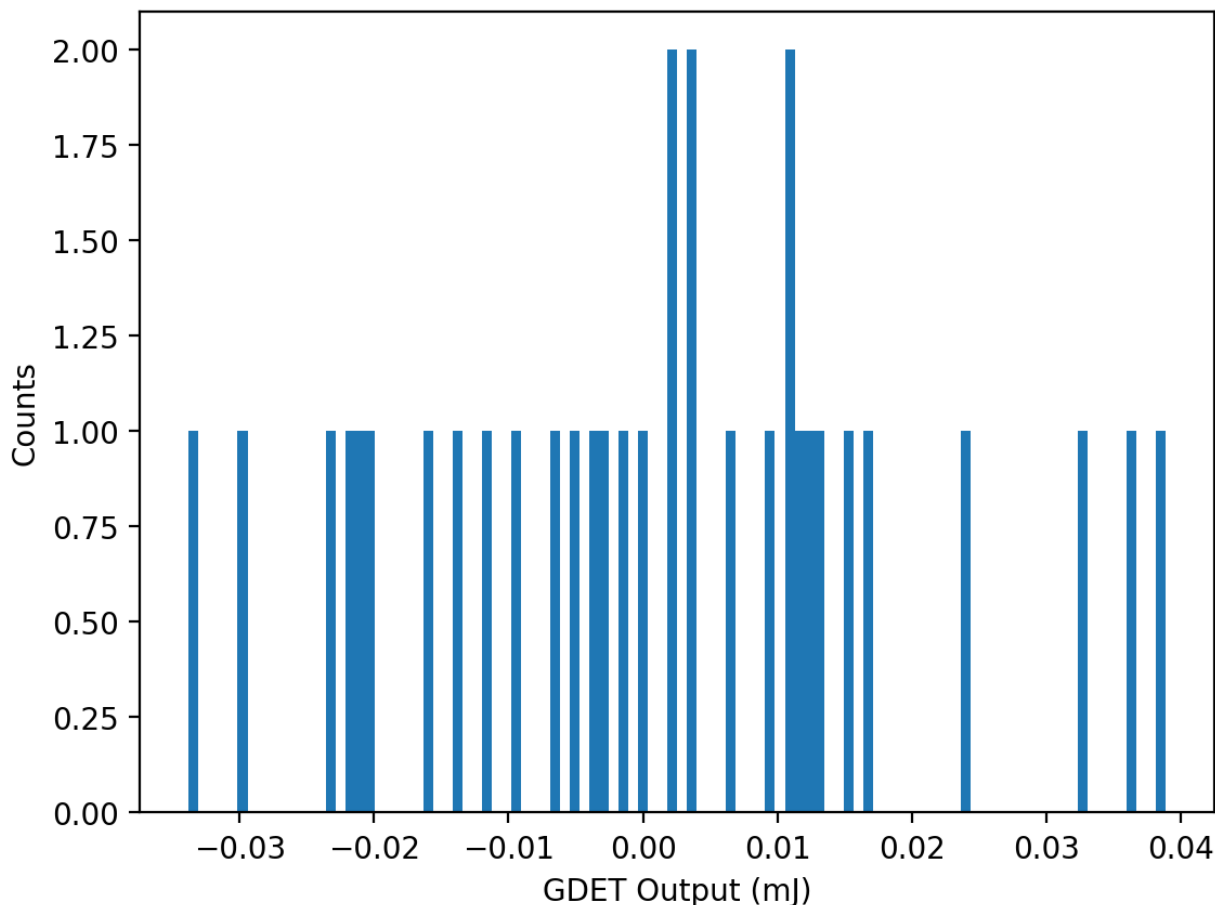
In this internship, I used a variety of software and sites. I had to work with SLAC HR and IT departments to get a SLAC email and calendar up, as well as unix. I had to learn how to SSH into areas, create keys, and use SDF. This involved learning how to use the computer terminals, like Command Prompt, Pycharm, and Anaconda. I had to learn how to use directories, environments, packages and executables, GitHub, and JupyterLab and Notebook later on. The majority of my work was done using the LCLS-Archive notebooks off of GitHub and other example notebooks, and later I merged some of the code with simple regression Gaussian Process tutorials. I was also given several papers that I read to provide a deeper understanding of the concepts applied in the research I was looking at.

My work consisted of three primary tasks, all of which involved learning the necessary tools, troubleshooting, and communicating with my internship supervisors or others. My first task was to use the LCLS-Archive template, a range of dates and times, a folder of yaml files, and a list of PVs given to me to plot the settings for the quadrupoles, multi knobs, delay settings, the individual gas detectors and their combined output both in relation to iteration and to time,







plus histograms.



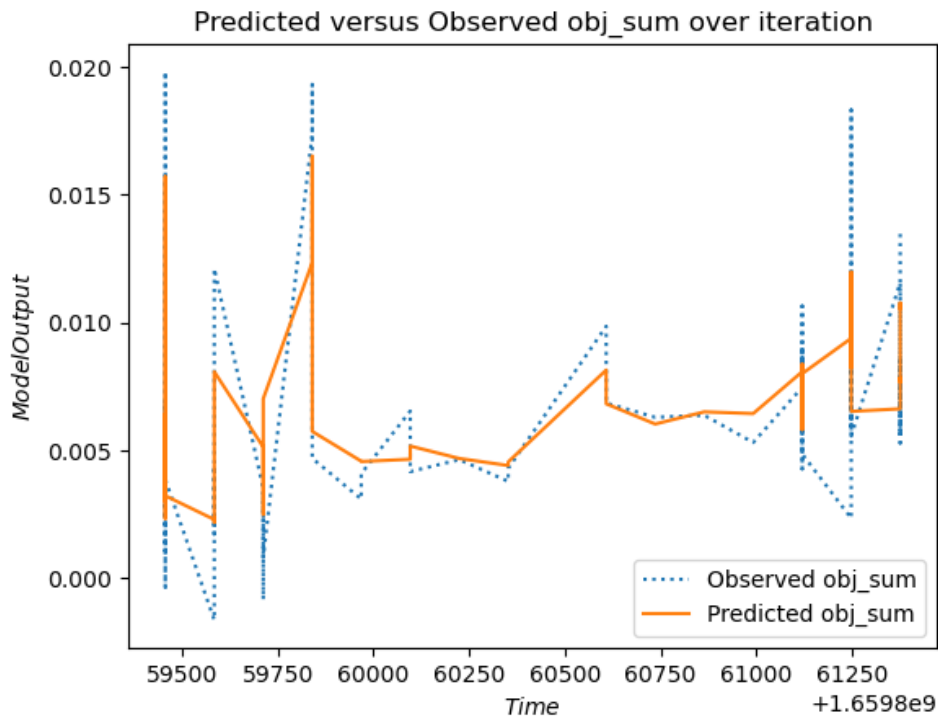




My second task was to use an example plotting notebook plus my earlier work to create documents that recorded relevant information about the PVs in each yaml file in the folder, and to plot the results. This included the keys, whether individual quads were adjusted, whether multi knobs were adjusted, whether the file contained data for the individual gas detectors or combined gas detector output, the variables, objectives, and constraints.

YAML FILE	KEYS	LI26, LTU Quad	Multiknobs Adj	SIOC.SYS0:MLC Variables	Objectives	Constraints	GDET	GDET1	GDET2
20220806_two	Index(['MKB:SYS0:3:V', 'MKB:SYS0:4:V', 'TDLY:LI21:1:A', 'TDLY:LI21:1:C'], dtype='object')	No	Yes	No	{'obj_sum': 'MAX'}	{'GDET_constraint_check': ['GREATER_THAN', 0.001], 'SIOC.SYS0:ML07:AO736_constraint': ['GREATER_THAN', 0.001], 'SIOC.SYS0:ML07:AO737_constraint': ['GREATER_THAN', 0.001]}			
20220806_two	Index(['ACCL:IN20:300:L0A_WF', 'ACCL:IN20:400:L0B_WF', 'ACCL:LI21:180:L1X_WF', 'ACCL:LI21:1:L1S_WF', 'ACCL:LI24:100:KLY_WF', 'ACCL:LI24:200:KLY_WF', 'ACCL:LI24:300:KLY_WF', 'DIAG:FEE1:202:241:DATA', 'GDET', 'GDET:FEE1:241:ENRC', 'GDET_constraint_check', 'GUN:IN20:1:GUN_WF', 'MKB:SYS0:3:VAL', 'MKB:SYS0:4:VAL', 'SIOC.SYS0:ML07:AO736', 'SIOC.SYS0:ML07:AO736_constraint', 'SIOC.SYS0:ML07:AO737', 'SIOC.SYS0:ML07:AO737_constraint', 'TCAV:DMPH:360:TCA_WF', 'TDLY:LI21:1:ADelaySet', 'TDLY:LI21:1:CDelaySet', 'TMITH', 'obj_sum', 'obj_sum_constraint', 'time', 'xopt_error', 'xopt_error_str'], dtype='object')				{'obj_sum': 'MAX'}	{'GDET_constraint_check': ['GREATER_THAN', 0.001], 'SIOC.SYS0:ML07:AO736_constraint': ['GREATER_THAN', 0.001], 'SIOC.SYS0:ML07:AO737_constraint': ['GREATER_THAN', 0.001]}			
20220806_two	Index(['ACCL:IN20:300:L0A_WF', 'ACCL:IN20:400:L0B_WF', 'ACCL:LI21:180:L1X_WF', 'ACCL:LI21:1:L1S_WF', 'ACCL:LI24:100:KLY_WF', 'ACCL:LI24:200:KLY_WF', 'ACCL:LI24:300:KLY_WF', 'DIAG:FEE1:202:241:DATA', 'GDET', 'GDET:FEE1:241:ENRC', 'GDET_constraint_check', 'GUN:IN20:1:GUN_WF', 'MKB:SYS0:3:VAL', 'MKB:SYS0:4:VAL', 'SIOC.SYS0:ML07:AO736', 'SIOC.SYS0:ML07:AO736_constraint', 'SIOC.SYS0:ML07:AO737', 'SIOC.SYS0:ML07:AO737_constraint', 'TCAV:DMPH:360:TCA_WF', 'TDLY:LI21:1:ADelaySet', 'TDLY:LI21:1:CDelaySet', 'TMITH', 'obj_sum', 'obj_sum_constraint', 'time', 'xopt_error', 'xopt_error_str'], dtype='object')	No	Yes	Yes	{'obj_sum': 'MAX'}	{'GDET_constraint_check': ['GREATER_THAN', 0.001], 'SIOC.SYS0:ML07:AO736_constraint': ['GREATER_THAN', 0.001], 'SIOC.SYS0:ML07:AO737_constraint': ['GREATER_THAN', 0.001]}			

My final task was to run a simple regression Gaussian process on all of the data that I had dealt with to get a conceptual understanding of how we optimize for our combined gas detector output despite the number of data points and input variables. I finished the internship with a write-up document and powerpoint slides. The packages I had to use in my studies were `lcls_live.archiver`, `json`, `os`, `yaml`, `pandas`, `matplotlib`, `math`, `torch`, `botorch`, `gpytorch`, `tensorflow`, `numpy`, `mamba`, `jupyter`, `jupyter_core`, `jupyter_client`, `jupyter-client`, `jupyter-console`, `jupyter-notebook`, `jupyter-qtconsole`, `jupyter-nbconvert`, `jupyter-trust`, `traitlets`, and `ipykernel`.



Results and Discussion

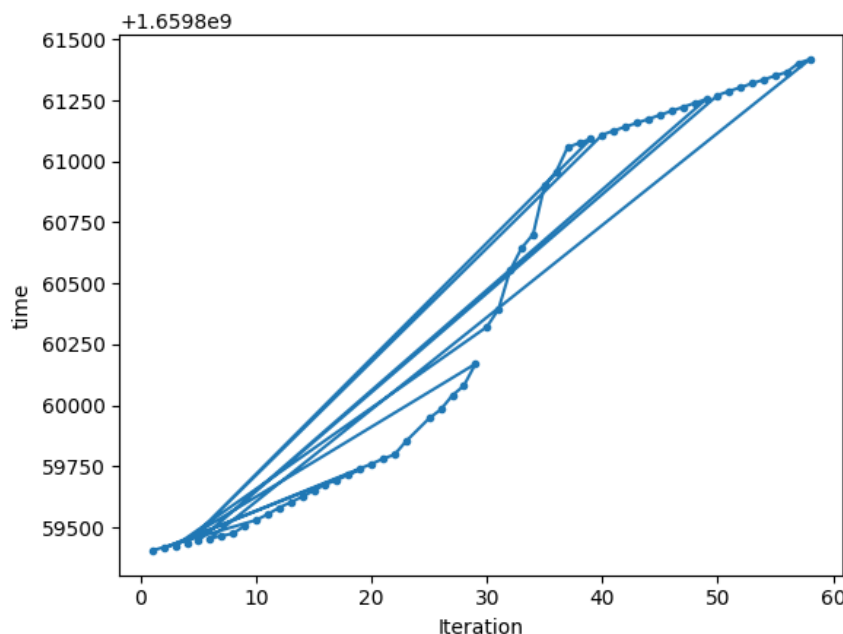
In the first task, we noticed that over the course of the run, there was a short period where it looked like some of the PVs had discontinuous changes to them. The conclusion I came to was that this was the period of the time where SLAC was making changes to the settings, like the quads, multi knobs, and delay settings. In order to analyze these specific areas, you can enter the section of the code where dates and times are inputted, and just change them to where the interval is. I also found out that the “Optional” section of the archiver template notebook just doesn’t function properly and won’t allow the notebook to run unless you delete it.

Optional: off-site setup

```
In [3]: # Optional:







# Open an SSH tunnel in a terminal like:
# ssh -D 8080 @
# And then set:
os.environ['http_proxy']='socks5h://localhost:8080'
os.environ['HTTPS_PROXY']='socks5h://localhost:8080'
os.environ['ALL_PROXY']='socks5h://localhost:8080'
```

I plotted the data both over time and over iteration, as well as with histograms. Another thing that was discovered when I did this was that the iterations didn't necessarily align perfectly with the time, some areas had iterations that were closer than others. One of my tasks was to see how each of the yaml files had the time compared to the iteration, but we ended up scrapping this task because the archive saved it improperly, and you have the same iteration versus time plot for every single yaml file.



In the second task, I found that many things were not uniform throughout the yaml files. Some files were grouped together and shared similarities, and others were completely different from the rest. The first yaml file and the last two were all completely independent of the rest, having different keys and graphs from every other file, while the rest of the files were organized into pairs that shared keys, vocs, whether individual quads or multi knobs were shifted, and whether the gas detector PVs were present, as well as many similarities for graphs of the PVs. Interestingly, of the 11 yaml files, only the second and third ones contained information about the

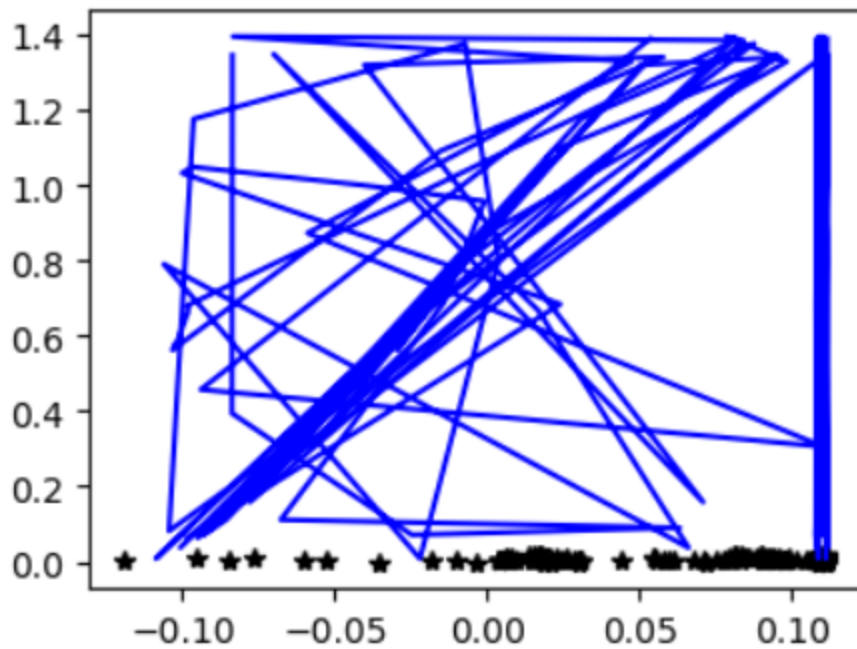
individual gas detector outputs under the original PV names. However, many of the yaml files contained new PV names GDET, GDETC, and GDET:FEE1:241:ENRC. It was the same situation for the combined outputs, not all of the yaml files contained GDETC, some contained obj_sum instead. I organized it according to color code.

YAML FILE	KEYS	LI26, LTU Qua	Multiknobs Adj	SIOC:SYS0:MLC	Variables	Objectives	Constraints	GDET	GDET1	GDET2
20220806_two	Index(['MKB.SYS0:TDLY:LI21:1:A', 'dtype='objec	No	Yes	No	{'MKB.SYS0:3.V', 'MKB.SYS0:4.V', 'TDLY:LI21:1.A', 'TDLY:LI21:1.C	{'obj_sum': 'MAX	{'GDET_constraint_check': ['GREATER_THAN', 0.001], 'SIOC:SYS0:ML07-AO736_constraint': ['GREATER_THAN', 0.001], 'SIOC:SYS0:ML07-AO737_constraint': ['GREATER_THAN', 0.001]}			
20220806_two	Index(['ACCL:IN', 'ACCL:LI21:1.A', 'ACCL:LI24:1.A', 'GDET.FEE1:241:ENRC', 'MKB.SYS0:SIOC:SYS0:TDLY:LI21:1.A', 'obj_sum_co', 'dtype='objec	No	Yes	Yes	{'MKB.SYS0:3.V', 'MKB.SYS0:4.V', 'TDLY:LI21:1.A', 'TDLY:LI21:1.C	{'obj_sum': 'MAX	{'GDET_constrai			
20220806_two	Index(['ACCL:IN', 'ACCL:LI21:1.A', 'ACCL:LI24:1.A', 'GDET.FEE1:241:ENRC', 'MKB.SYS0:SIOC:SYS0:TDLY:LI21:1.A', 'obj_sum_co', 'dtype='objec	No	Yes	Yes	{'MKB.SYS0:3.V', 'MKB.SYS0:4.V', 'TDLY:LI21:1.A', 'TDLY:LI21:1.C	{'obj_sum': 'MAX	{}			
20220805_two	Index(['DUMMY', 'QUAD:LI26:20:1.A', 'QUAD:LI26:30:1.A', 'QUAD:LI26:40:1.A', 'QUAD:LI26:50:1.A', 'QUAD:LI26:60:1.A', 'QUAD:LI26:70:1.A', 'QUAD:LI26:80:1.A', 'QUAD:LI26:90:1.A', 'TOTAL_HA', 'dtype='objec	Yes	No	No	{'QUAD:LI26:20:1.A', 'QUAD:LI26:30:1.A', 'QUAD:LI26:40:1.A', 'QUAD:LI26:50:1.A', 'QUAD:LI26:60:1.A', 'QUAD:LI26:70:1.A', 'QUAD:LI26:80:1.A', 'QUAD:LI26:90:1.A', 'TDLY:LI21:1.A	{'GDET': 'MAXIN	{}			
	Index(['GDET', 'QUAD:LI26:20:1.A', 'QUAD:LI26:30:1.A', 'QUAD:LI26:40:1.A', 'QUAD:LI26:50:1.A', 'QUAD:LI26:60:1.A', 'QUAD:LI26:70:1.A', 'QUAD:LI26:80:1.A', 'QUAD:LI26:90:1.A', 'dtype='objec				{'QUAD:LI26:20:1.A', 'QUAD:LI26:30:1.A', 'QUAD:LI26:40:1.A', 'QUAD:LI26:50:1.A', 'QUAD:LI26:60:1.A', 'QUAD:LI26:70:1.A', 'QUAD:LI26:80:1.A', 'QUAD:LI26:90:1.A', 'TDLY:LI21:1.A	{'GDET': 'MAXIN	{}			

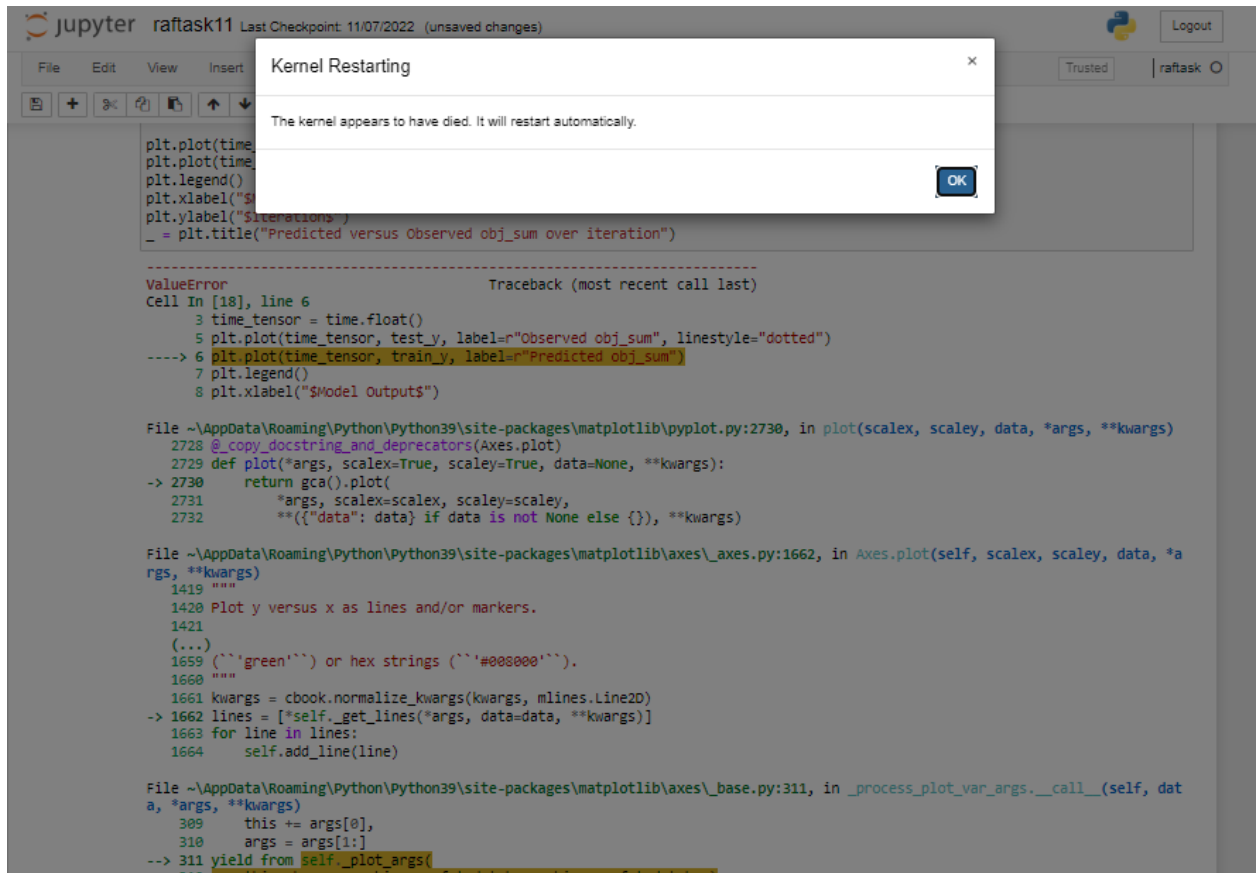
In the third task, I found that the tutorial notebooks I was given were not fully compatible with my directions. The tutorials weren't assigned to actual data, but rather to typed functions. In addition, they both contained a single input and a single output, while my job was to run the multi-knobs, quads, and delay settings as the x-input, with the obj_sum or GDETC as the y-variable.

```
In [2]: # Training data is 100 points in [0,1] inclusive regularly spaced
train_x = torch.linspace(0, 1, 100)
# True function is sin(2*pi*x) with Gaussian noise
train_y = torch.sin(train_x * (2 * math.pi)) + torch.randn(train_x.size()) * math.sqrt(0.04)
```

This caused two problems. The first was that I found I needed a new way to describe my results, as an n-dimensional input doesn't allow for a graph that makes sense if n isn't equal to one. I circumvented this problem by instead graphing several different data points or "slices" where I would keep all of the inputs except one constant, and then vary that one against the obj_sum. I did this for each input in turn.



The second problem with this was that the memory required to produce this run was extremely large, and would kill the kernel every time it was run. Unfortunately, I didn't find a way around this, even though I tried uninstalling and reinstalling every package with conda and pip, trying both locally and through SDF, and using a dll file that a colleague sent me.

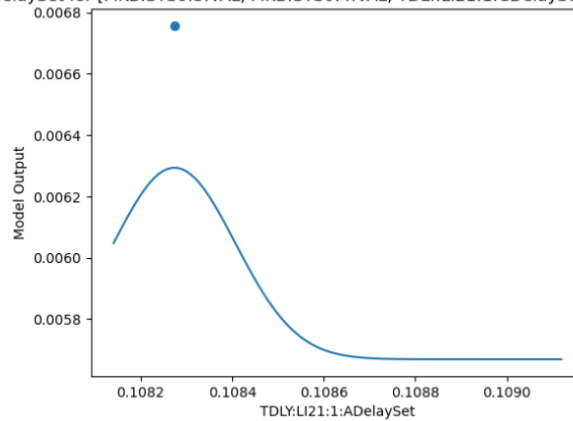


Another initial issue that I found was that the changes in some of the inputs for the x-variable tensor varied very little over the interval, while some varied quite a bit. This led some of the initial testing to have some bad data, as different variables carried different weight in the Gaussian process. I fixed this by incorporating different length scales to each of the columns of the x-variable tensor.

```
In [14]: model.covar_module.base_kernel.lengthscale #different lengthscales for each input dimensions
Out[14]: tensor([[44.5281, 0.0761, 0.6698, 0.8312]], grad_fn=<SoftplusBackward0>)
```

The end result for the plots that I managed to get showed that each of the data points would “pull” the predicted `obj_sum` closer to the point, so we could see that the model was working.

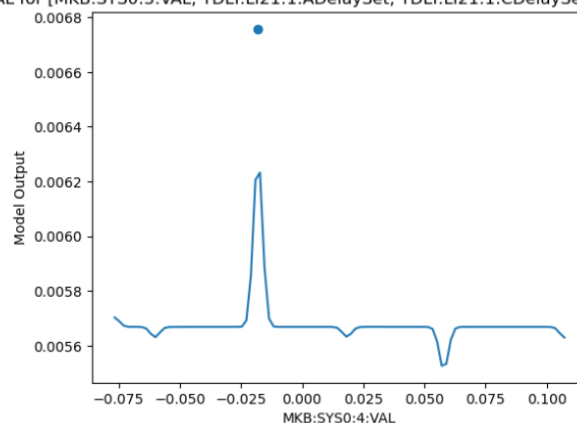
Posterior Mean vs TDLY:LI21:1:ADelaySet for [MKB:SYS0:3:VAL, MKB:SYS0:4:VAL, TDLY:LI21:1:CDelaySet] = [0.059973408, 0.10827365, 0.1113082]



Conclusion

In conclusion, my primary tasks were to use the LCLS-archive to plot PVs from each yaml file relative to time and to iteration and also histograms, to create documents to record and plot interesting properties from the Pvs within the yaml files, and to run a simple regression Gaussian process on the data. These tasks required several tutorials and examples, and quite a bit of time to learn how to use the necessary tools and gain familiarity with SLAC's systems. The first task required troubleshooting with iterations vs time, and I found that the iterations didn't perfectly coincide with the times because the runs weren't evenly spaced. In the second task, we found that the yaml files weren't uniform in their properties, but some of the files were pairs that shared similarities. I also found that some PVs changed names depending on the file. In the final task, after troubleshooting issues with dimensionality and memory, we found that the data points pulled the predicted obj_sum towards the point, as expected.

Posterior Mean vs MKB:SYS0:4:VAL for [MKB:SYS0:3:VAL, TDLY:LI21:1:ADelaySet, TDLY:LI21:1:CDelaySet] = [0.059973408, 0.10827365, 0.1113082]



Recommendations

If this internship was to be repeated, there are several things that could be improved. The first is that it is extremely difficult to gain access to materials online without having necessary identification like SLAC email and SSH keys. This isn't necessarily an issue in and of itself, as it is a good precaution against identity theft or cybersecurity breaches, but it causes problems when you can't access any of the tutorials or necessary resources to do other things like SDF or any of the instructional pages. A good change would be to change the pages that aren't threatening like instructional pages or tutorials to not require this identification. My next recommendation would be to create instructional resources specifically for the internship, especially things having to do with computer science. As a physics major who had little experience with programming, I found many basic things difficult to learn in tandem with SLAC tasks, like directories, environments, packages, GitHub, unix, etc. I would also recommend making a GitHub for these things beyond what I have done, things like environments, kernels, etc. The Xopt GitHub was good, but there were issues that were impossible for an amateur to understand, like the incompatibility with the archive template notebook that we didn't know of beforehand. The same kind of issue is present with PyCharm, Ubuntu, and Jupyter, and I even had some exposure with these before the internship. I would also recommend having a specified time each week where there is a live 1-on-1 interaction with the intern where you go over instructions for the tasks, since these huddles were by far the most insightful periods during the internship. Having guidance at some points made what would take me hours alone take minutes instead. My last suggestion would be to have clearer instructions in general with some of the tasks. For example, some of the tasks were supposed to be tackled before I was given resources like 1-on-1 sessions or example notebooks, which would have greatly improved work efficiency if given early. In addition, certain issues would be avoided earlier or entirely, I wouldn't discover things like the dimensionality issue when plotting if it was mentioned earlier. This one is debatable to be worth changing though, as it was the final project so it makes sense that there is a lot less help and a lot more troubleshooting, and I also tried to do it largely alone to create a more realistic research experience.

Acknowledgements

Special thanks to Zhirong Huang, Auralee Edelen, Christopher Mayes, Alex Halavanau, and Ryan Roussel for supervising my internship, and to Yee Ting-Li, Michael Kelsey, Lance Nakata, Dylan Kennedy, Matt Gibbs, and William Colacho for assisting me from outside the internship.