ML Python package Xopt for online tuning*

Ryan Roussel

rroussel@slac.stanford.edu

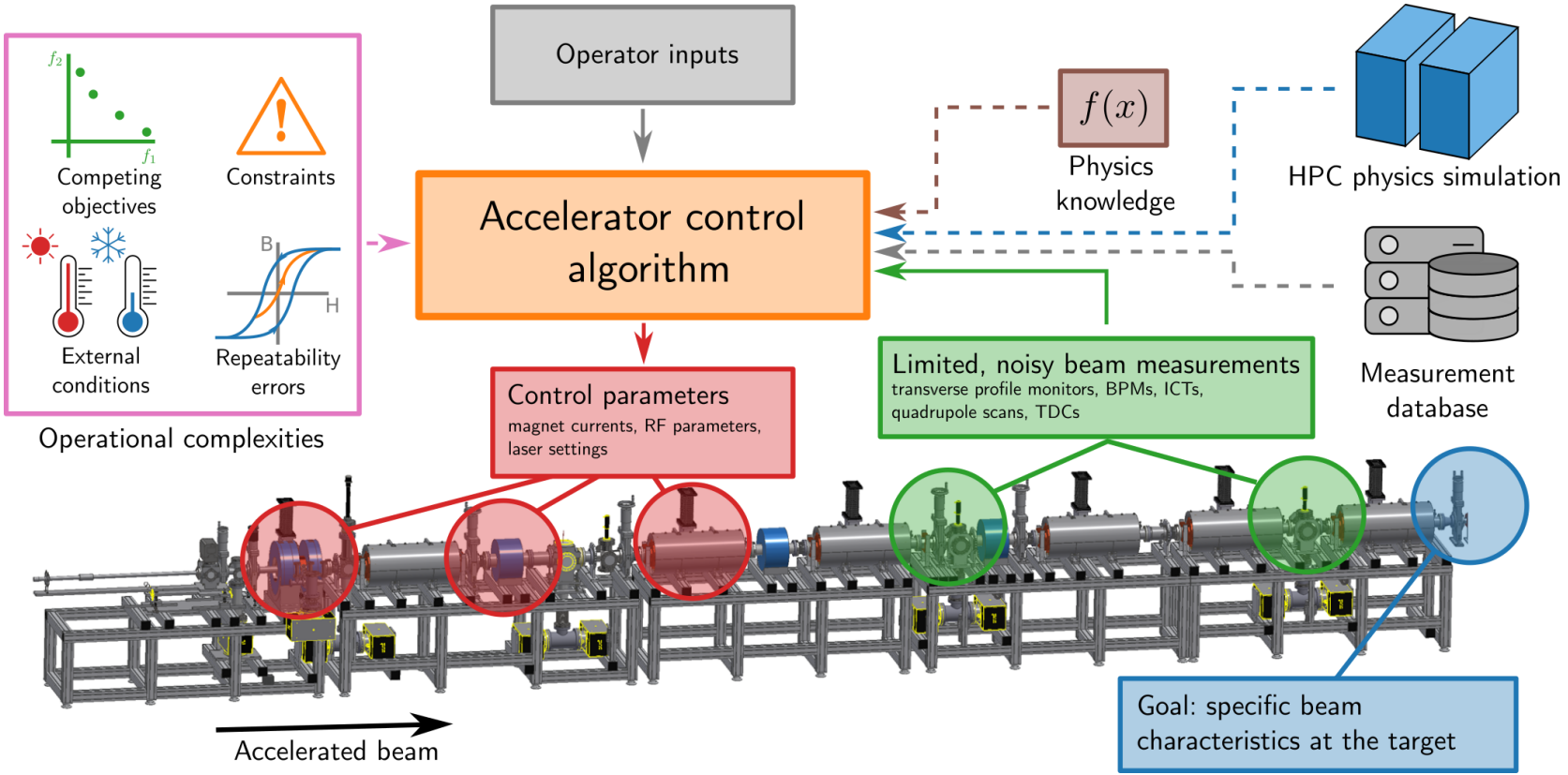
6/20/2024

**and some other stuff*

U.S. DEPARTMENT OF ENERGY | Stanford University

SLAC NATIONAL ACCELERATOR LABORATORY

Machine Learning Based Accelerator Control



What is Xopt?

- Flexible **framework** for optimization of arbitrary problems using python
- **Independent** of problem type (simulation or experiment)
- **Independent** of optimization algorithm + easy to incorporate custom algorithms
- **Easy to use** text interface and/or advanced customized use for professionals

https://github.com/ChristopherMayes/Xopt

Overview

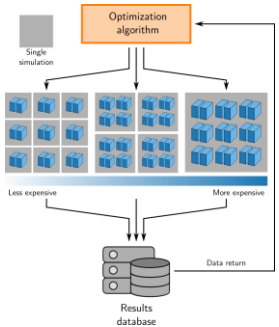
Xopt algorithm implementation

Production ready control

Badger GUI interface

https://slaclab.github.io/Badger/

https://christophermayes.github.io/Xopt/



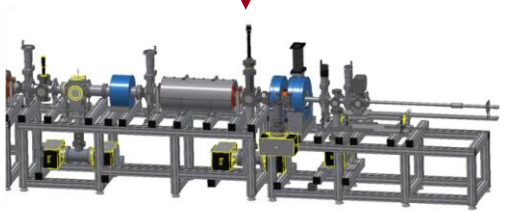

```
xopt:
    max_evaluations: 6400

generator:
    name: cnsga
    population_size: 64
    population_file: test.csv
    output_path: .

evaluator:
    function: xopt.resources.test_functions.tnk.evaluate_TNK
    function_kwargs:
      raise_probability: 0.1

vocs:
    variables:
        x1: [0, 3.14159]
        x2: [0, 3.14159]
    objectives: {y1: MINIMIZE, y2: MINIMIZE}
    constraints:
        c1: [GREATER_THAN, 0]
        c2: [LESS_THAN, 0.5]
    linked_variables: {x0: x1}
    constants: {a: dummy_constant}
```

YAML file

Online Control R&D

Python interface

```
# create Xopt object.
X = Xopt(YAML)

# take 10 steps and view data
for _ in range(10):
    X.step()

X.data
```



Accelerator simulation

Arbitrary problem

Experiment facility

Technology Transfer Between Accelerator Facilities

Argonne NATIONAL LABORATORY

Accel. Control R&D

SLAC NATIONAL ACCELERATOR LABORATORY

Xopt

ESRF The European Synchrotron

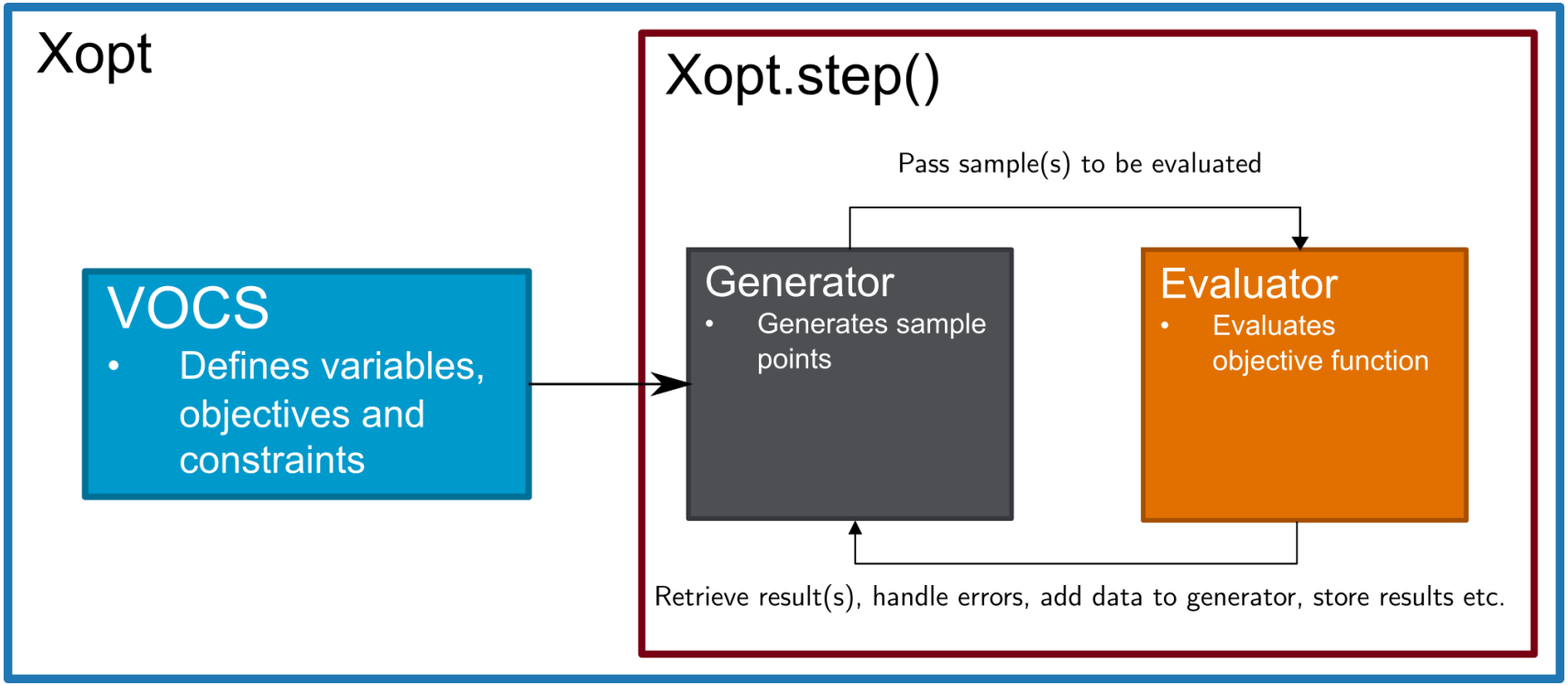
Cornell University Founded A.D. 1865

Fermilab

Brookhaven National Laboratory

DESY.

Xopt structure



Note: this process can also be done asynchronously

Xopt Script Overview – Defining the problem

Define the domain/goals

$$x_1, x_2 \in [0, \pi]$$

$$\mathbf{x}^* = \arg\min f(\boldsymbol{x})$$
$$g(\boldsymbol{x}) \leq 0$$

In [2]:

```
from xopt import VOCS
import math

vocs = VOCS(
    variables = {
        "x1": [0, math.pi],
        "x2": [0, math.pi]
    },
    objectives = {"f": "MINIMIZE"},
    constraints = {"g": ["LESS_THAN", 0]}
)
```

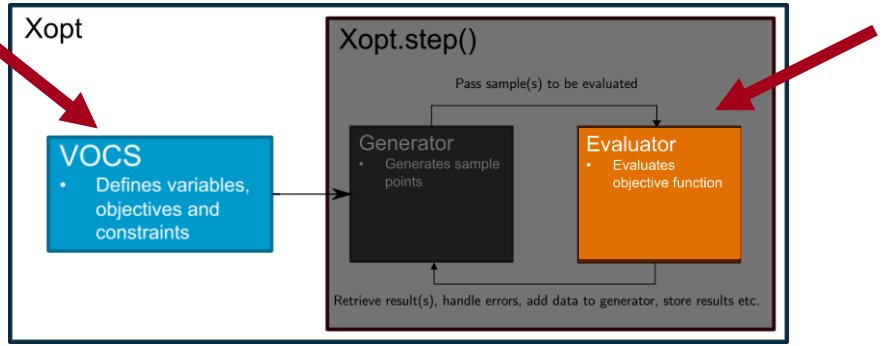
Define the objectives/constraints

$$f(x_1, x_2) = x_1^2 + x_2^2$$
$$g(x_1, x_2) = 1 - x_1^2 - x_2^2$$

```
evaluate(inputs: dict) -> dict
```

In [1]:

```
from xopt import Evaluator

def evaluate_function(inputs: dict) -> dict:
    objective_value = inputs["x1"]**2 + inputs["x2"]**2
    constraint_value = -inputs["x1"]**2 - inputs["x2"]**2 + 1
    return {"f": objective_value, "g": constraint_value}

evaluator = Evaluator(function=evaluate_function)
```

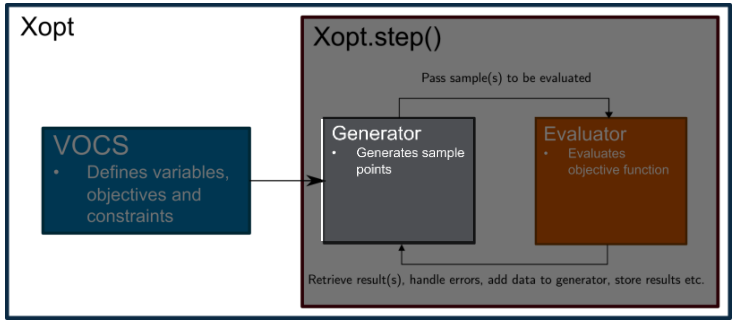


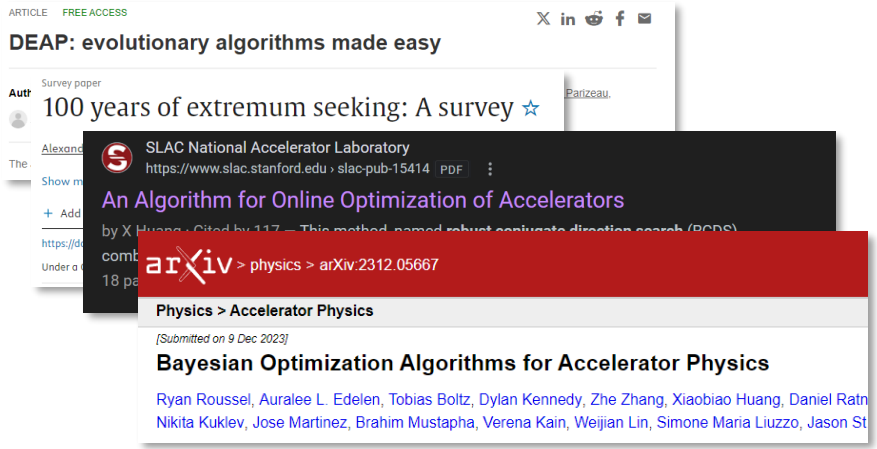
https://christophermayes.github.io/Xopt/examples/basic/xopt_basic/

Xopt Script Overview – Defining the algorithm

Choose from available generators
(or define your own)

- Optimization algorithms:
  - Genetic algorithms
    - `cnsga` Continuous NSGA-II with constraints
  - Bayesian optimization (BO) algorithms:
    - `upper_confidence_bound` BO using Upper Confidence Bound acquisition function (w/ or w/o constraints, serial or parallel)
    - `expected_improvement` BO using Expected Improvement acquisition function (w/ or w/o constraints, serial or parallel)
    - `mobo` Multi-objective BO (w/ or w/o constraints, serial or parallel)
    - `bayesian_exploration` Autonomous function characterization using Bayesian Exploration
    - `mggpo` Parallelized hybrid Multi-Generation Multi-Objective Bayesian optimization
    - `multi_fidelity` Multi-fidelity single or multi objective optimization
    - `BAX` Bayesian algorithm execution using virtual measurements
    - BO customization:
      - Trust region BO
      - Heteroskedastic noise specification
      - Multiple acquisition function optimization stratigies
  - `extremum_seeking` Extremum seeking time-dependent optimization
  - `rcds` Robust Conjugate Direction Search (RCDS)
  - `neldermead` Nelder-Mead Simplex





https://arxiv.org/abs/2312.05667 (accepted by PRAB)

Xopt Script Overview – Putting it all together

Create Xopt object

In [5]:
```
from xopt import Xopt
X = Xopt(vocs=vocs, generator=generator, evaluator=evaluator)
```

Evaluate explicit points

In [10]:
```
# evaluate some points additionally
points = {"x1": [1.0, 0.5, 2.25],"x2":[0,1.75,0.6]}
X.evaluate_data(points)
```

Visualize results

```
# view objective values
X.data.plot(y=X.vocs.objective_names)

# view variables values
X.data.plot(*X.vocs.variable_names, kind="scatter")
```

Run optimization

In [12]:
```
# Take one step (generate a single point)
X.step()
```

https://christophermayes.github.io/Xopt/examples/basic/xopt_basic/

Example: Online Optimization at SLAC - Setup

Create beam size objective function

```python
from epics import caput, caget_many
from time import sleep
import numpy as np
def eval_beamsize(inputs):
        global image_diagnostic
        # set PVs
        for k, v in inputs.items():
            print(f'CAPUT {k} {v}')
            caput(k, v)

        sleep(2.0)

        # get beam sizes from image diagnostic
        metadata = inputs
        results = image_diagnostic.measure_beamsize(5, **metadata)
        results["S_x_mm"] = np.array(results["Sx"]) * 1e-3
        results["S_y_mm"] = np.array(results["Sy"]) * 1e-3

        # add total beam size
        results["total_size"] = np.sqrt(np.array(results["Sx"]) ** 2 + np.array(results["Sy"]) ** 2)
        # results["total_size"] = np.sqrt(np.abs(np.array(results["Sx"])) * np.array(results["Sy"]))
        return results
```

Set beamline parameters

Wait for power supplies/feedback to settle

Measure beam size

Calculate the objective

Image measurement class

```python
class ImageDiagnostic(BaseModel):
    screen_name: str
    array_data_suffix: str = "Image:ArrayData"
    array_n_cols_suffix: str = "Image:ArraySize0_RBV"
    array_n_rows_suffix: str = "Image:ArraySize1_RBV"
    resolution_suffix: Union[str, None] = "RESOLUTION"
    resolution: float = 1.0
    beam_shutter_pv: str = None
    extra_pvs: List[str] = []

    background_file: Optional[str] = None
    save_image_location: Optional[str] = None
    roi: Optional[ROI] = None

def measure_beamsize(self, n_shots: int = 1, fit_image=True, **kwargs):
    """
    conduct a multi-shot measurement to get the beam size from images, r
    sizes in units of `resolution`

    allows attaching extra information to dataset via kwargs
    """
```

Example: Online Optimization at SLAC - Results

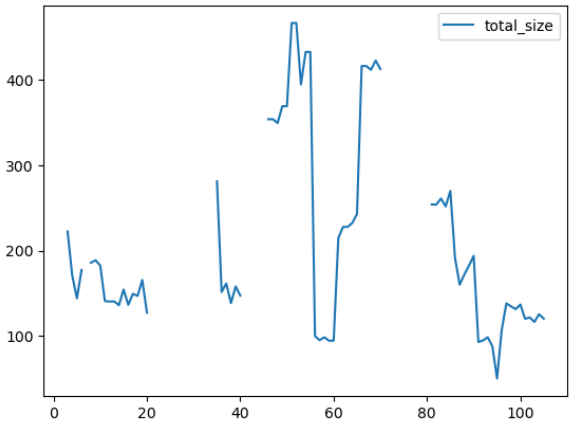
Results data frame (incl. metadata)

SOLN:IN20:121:BCTRL	…	…	0.474877
QUAD:IN20:121:BCTRL	…	…	-0.00484
QUAD:IN20:122:BCTRL	…	…	0.0018
QUAD:IN20:361:BCTRL	…	…	-3.16
QUAD:IN20:371:BCTRL	…	…	2.53527
QUAD:IN20:425:BCTRL	…	…	-1.1
QUAD:IN20:441:BCTRL	…	…	-0.81186
QUAD:IN20:511:BCTRL	…	…	3.649406
QUAD:IN20:525:BCTRL	…	…	-3.252219
Cx	…	…	479.324935
...	…	…	...
Sy	…	…	136.386527
bb_penalty	…	…	-145.364148
total_intensity	…	…	1245909.6
log10_total_intensity	…	…	6.095487
save_filename	…	…	/home/physics3/ml_tuni
S_x_mm	…	…	0.175659
S_y_mm	…	…	0.136387
total_size	…	…	222.390057
xopt_runtime	…	…	6.993356
xopt_error	…	…	False

Visualization

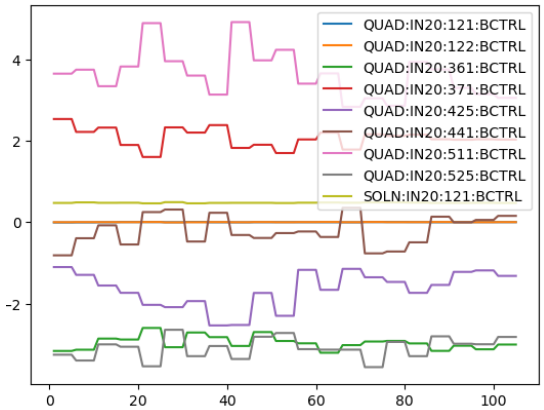
Objectives

```
X.data.plot(y="total_size")
```



Variables

```
X.data.plot(y=X.vocs.variable_names)
```



… can dump data to database / xarray instead of pd.DataFrame

Pydantic Integration

Xopt objects are robustly validated and serialized/de-serialized via Pydantic

YAML file

```yaml
xopt:
    max_evaluations: 6400

generator:
    name: cnsga
    population_size: 64
    population_file: test.csv
    output_path: .

evaluator:
    function: xopt.resources.test_functions.tnk.evaluate_TNK
    function_kwargs:
      raise_probability: 0.1

vocs:
    variables:
        x1: [0, 3.14159]
        x2: [0, 3.14159]
    objectives: {y1: MINIMIZE, y2: MINIMIZE}
    constraints:
        c1: [GREATER_THAN, 0]
        c2: [LESS_THAN, 0.5]
    linked_variables: {x9: x1}
    constants: {a: dummy_constant}
```

Python object(s)

```python
X = Xopt.from_yaml(open("my_file.yml"))
fig, ax = X.generator.visualize_model(
    variable_names = X.vocs.variable_names
)
```

```python
X.dump()
```

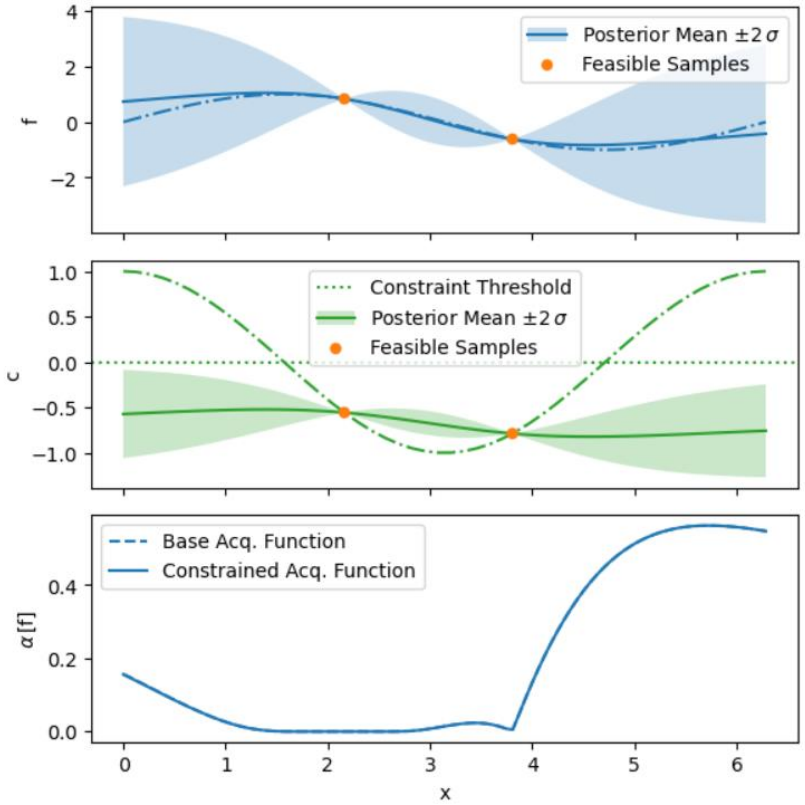
YAML file

```yaml
xopt:
    max_evaluations: 6400

generator:
    name: cnsga
    population_size: 64
    population_file: test.csv
    output_path: .

evaluator:
    function: xopt.resources.test_functions.tnk.evaluate_TNK
    function_kwargs:
      raise_probability: 0.1

vocs:
    variables:
        x1: [0, 3.14159]
        x2: [0, 3.14159]
    objectives: {y1: MINIMIZE, y
    constraints:
        c1: [GREATER_THAN, 0]
        c2: [LESS_THAN, 0.5]
    linked_variables: {x9: x1}
    constants: {a: dummy_constant}
data:
  Cx:
    '1': 378.5739219281
    '10': 420.7214465998
    '100': 438.3514501154
    '101': 466.4557444371
```

Example: BO w/ introspection

```python
from xopt.evaluator import Evaluator
from xopt.generators.bayesian import UpperConfidenceBoundGenerator
from xopt import Xopt

evaluator = Evaluator(function=sin_function)
generator = UpperConfidenceBoundGenerator(vocs=vocs)
X = Xopt(evaluator=evaluator, generator=generator, vocs=vocs)
```

```python
for i in range(n_steps):

    model = X.generator.train_model()
    fig, ax = X.generator.visualize_model(n_grid=100)


    # add ground truth functions to plots
    out = test_function({"x": test_x})
    ax[0].plot(test_x, out["f"], "C0-.")
    ax[1].plot(test_x, out["c"], "C2-.")

    # do the optimization step
    X.step()
```

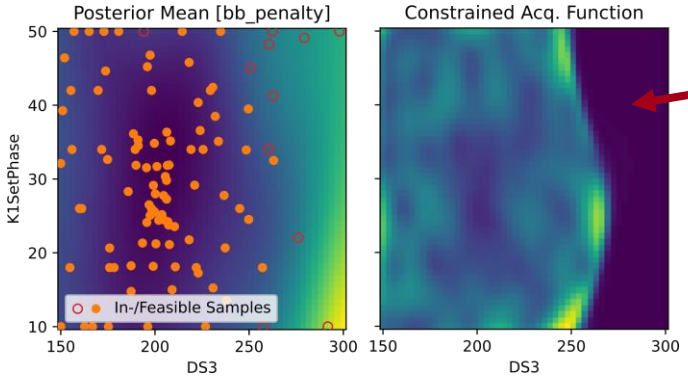


https://christophermayes.github.io/Xopt/examples/single_objective_bayes_opt/constrained_bo_tutorial/

Example: Automatic Characterization w/ Constraints

AWA Example

- photoinjector characterization for model calibration

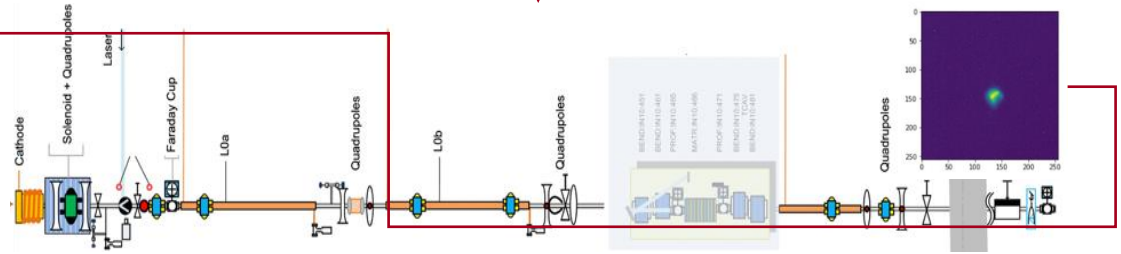


FACET-II Example

Automatic Exploration *(constrained to useful values of emittance and match)*

Comprehensive ML Models of Injector

Setting changes on 10 variables (solenoid, bucking coil, corrector quads and matching quads)

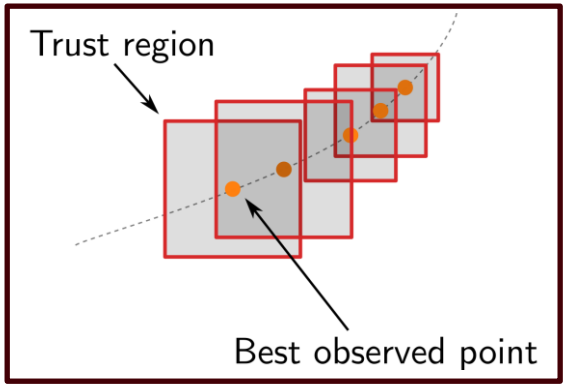


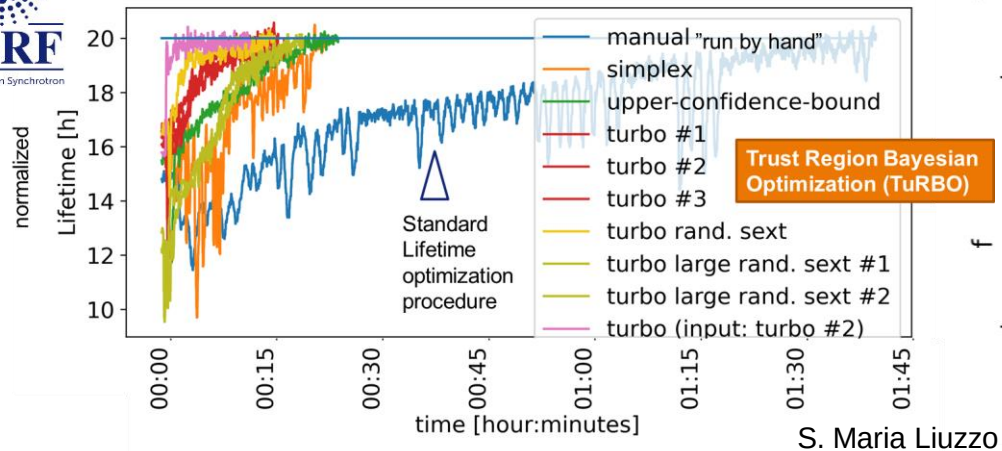
x-y emit, match, and beam images

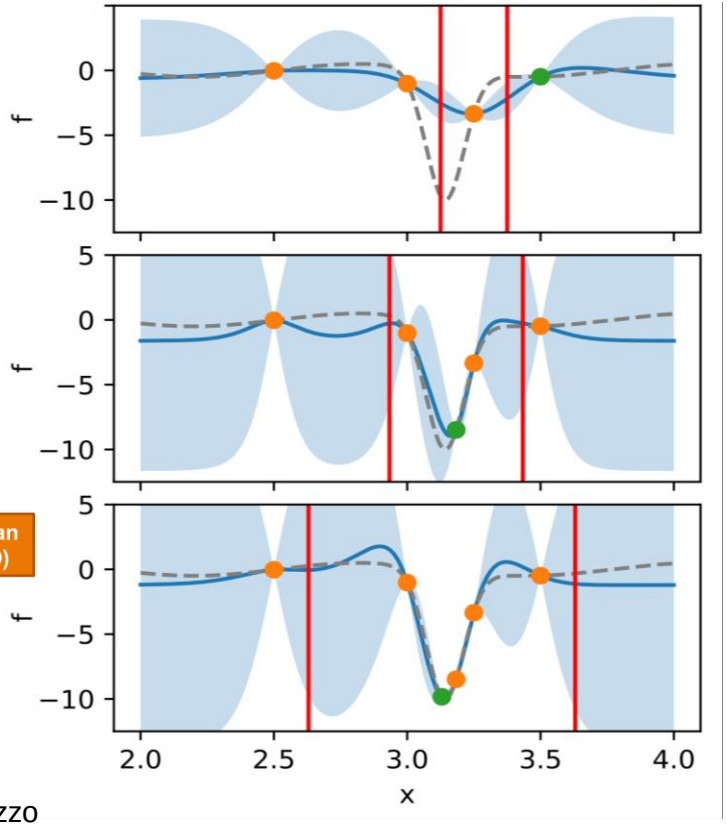
FACET-II Injector

Efficient high-dimensional characterization (10 variables) of emittance and match at 700pC:
2 hrs for 10 variables vs. to 5 hrs for 4 variables with N-D parameter scan

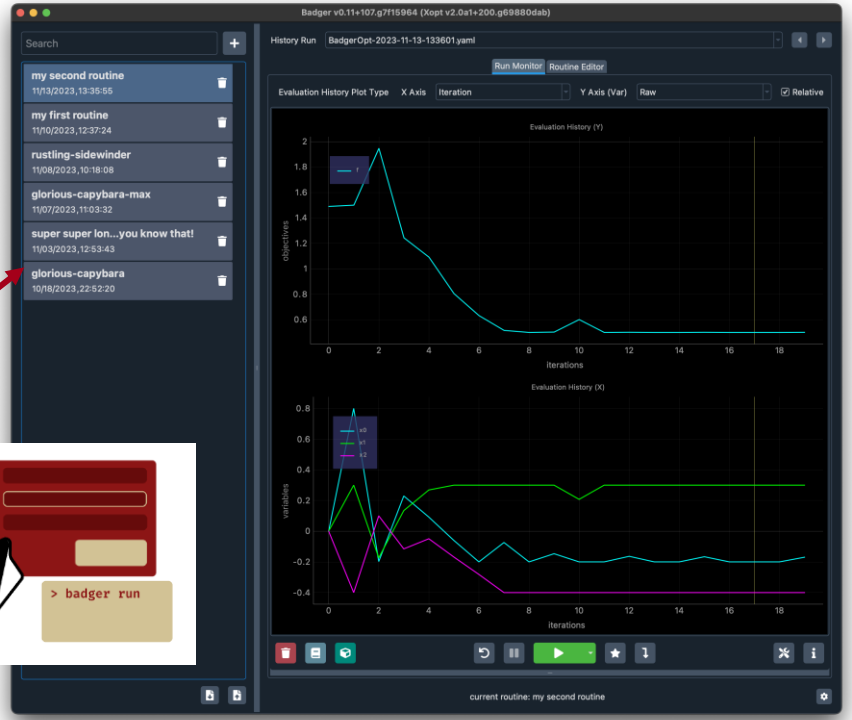
Example: Trust Region BO







Control Room Optimization w/ Badger - Interface



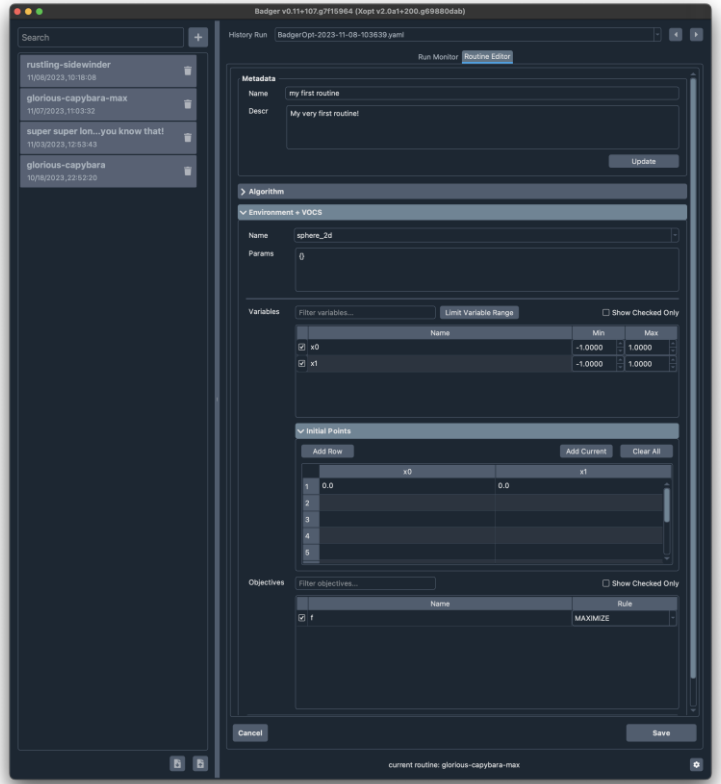
Re-run old optimization configurations

Optimization monitoring

Easy run control (play/pause/load optimal)

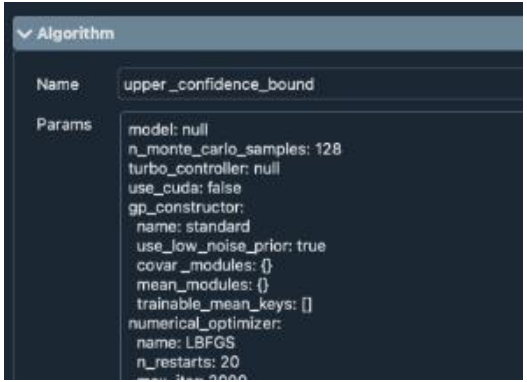
Control Room Optimization w/ Badger – Routine

Routine Specification



Routine metadata

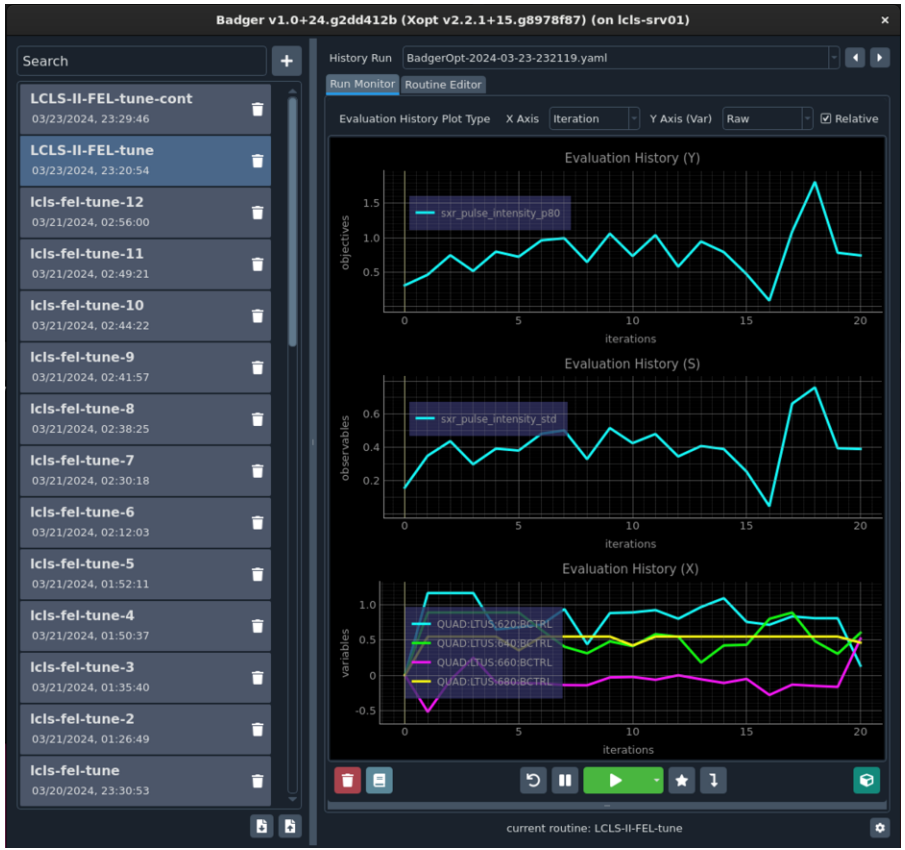
Algorithm specification



Variables / constraints / objectives specification

Example: LCLS-II Optimization w/Badger

- Maximize 80th percentile FEL energy using a set of quadrupoles (integrated over 100 pulses)
- Includes a constraint on the FEL pulse intensity jitter



Connecting Machine Operation to Badger

```python
class Environment(environment.Environment):

    name = 'sphere_3d'  # name of the environment
    variables = {  # variables and their hard-limited ranges
        'x0': [-1, 1],
        'x1': [-1, 1],
        'x2': [-1, 1],
    }
    observables = ['f']  # measurements
```

```python
    # Variable getter -- tells Badger how to get current values of the variables
    def get_variables(self, variable_names):
        variable_outputs = {v: self._variables[v] for v in variable_names}

        return variable_outputs

    # Variable setter -- how to set variables to the given values
    def set_variables(self, variable_inputs: dict[str, float]):
        for var, x in variable_inputs.items():
            self._variables[var] = x

        # Filling up the observations
        f = self._variables['x0'] ** 2 + self._variables['x1'] ** 2 + \
            self._variables['x2'] ** 2

        self._observations['f'] = [f]

    # Observable getter -- how to get current values of the observables
    def get_observables(self, observable_names):
        return {k: self._observations[k] for k in observable_names}
```

Simple python interface to define problem variables / observables (objectives/constraints)
→ populates Badger fields

Xopt Summary

Easy to control

```python
# create Xopt object.
X = Xopt(YAML)

# take 10 steps and view data
for _ in range(10):
    X.step()

X.data
```

Python interface

Badger GUI interface

Simple to connect with simulations / machine (single python function!)

```python
evaluate(inputs: dict) -> dict
```

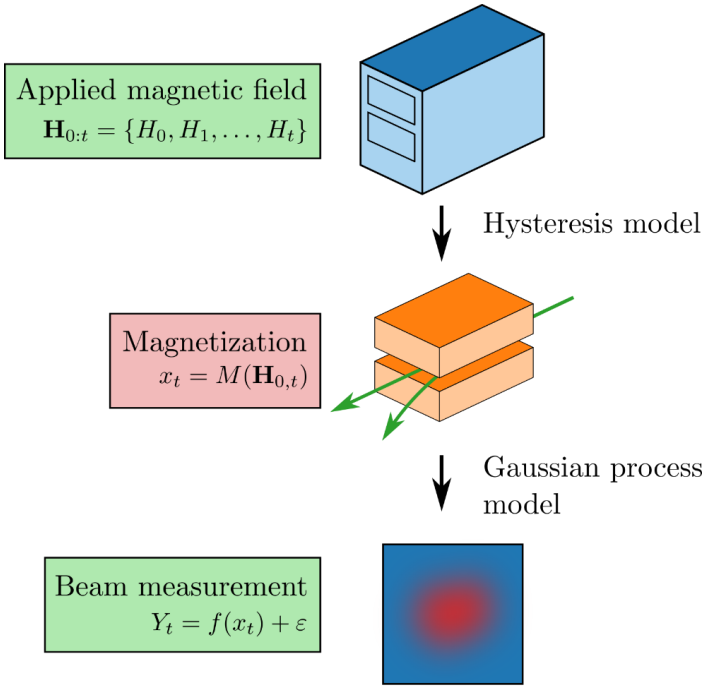
https://christophermayes.github.io/Xopt/

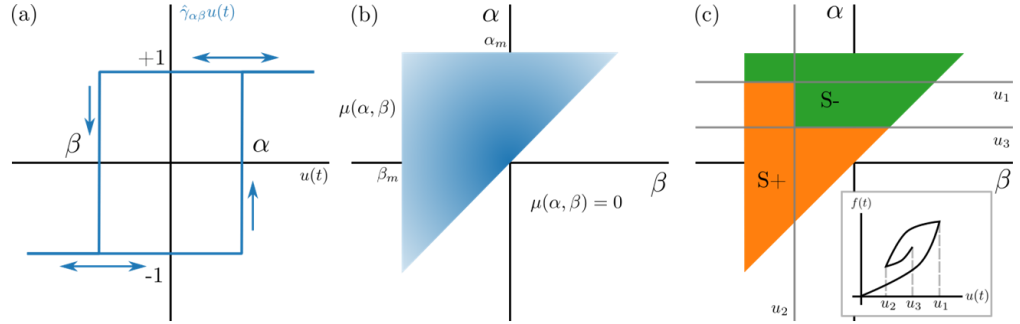
Future Work

- Additional optimization algorithms
  - Reinforcement learning
  - Adv. Bayesian optimization
  - Genetic algorithms
  - Asynchronous optimization
- Improve interoperability
  - Generator standards (optimus, libensemble)
  - Surrogate model packages (LUME)
- Leverage pyML tools


“And now for something completely different.”
Monty Python

Modeling Hysteresis in Accelerators





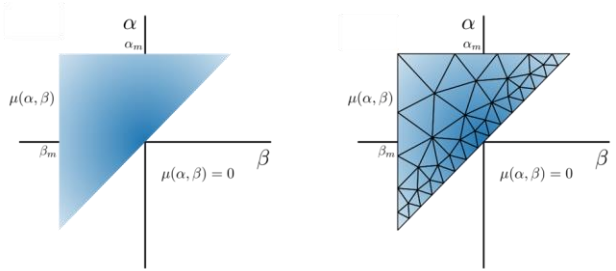
Model magnetic domains as individual "hysterons"

Determine the hysteron density

Determine hysteron states via history of applied fields

$$M(\mathbf{H}_{0:t}) = \int_{S+} \mu(\beta,\alpha)d\beta d\alpha - \int_{S-} \mu(\beta,\alpha)d\beta d\alpha$$

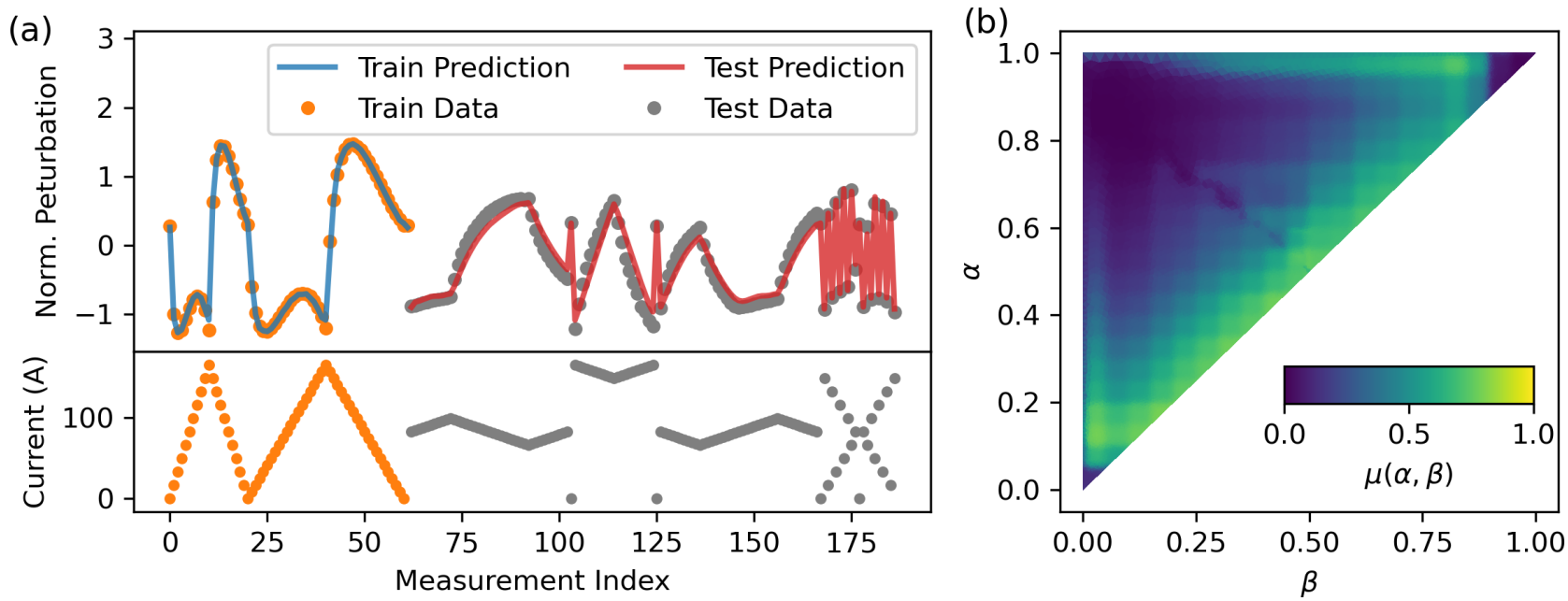
Limited by model selection



Use ML techniques to fit high-fidelity hysteresis model

$$\mu(\alpha,\beta) = f(\alpha,\beta;\theta) \rightarrow \mu_i = \mu(\alpha_i,\beta_i)$$

Modeling SLAC Quadrupoles

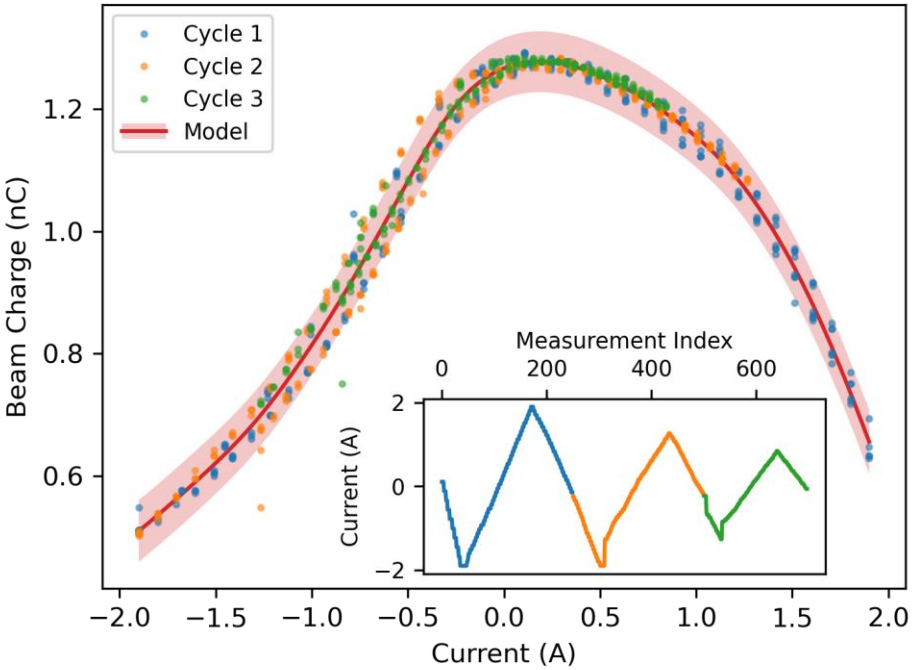


Polynomial fit error: 0.23% Train error: 0.015% Test error: 0.051%

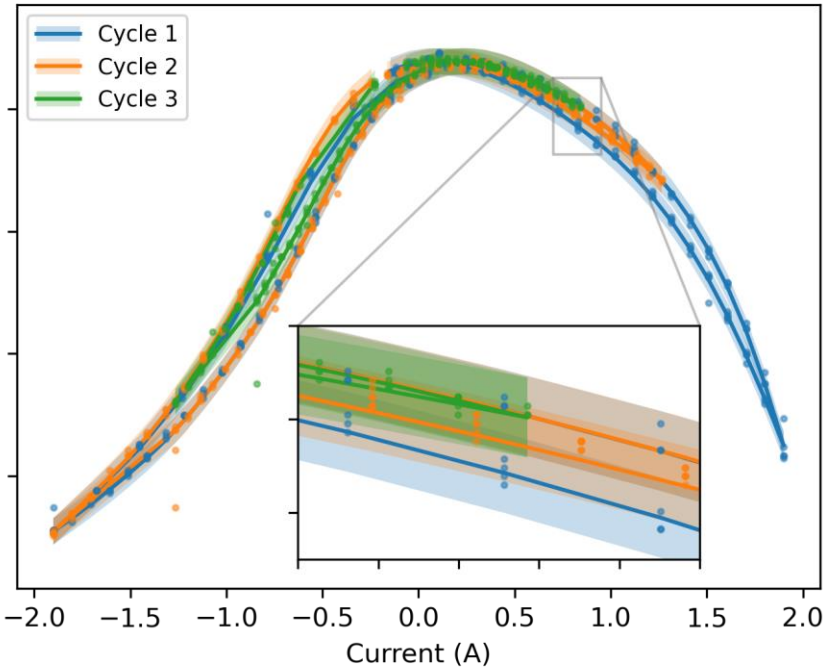
Roussel R. et al., PRL 2022

Full Stack Hysteresis Modeling

Modeling w/o Hysteresis



Modeling w/ Hysteresis

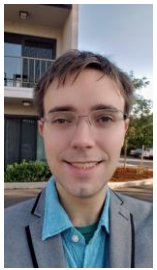


Roussel R. et al., PRL 2022

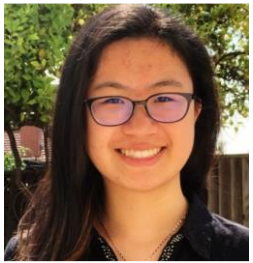
Questions?

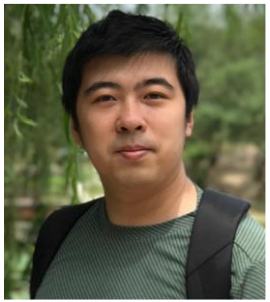
Thanks to the team!

Note: there are another applications of machine learning in python middle layer processes.







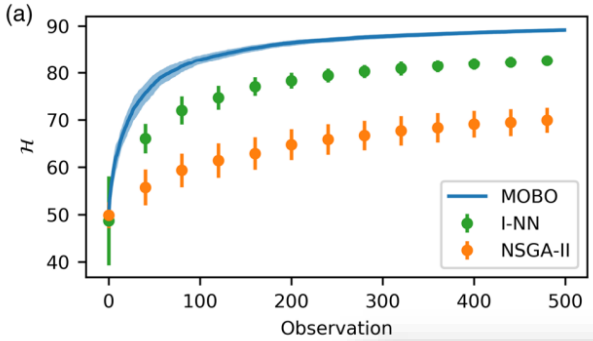


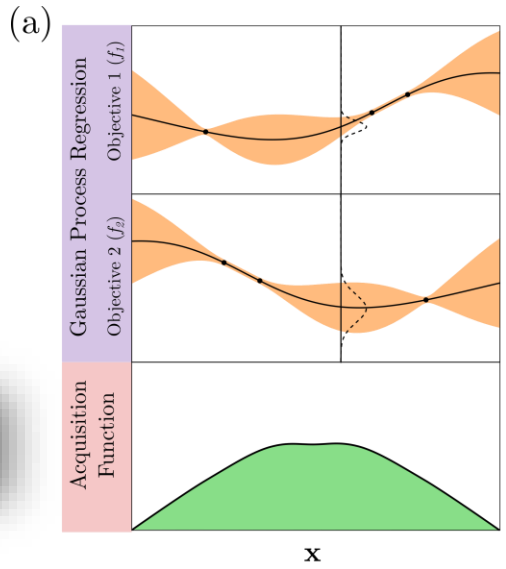


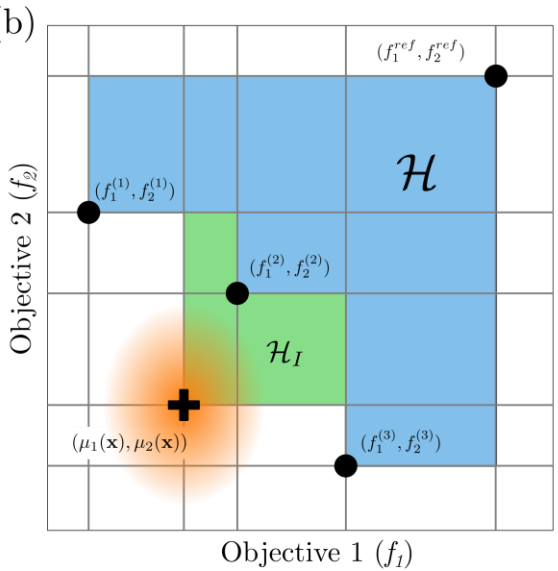
Backup

Multi-Objective Optimization

Determine the optimal trade-off between objectives -> the Pareto front







$$\alpha_{EHVI}(\boldsymbol{\mu},\boldsymbol{\sigma},\mathcal{P},\boldsymbol{r}) := \int_{\mathbb{R}^P} \mathrm{HVI}(\mathcal{P},\boldsymbol{y},\boldsymbol{r}) \cdot \boldsymbol{\xi}_{\mu,\sigma}(\boldsymbol{y}) d\boldsymbol{y}$$

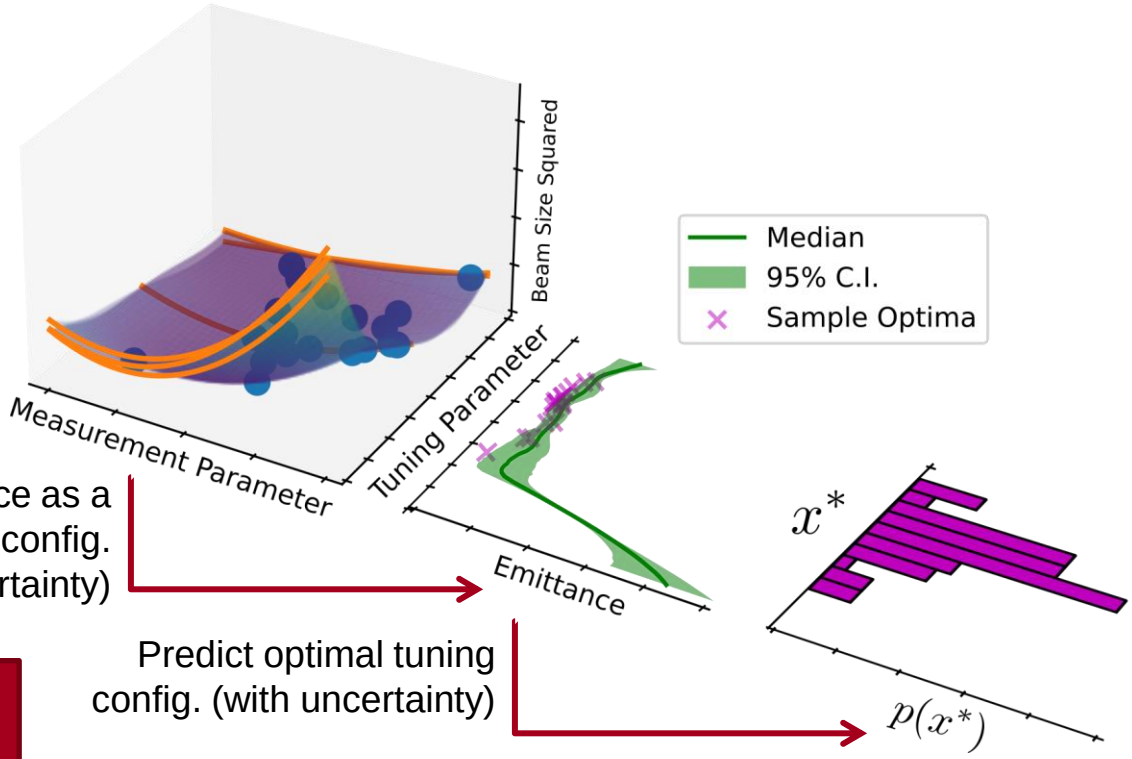
Xopt example

Roussel et. al. PRAB **2021**

Bayesian Algorithm Execution (BAX)

Our goal:
Determine the **ideal tuning configuration x^*** which minimizes emittance

Beam size model



Predict beam emittance as a function of tuning config. (with uncertainty)

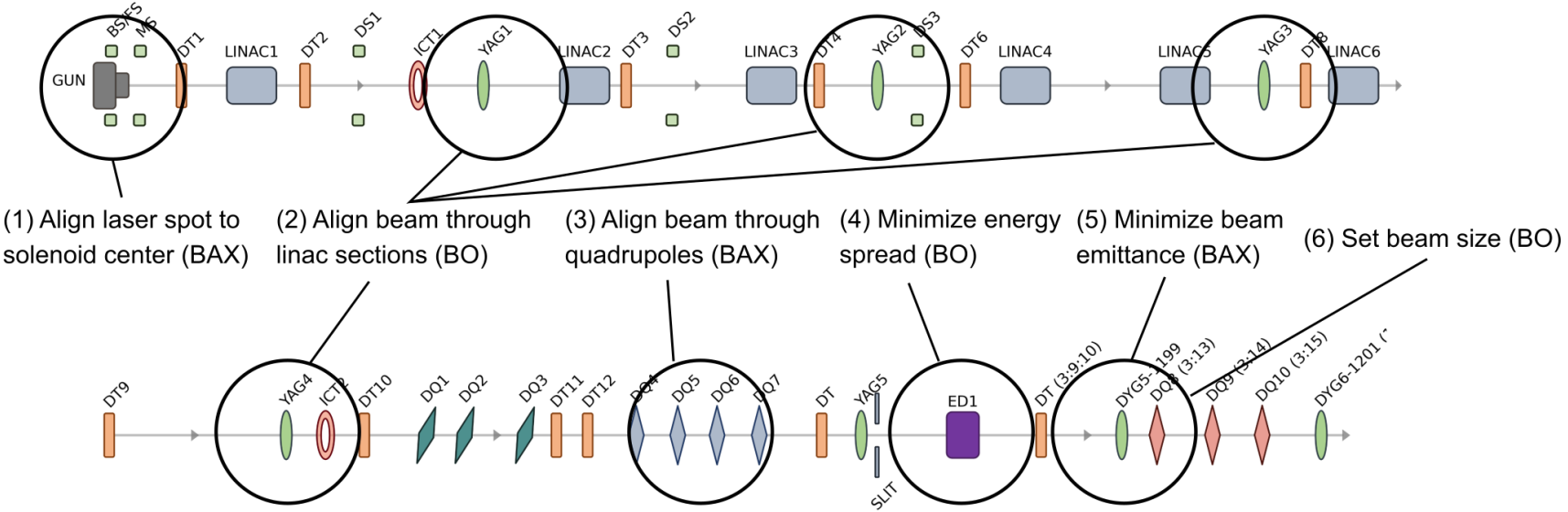
Predict optimal tuning config. (with uncertainty)

The BAX algorithm chooses beam size measurements that reduce uncertainty in x^* **without measuring emittance directly → 20x speed up**

Xopt example

Miskovich, et. al., arXiv:2209.04587

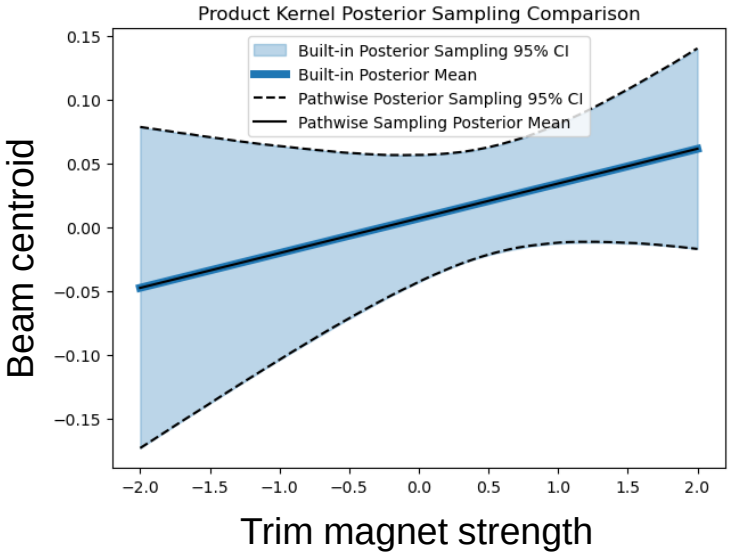
Autonomous Operation of AWA



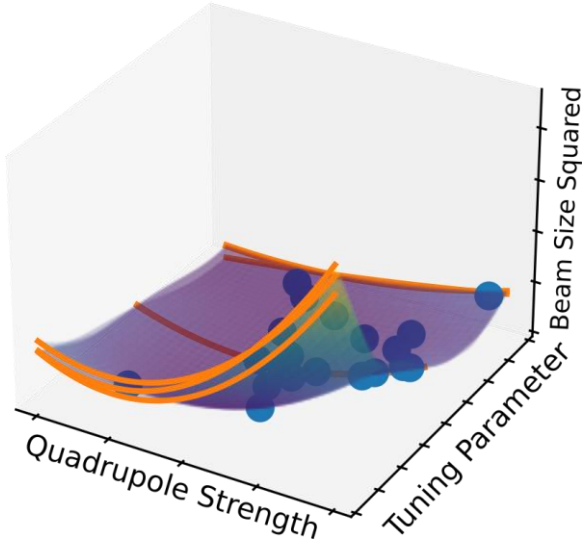
(2),(3),(5) --> demonstrated (1),(4),(6) --> in progress

Incorporating Physics Information into Kernels

Enforce linear centroid response to steering magnets

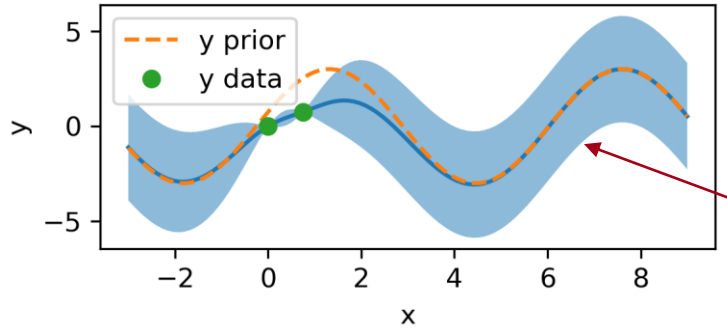


Enforce quadratic beam size squared response to quadrupole magnets



Combining GP Modeling with Neural Networks

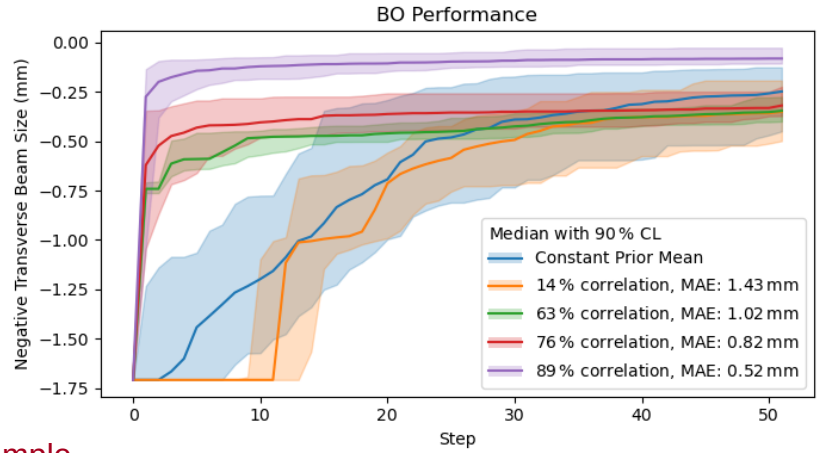
We can specify a **prior mean** function to bias the model where data does not exist



Model prediction returns to prior

Using a NN prior improves optimization performance even with limited model accuracy; can adapt to bad models

LCLS injector surrogate



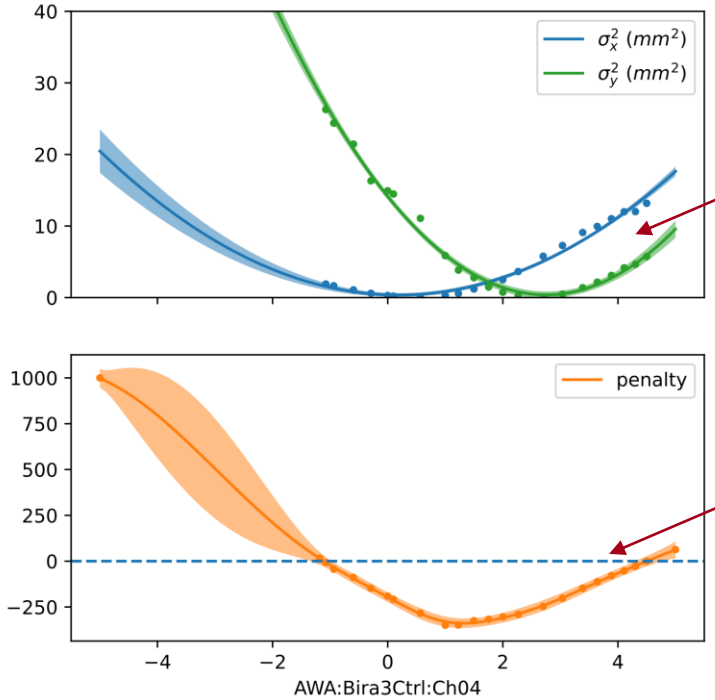
Xopt example

Example Use Case: Automated Quadrupole Scans

Example: Automate quadrupole scan measurements starting from a single valid measurement

- Independent of upstream accelerator parameter configuration

- Bayesian Exploration w/constraints
- Enforce quadratic dependence using kernel
- Can be extended to include other accelerator parameters (solenoid, skew quad, etc.) to do full emittance characterization
- Can tune safety factor as necessary for critical constraints

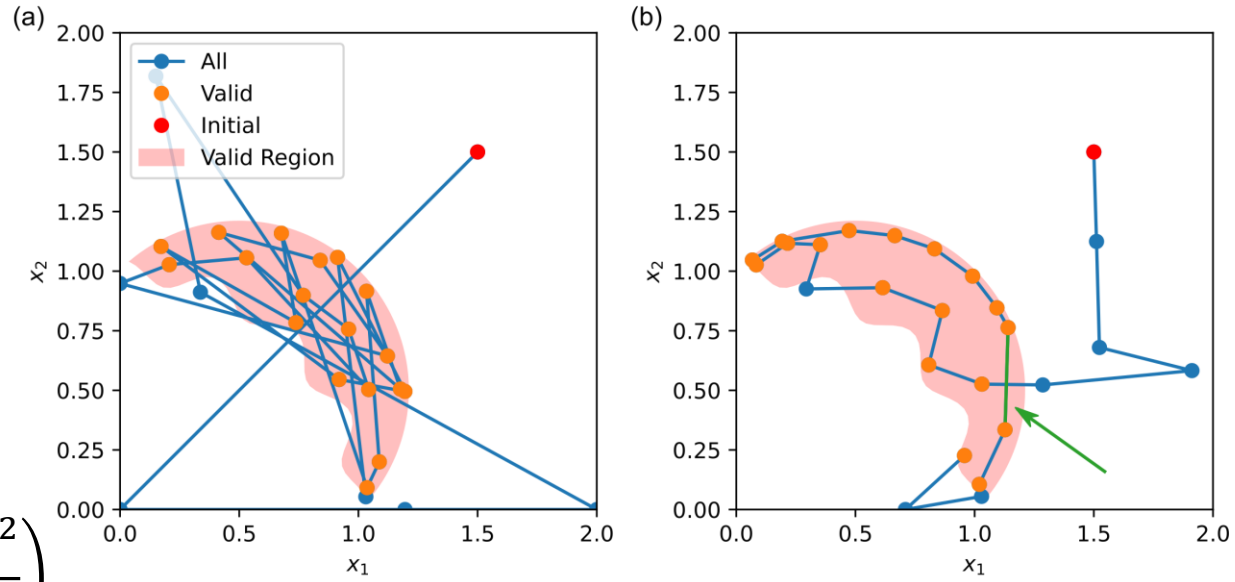


2nd order polynomial fits to data

Beam bounding box penalty

Proximal Biasing

Poor optimization behavior for experimental beamlines



Weight the acquisition function by travel distance → better than hard limits

$$\hat{\alpha}(x) \rightarrow \alpha(x) \exp\left(-\frac{(x - x_0)^2}{2\sigma^2}\right)$$

Warning: Requires $\alpha(x) \geq 0$

Roussel et. Al. *Nat. Comm.* **2021**