Supplementary Material of "Bayesian Optimization of `Pythia8` Tunes"

Ali Al Kadhim* and Harrison B Prosper†
Department of Physics, Florida State University, Tallahassee, USA

Stephen Mrenna
Fermilab, Batavia, USA
(Dated: May 13, 2025)

CONTENTS

I. SUPPLEMENTAL MATERIAL

A. yoda (.yoda) format

The .yoda histograms for the data have the following column headings:

```
# xval  xerr-  xerr+  yval  yerr-  yerr+
```

`xval` is the bin center, while 'yval' is the height of the bin center. `xerr-=xerr+` is the positive and uncertainties in the bin center location, and `yerr-=yerr+` is the uncertainty in the height.

The data recorded is the cross section. so let's call that cross section in bin i $\hat{\sigma}_i^{Data} = yval$ and the uncertainty in bin i $\Delta_i^{Data} = yerr-$.

while the simulation histograms have the following column headings:

```
xlow  xhigh  sumw  sumw2  sumwx  sumwx2  numEntries
```

'xlow' is the left edge and 'xhigh' is the right edge for a bin i. 'sumw' is the sum of weights per bin i $sumw^{bin,i} \sum_{k=1}^{N_{counts\ in\ bin\ i}} w_k^{bin\ i}$ and 'sumw2' is sum of weights squared $sumw2^{bin\ i} = \sum_{k=1}^{N_{counts\ in\ bin\ i}} (w_k^{bin\ i})^2$. To convert to differential cross section we divide by the bin widths, i.e. we calculate $\Delta x = xhigh - xlow$ and divide

* Florida State University; aalkadhim@fsu.edu.
† hprosper@fsu.edu

$\hat{\sigma}_i^{MC} = sumw/\Delta x$ and $sumw2/\Delta x^2$ and $sumwx/\Delta x$ and $sumwx2/Deltax^2$. Then the MC error for the cross section per bin is $\Delta_i^{MC} = \sqrt{sumw2/\Delta x^2}$

We can also write the effective count can be as

$$\frac{\sqrt{\sum_i w_i^2}}{\sum_i w_i} = \frac{1}{\sqrt{N_{eff}}}$$

So

$$N_{eff,\ i} = \left(\frac{\sum_i w_i}{\sqrt{\sum_i w_i^2}} \right)^2 = \left(\frac{\sum_i w_i}{\Delta_{i,MC}} \right)^2$$

Therefore the scale factor per bin is

$$K_i = \frac{\sum_i w_i}{\sum_i w_i^2} = \frac{\sum_i w_i}{\Delta_{i,MC}^2}$$

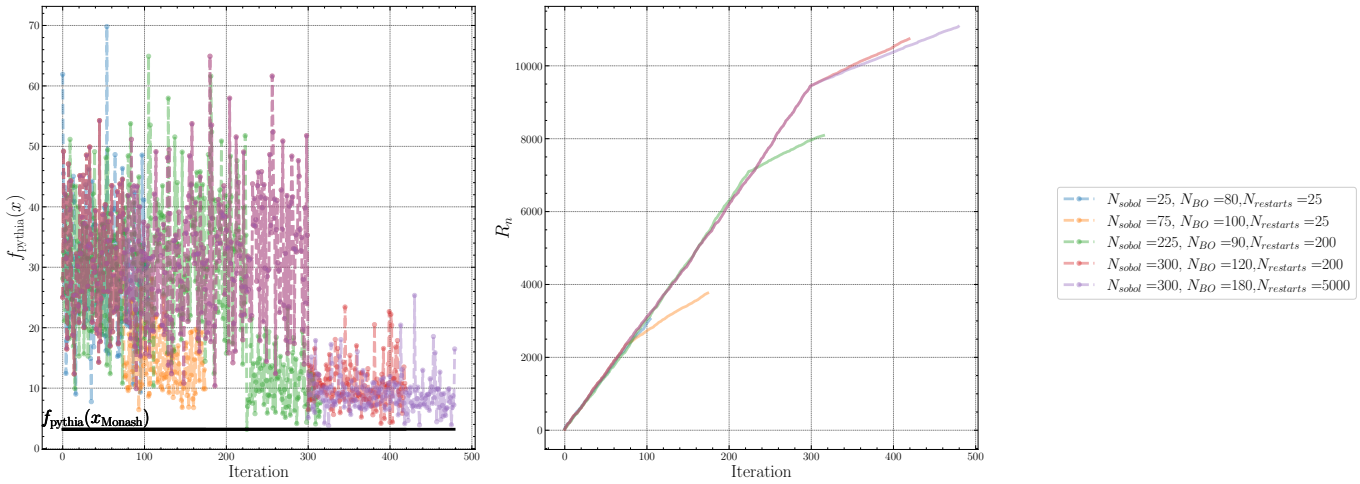

FIG. 1: Pythia objective functions (left) and cumulative regrets (right) for EI GPytorch results (Table **??**). The bold line on the left plot shows $f_{\text{pythia}}(x_{\text{Monash}}) = 3.2$.

B. EI with GPytorch visualization of objective funtions

C. qEI with BOTORCH on all histograms

For the following we use `BOTORCH` [1], which implements a noisy version of the expected improvement: in problems with stochastic observations, the surrogate posterior at any query point $\mathbf{x}$ is

$$f_{\mathcal{D}}(\mathbf{x}) \sim \mathcal{N}\big(\mu(\mathbf{x}),\, \sigma^2(\mathbf{x})\big).$$

Because the incumbent value $f^+ = \max_{x' \in \mathcal{D}} f(x')$ itself is uncertain, `BOTORCH` defines the noisy expected improvement (NEI) acquisition as

$$\begin{aligned} \alpha_{\text{NEI}}(\mathbf{x} \mid \mathcal{D}) &= \mathbb{E}_{\mathbf{f} \sim p(\cdot \mid \mathcal{D})}\big[\alpha_{\text{EI}}(\mathbf{x} \mid \mathbf{f})\big] \\ &= \int \alpha_{\text{EI}}(\mathbf{x} \mid \mathbf{f})\, p(\mathbf{f} \mid \mathcal{D})\, \mathrm{d}\mathbf{f}\,. \end{aligned} \tag{1}$$

Since no analytic form exists for α_{NEI} [2], it is approximated by Monte Carlo. We draw N independent samples

$$\{\xi^i_{\mathcal{D}}(\mathbf{x})\}_{i=1}^N \;\sim\; f_{\mathcal{D}}(\mathbf{x})$$

and compute

$$\hat{\alpha}_N(\mathbf{x} \mid \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \big[\max\{0,\; \xi^i_{\mathcal{D}}(\mathbf{x}) - f^+\}\big], \tag{2}$$

which directly estimates the expected gain over the noisy incumbent. Equivalently, for batch queries of size q, `BOTORCH` evaluates

$$\alpha_{\text{qNEI}}(\mathbf{X}; \mathcal{D}) = \mathbb{E}\big[\max(\max_{i \le q} f(\xi_i) - \max f(\xi_{\text{obs}}),\, 0) \mid \mathcal{D}\big], \tag{3}$$

by sampling joint realizations $\{\xi_i, \xi_{\text{obs}}\}$ from the posterior and computing the improvement over previous observations.

We refer to our BO results using Eq. 2 as "BO-qEI" in Table **??** and Fig. **??** to see the agreement of the best fit (`BO-qEI29`) with data. We see that with qNEI, our fits consistently lead to better objective functions than $f_{\text{pythia}}(x_{\text{Monash}})$ in a fewer number of iterations than with `GPytorch` using EI.

To be consistent with Sec. **??** and due to our insights of the most effective BaysOpt algorithm, for all our "BO-qEI" studies we use the Matern Kernel and `ExactGP` with `ExactMarginalLogLikelihood` with $N_{\text{events}} = 250,000$ events simulated per parameter point.

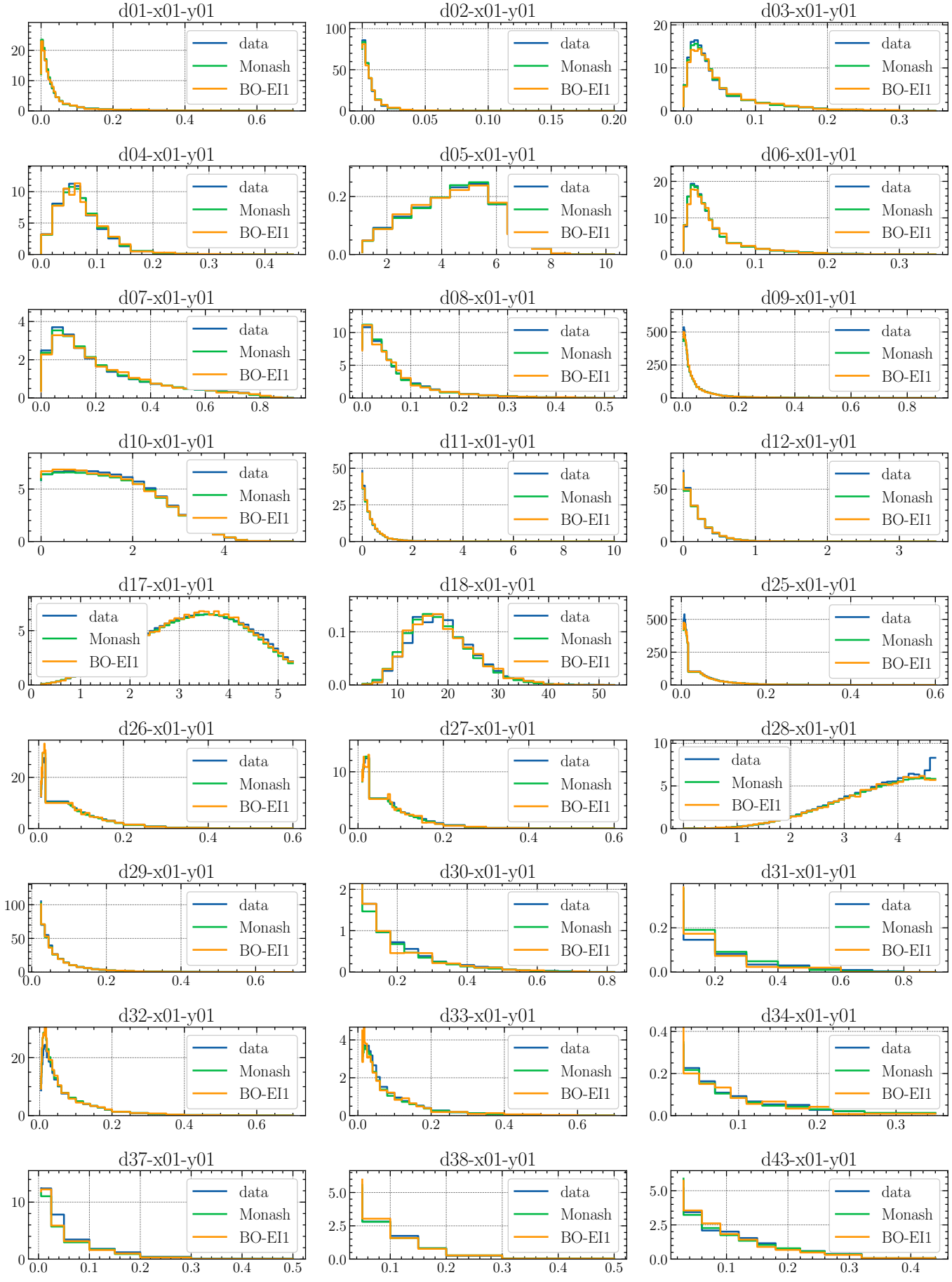

FIG. 2: The BO EI tune results using `GPytorch` for label "BO-EI1" in Table **??**

D. BayesOpt vs Professor Table

Feature	**GP Surrogate (BayesOpt)**	**Polynomial Emulator (Professor)**
Model type	Nonparametric—each new data point updates the entire surface	Parametric—pre-chosen basis (often low-order polynomials)
Flexibility	Can model arbitrary smooth, non-linear interactions and noise	Limited to the expressivity of the chosen polynomial form
Uncertainty quantification	Built-in posterior variance → drives exploration vs. exploitation	Typically none (or uses ad-hoc Hessian approximations)
Evaluation cost	$O(n)$ per prediction	$O(p)$ per evaluation (p = number of polynomial terms)
Empirical robustness	Good at capturing unexpected shapes or heavy-tailed behavior	Often breaks down for distributions $\sim 1/x$ or other singular shapes

TABLE I: Comparison of Gaussian-process surrogates used in Bayesian optimization versus polynomial emulators in the Professor approach.

TABLE II: EI with `GPytorch` Results on a subset of the histograms and Eq. 2. We refer to the best fit using EI, namely the bolded third row in this table, as BO-EI. $N_{restarts} = 25$.

	N_{BO}	N_{sobol}	E_{opt}	f_{best}	aLund	bLund	ProbStoUD	probQQtoQ	alphaSvalue	pTmin
Monash				**581.493**	**0.68**	**0.98**	**0.217**	**0.081**	**0.1365**	**0.500**
	70	25	50	780.736	1.216	1.569	0.898	0.116	0.143	0.865
	140	**25**	**50**	**426.708**	**1.693**	**1.458**	**0.707**	**0.221**	**0.139**	**0.513**
	280	25	50	881.562	0.845	0.623	0.680	0.428	0.141	1.130
	400	25	100	538.250	0.551	0.567	0.517	0.184	0.136	0.949

E. Pythia Results with removing histograms

1. *Pythia BayesOpt Tune Results (EI with `GPytorch` on a subset of the histograms)*

It is very tempting to manually remove some histograms from the fit. In all of the follwoing results, we remove the following histograms from the sum in the objective function:

- `d35-x01-y01`
- `d36-x01-y01`
- `d39-x01-y01`
- `d40-x01-y01`

One could elect to remove these histograms from observing that they do not describe the data well. In the following, we also use the following objective function:

$$f_{\text{pythia}}(X;\mathbf{x}) = \sum_{\text{histograms}} \sum_{i=1}^{N_{\text{bins}}} \left(\frac{D_i - T_i(\mathbf{x})}{\delta_i} \right)^2 , \tag{2}$$

Another reason to choose the objective function in Eq. 2 is that otherwise we cannot correctly interpret the minos errors as confidence intervals because the objective function doesn't have a χ^2 distribution. On the other hand using Eq. 2 we can interpret this as a χ^2 function for which the usual rules can be applied to derive the errors.

We note, however, that taking the square root and normalizing by the sum of bins in each histogram does not necessarily lead to better results. The reason we have bad results here is that we are removing 4 histograms from the sum in the objective function.

We refer to the best fit using EI, namely the bolded third row in table **??**, as `BO-EI`.

FIG. 3: `GPytorch` EI results on a subset of the histograms and Eq. 2 with $N_{BO} = 140, N_{sobol} = 25$, corresponding to the bolded row in table II

2. *Doing Better:* `BOTORCH` *qEI (*`BOTORCH` *results on a subset of the histograms)*

Using the same `Pythia8` objective function in 2 we now consider using qNoisyExpectedImprovement with `BOTORCH`. In all the following we use `ExactGP` with `ExactMarginalLogLikelihood`. See table III and Fig. 4.

	N_{BO}	N_{sobol}	f_{best}	aLund	bLund	ProbStoUD	probQQtoQ	alphaSvalue	pTmin
Monash			**581.493**	**0.68**	**0.98**	**0.217**	**0.081**	**0.1365**	**0.500**
	20	15	856.565	2.000	2.000	0.868	0.167	0.143	0.528
	40	15	290.901	1.207	1.728	0.980	0.000	0.136	0.565
	60	15	624.862	2.000	1.578	0.000	0.555	0.141	0.100
	80	25	479.285	1.576	1.764	0.279	0.312	0.142	0.440
	90	25	257.061	1.158	1.911	1.000	0.000	0.136	0.100

TABLE III: `BOTORCH` qEI results on a subset of the histograms and Eq. 2

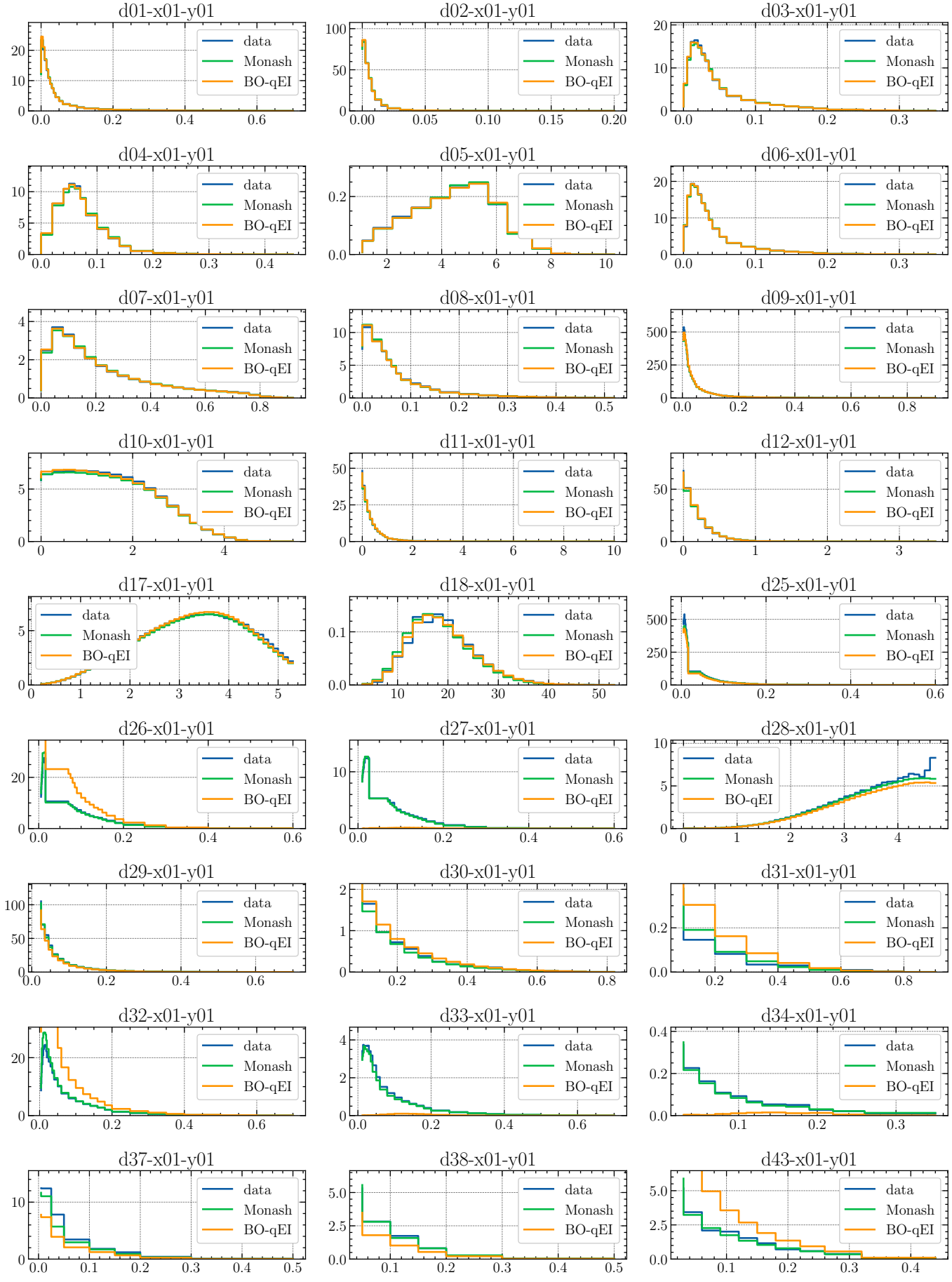

FIG. 4: BOTorch results on a subset of the histograms and Eq. 2 with $N_{BO} = 90, N_{sobol} = 25$

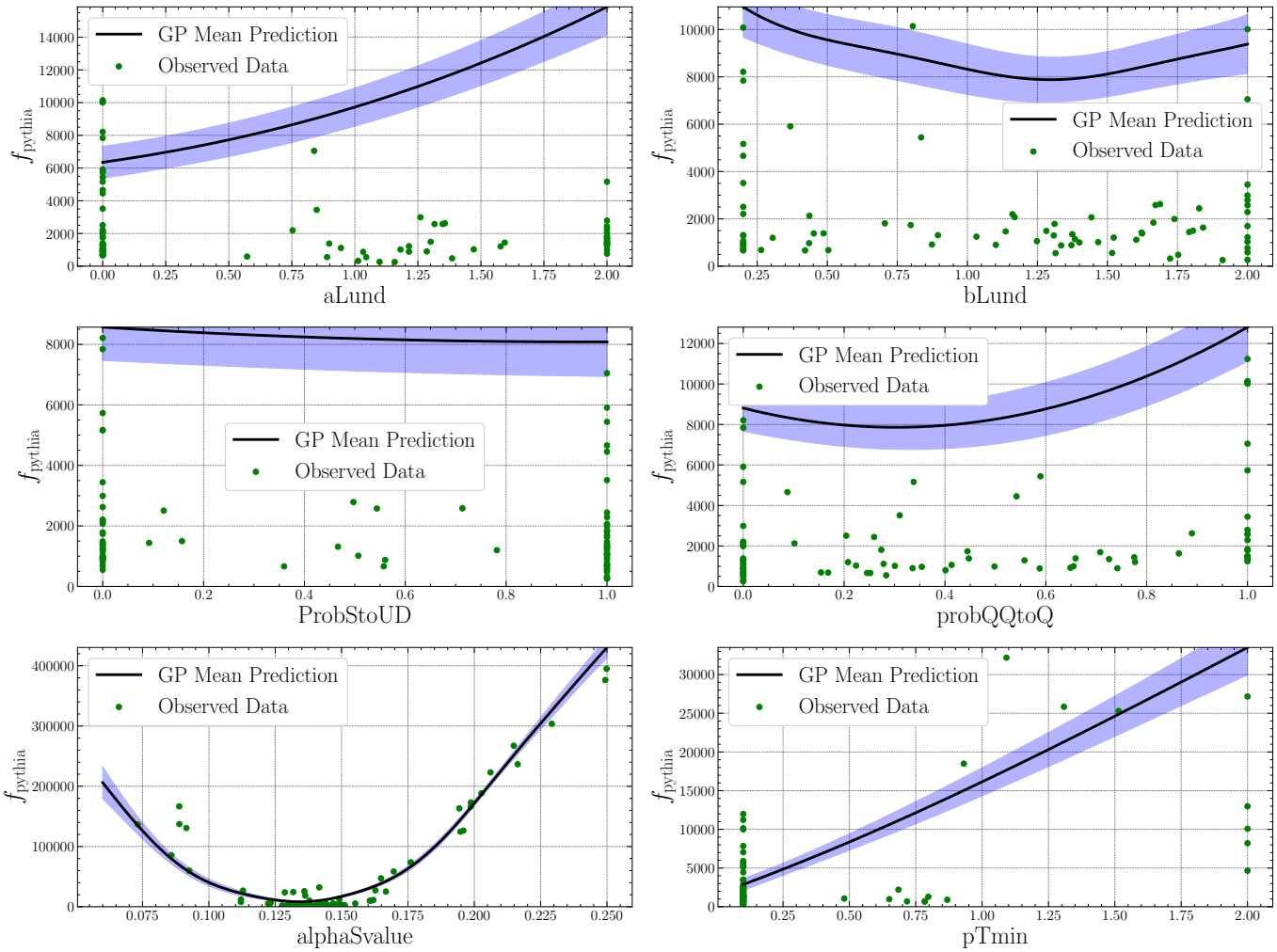

FIG. 5: BOTorch qEI "sausage plots" results on a subset of the histograms and Eq. 2 for $N_{BO} = 90, N_{sobol} = 25$

F. Analysis on toy problems

For all our results that we present, we only use `GPytorch` [3] for the surrogate model, and we implement the BO algorithm from scratch as described in previous sections. This gives us complete control and understanding of all the inner workings of the BO algorithm and understand the effect of all the hyperparameters and algorithm choices identified in sec. **??**. In all our toy experiments we use `gpytorch.models.ExactGP` with `gpytorch.likelihoods.GaussianLikelihood`, that is, the likelihood has a noise standard deviation term σ^2_{noise} as described in Sec. **??**.

In all of our toy problems we use the Matern Kernel with $\nu = 5/2$. Other kernel functions were studied such as RBF but yielded suboptimal results. Other lesser known kernel functions were also experimented with including RQ, Cosine, Polynomial and Spectral Mixture kernel functions. The choice of using Matern Kernel is well motivated in literature CITE, where it is noted that $k_{M5/2}$ is the most effective and popular choice. We also only use the expected improvement (EI) acquisition function as in Eq. **??** in all of our studies (except in Sec. I E 2). Other acquisition functions were studied such as the upper confidence bound (UCB), but EI produced by far the best results.

1. Toy problems we consider

An objective function $f(\mathbf{x})$ is used to quantify the distance between a parameter point $\mathbf{x}$ and its minimum. Throughout this paper, we present results in six dimensions $d = 6$, that is, $\mathbf{x} \in \mathbb{R}^6$. See Sec. **??** where we review why BO only works in $d \leq 20$, therefore one has to limit the number of parameters studied up that cutoff where BO works. Dimensions up to $d \leq 14$ were studied, but no significant difference in the results was observed. Furthermore $d = 6$ is chosen since our desired Pythia tune is in $d = 6$, which will be motivated in Sec. **??**.

In order to study Bayesian optimization use in a similar problem as ours, as in it has the same dimension size, we have conducted the following study on a toy objective function in order to study the effect of different algorithms and hyperparameters. This objective function has a minimum, which is chosen by construction

$$\mathbf{x}^* = \mathbf{x}_{\text{Monash}}, \tag{4}$$

where $\mathbf{x}_{\text{Monash}}$ is bolded in Table **??** (we will discuss this table in more detail in the coming sections). A quadratic form between two parameter vectors A and B can be written as

$$QF(A, B) = \sum_{i=1}^{|A|} \alpha_i \ (A_i - B_i)^2$$

where $\alpha_i = (1.2)^i$ for example, but can be any coefficient.

Therefore we can form an objective function that, by construction, has a one minimum at $\mathbf{x}_{\text{Monash}}$ as

$$f_{1\text{min}}(\mathbf{x}) = QF(\mathbf{x}, \mathbf{x}_{\text{Monash}}) \tag{5}$$

That is

$$\mathbf{x}_{\text{Monash}} = \text{argmin}_{\mathbf{x}} f_{1\text{min}}(\mathbf{x}) \tag{6}$$

An objective function that has multiple minima, say 3, can by analogy be constructed by introducing 2 new minima in addition to Eq. 4.

$$\mathbf{x}_1 = 0.5 \ \mathbf{x}_{\text{Monash}}$$

$$\mathbf{x}_2 = 1.5 \ \mathbf{x}_{\text{Monash}}$$

and

$$f_{3\text{min}}(\mathbf{x}) = \sqrt{QF(\mathbf{x}, \mathbf{x_0}) \times (QF(\mathbf{x}, \mathbf{x_1}) + 1) \times (QF(\mathbf{x}, \mathbf{x_2}) + 2)} \tag{7}$$

2. *Sausage plots for toy problem*

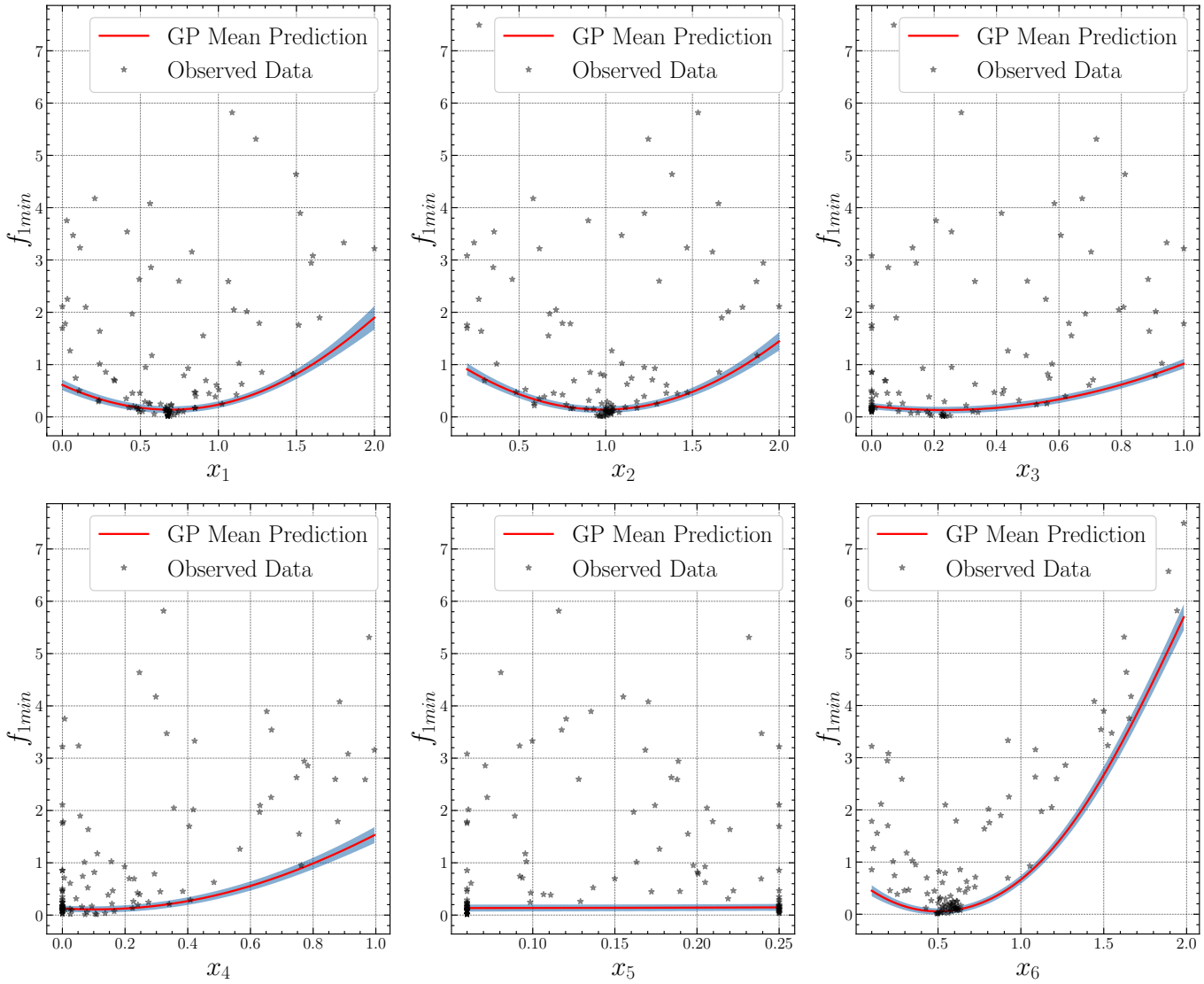

FIG. 6: Example set of "sausage plots" for $d = 6$ f_{1min} toy problem, with $N_{BO} = 70, N_{sobol} = 25, N_{restarts} = 25$, and Adam-R. EI acquisition function and Matern kernel are used as always.

It is important to understand what are sometimes called "sausage plots" for a given BO run such as the one we produce in Fig. 6. The black points are the sampled parameter points and their resulting objective function, the red line is the posterior mean μ function of the GP, which is used as the surrogate model of the objective function. At each iteration, the acquisition function is maximized in order to determine where to sample the data next (i.e. Eq. **??**). The shaded regions are the mean plus and minus the GP variance σ evaluated at that point.

3. *The effect of sampling algorithm*

Sobol vs uniform MC plot and more on sobol or hypercube sampling. Take $\mathbf{x_1}, \mathbf{x_2} \in [0.0, 2.0]$

4. *Values we consider and a first look at the objective and acquisition functions*

For our toy problem studies we study all possible combinations of the following BO algorithm hyperparameters:

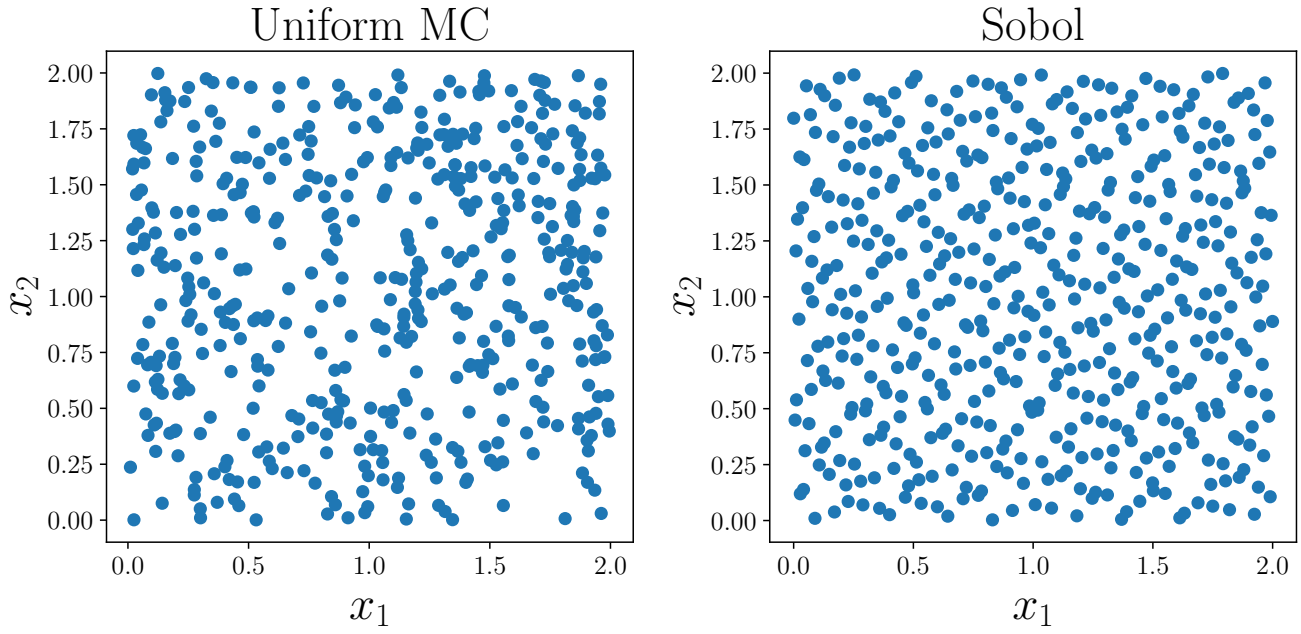

FIG. 7: uniform vs sobol sampling.

- Number of sobol "training" points $N_{\text{sobol}} \in [25, 200]$
- Number of BO optimization steps $N_{\text{BO}} \in [30, 100, 300]$
- Number of epochs in optimizing the acquisition function $E_{\text{acq}} \in [10, 50, 100]$
- Number of restarts in optimizing the acquisition function $N_{\text{restarts}} \in [5, 25]$
- The algorithm used in optimizing the acquisition function $\mathcal{A} \in$ [`Adam-R`, `Adam-NR`, `SLSQP`]

We perform an experiment for every possible combination of the values listed above, resulting in 108 different experiments. We repeat the whole analysis for $f_{1\text{min}}$ and $f_{3\text{min}}$ to observe the effect of multimodal functions. For more details on these, see the following sections.

Function	N_{sobol}	N_{BO}	E_{acq}	N_{restarts}	$\mathcal{A}$	f_{best}
$f_{1\text{min}}$	200	300	50	5	Adam-R	0.002
$f_{3\text{min}}$	25	300	10	5	Adam-R	0.179

TABLE IV: The best BO hyperparameter configurations for all 108 experiments done in each of $f_{1\text{min}}$ and $f_{3\text{min}}$.

Table IV shows the best BO hyperparameter configurations for all 108 experiments done in each of $f_{1\text{min}}$ and $f_{3\text{min}}$. From this we can immediately see that maximizing N_{BO} and using `Adam-R` clearly leads to the best results. It is difficult to draw general conclusions just by looking at this table. In the following section we look at each of the effect of each of the identified BO hyperparameters in more detail.

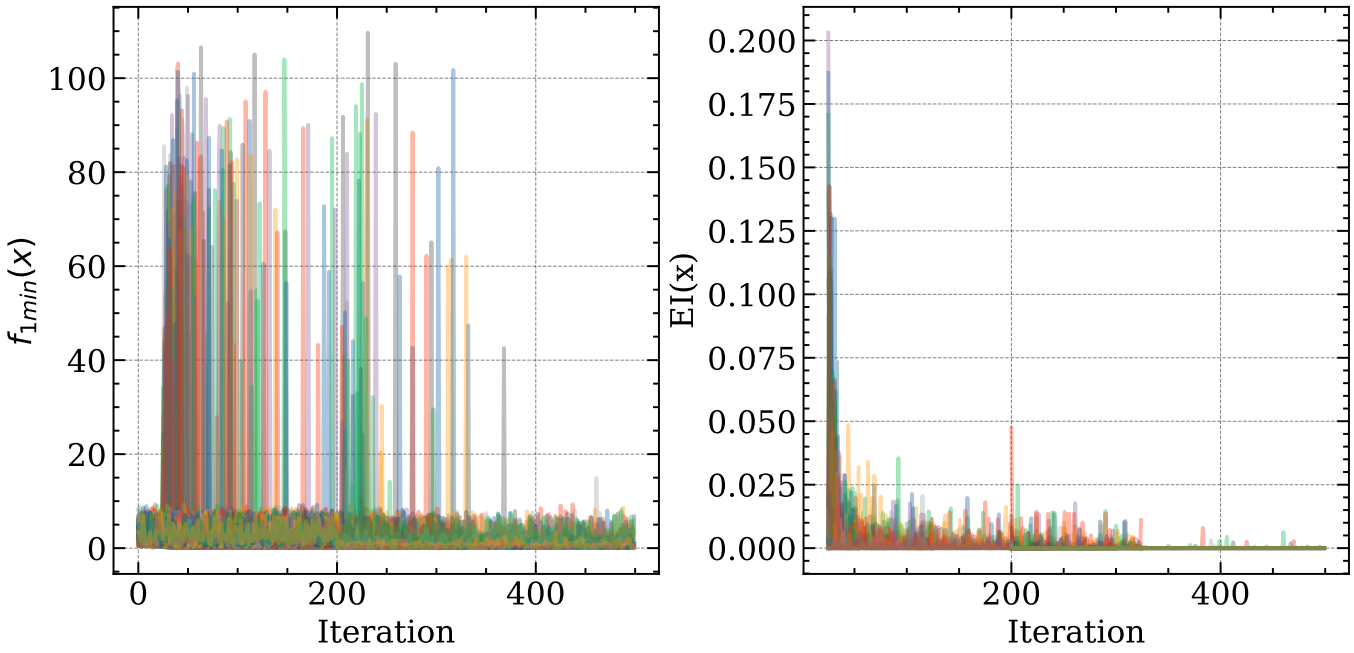

FIG. 8: f1min all

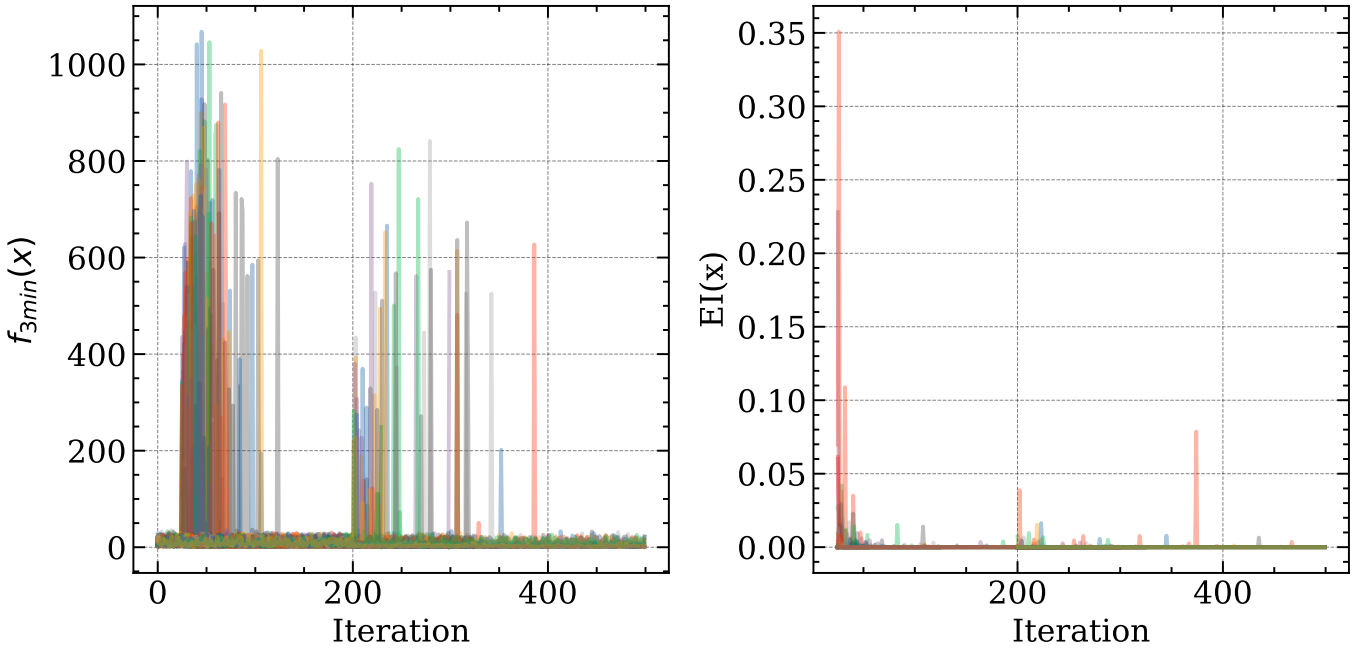

FIG. 9: f3min all

5. *The effect of optimization algorithm*

We observe from Fig. 10 that the choice of the algorithm $\mathcal{A}$ used to optimize the acquisition function as arguably the most important choice in BO: if one is not careful in this choice all the results could be zigzagging nonesense that does not converge. We explore 3 optimization algorithms:

- Adam without clipping or restarts, we denote this as `Adam-NR`
- Adam with clipping and restarts, we denote this as `Adam-R`

- `SLSQP`

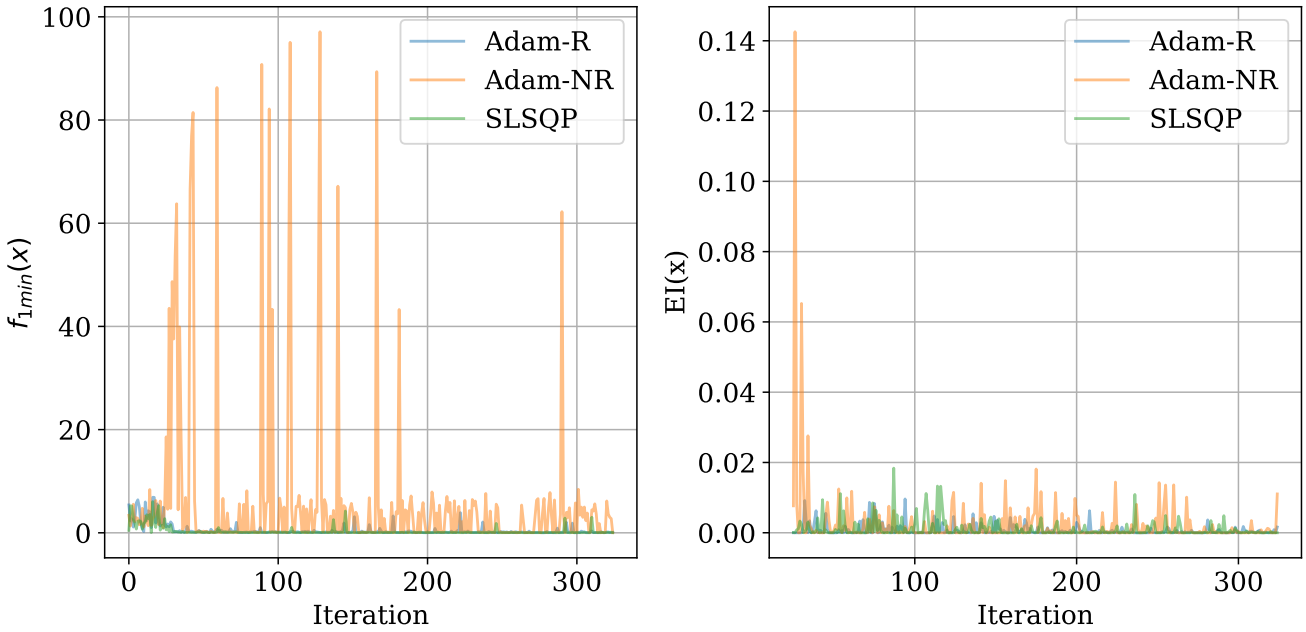

FIG. 10: $N_{sobol} = 25$, $N_{BO} = 300$, $I_{opt} = 100$, $N_{restarts} = 25$

From these results, we can clearly see that `Adam-NR` jumps across the objective function with a huge dynamic range, indicating that it may not serve as a robust optimization approach. however, based on this evidence alone, we cannot determine which algorithm truly performs best. To gain deeper insight, it is necessary to examine the regret metrics.

Since we are dealing with a minimization and not maximization problem, the simple regret in our case is:

$$r_n = f\left(x_n^+\right) - f\left(x^*\right) \tag{8}$$

Where $f(x^*)$ is the true global minimum, which, by construction $f(x^*) = 0$ and $f(x_n^+)$ is the lowest objective value up to iteration n. Another formulation of the regret, which we use is simply

$$r_n = f\left(x_n\right) - f\left(x^*\right) \tag{9}$$

It is important to look at the simple regret as a positive value, and the problem now turns into (cumulative) **regret minimization**. We now use figure of merit $R_n = \sum_n r_n$ with r_n defined in Eq. 9 to compare the behaviors of different algorithms.

We quantify the performance of different optimization algorithms by looking at the cumulative regret curves in Fig. 11. Interestingly, the regret functions for f_{1min} looks nearly identical to the ones for f_{3min} except for a scaling coefficient. **We can also confirm our hypothesis** that `Adam-NR` is the worst algorithm by looking at Fig. 11 (left): after $N_{\text{sobol}} = 200$, `Adam-R` and `SLSQP` improve in performance by reducing the cumulative regret, as expected, (because now is where the BO algorithm starts and chooses next points to sample intelligently) whereas `Adam-NR` worsens in performance. Therefore, because `SLSQP`'s performance closely matches that of `Adam-R`, yet `Adam-R` ultimately demonstrates superior results, we choose `Adam-R` for the subsequent studies.

In Table **??** we present of all our BO experiments with `Adam-R` chosen as the optimization algorithm for $f_{1\text{min}}$

But many questions remain in practically fitting them. For example, how many points or sample size one should use for the fit (i.e. how much does $N_{\text{sobol}} = |\mathcal{D}_{\text{sobol}}|$ matter?)

We first note that it is not immediately obvious to me by looking at for example Fig. 6 that more sobol points clearly lead to much better performance, because the points seem randomly scattered, hence too complicated for the GP to fit. We can see in Fig. 12 that increasing N_{sobol} clearly helps in performance with all other factors held fixed and $N_{\text{BO}} = 300$. However, from Fig. 12 one cannot tell exactly how much N_{sobol} matters, since the slopes of R_n look quite similar for the two curves *after* the start of the BO algorithm. We therefore need to plot the performance of N_{sobol} at the start of the BO algorithm, we refer to this as "BO iteration" if Fig. 13 as opposed to iteration in Fig 12.

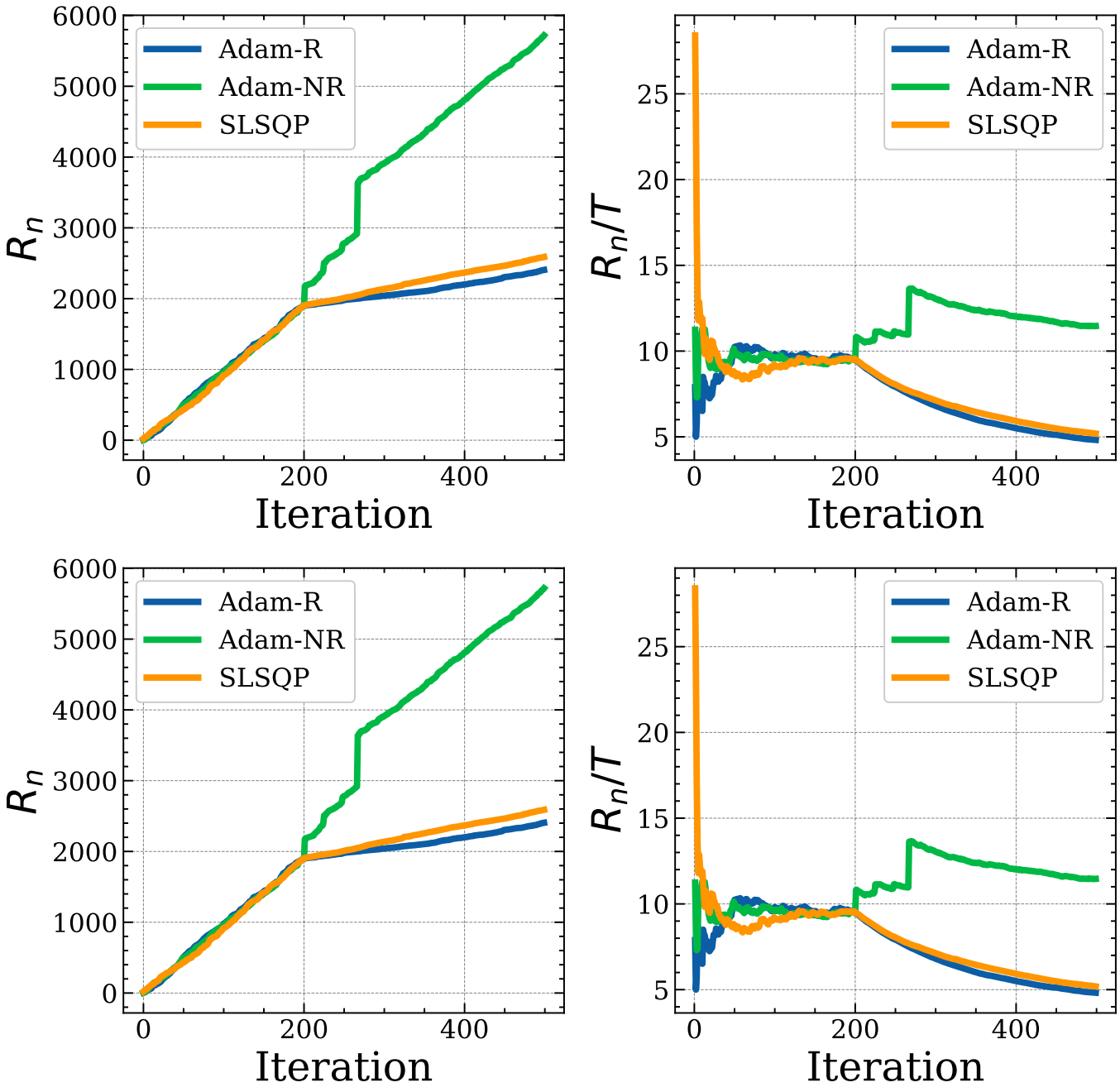

FIG. 11: Cumulative regret curves for different optimization algorithms. Top: $f_{1\text{min}}$, bottom: $f_{3\text{min}}$. To isolate the effect of the optimization algorithm the following hyperparameters were set: $N_{\text{sobol}} = 25$, $N_{\text{BO}} = 300$, $I_{\text{opt}} = 100$, $N_{\text{restarts}} = 25$.

In addition to plotting starting at the BO iteration, a more dense exploration of N_{sobol} is explored in Fig. 13 for different dimensions. We also plot the log of the objective function to explore the variability and convergenece behavior. We see that in $d = 1$, Fig. 13 shows that N_{sobol} clearly has a big impact and leads to more stable convergence.

In $d = 2$ there is also no surprises as shown in Fig. 14 and no surprises in $d = 6$ in Fig. 15. However, we do see **the curse of dimensionality** at work, as we increase the number of dimensions, the data becomes much more sparse requiring exponentially more points to achieve the same results.

For a multimodal objective function $f_{3\text{min}}$ in $d = 6$, the effect is illustrated in Fig. 16, where it is also seen that $f_{3\text{min}}$ does not reach as low minima as in $f_{1\text{min}}$. In $d = 6$, going from unimodal ($f_{1\text{min}}$) to multimodal ($f_{3\text{min}}$), selecting a

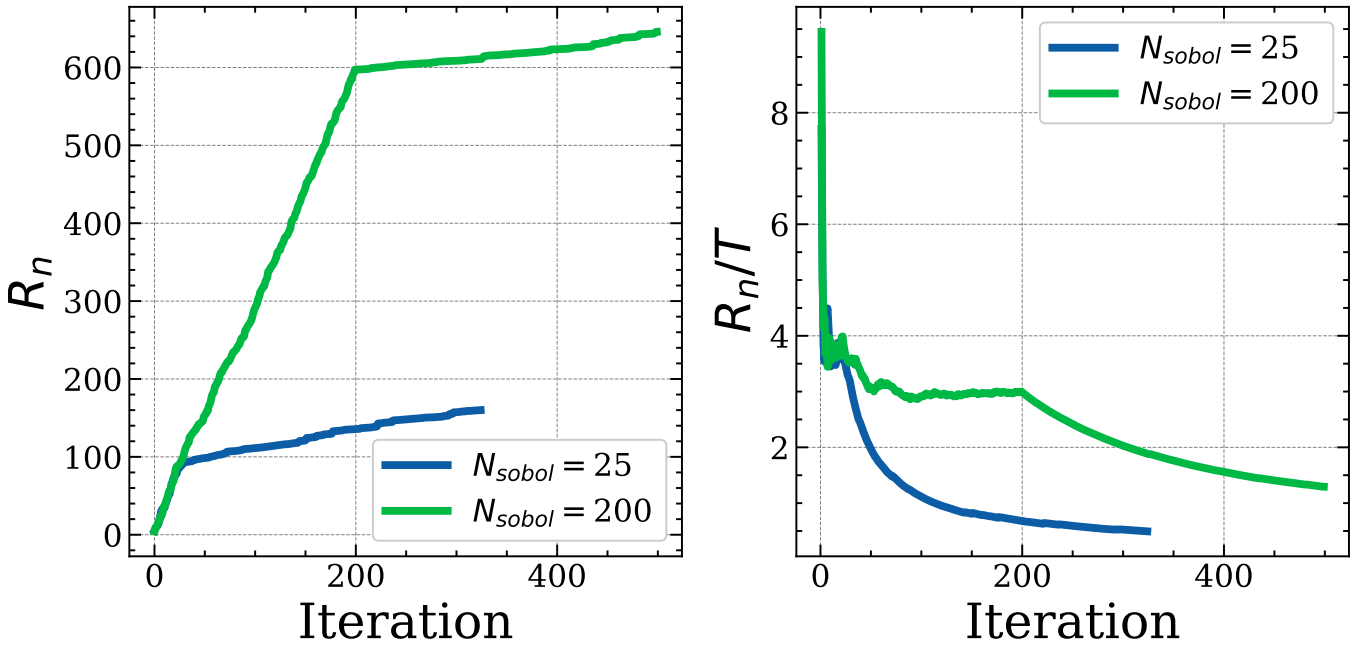

FIG. 12: Cumulative regret curves showing the effect of increasing N_{sobol} on the performance of BO for $f_{1\text{min}}$

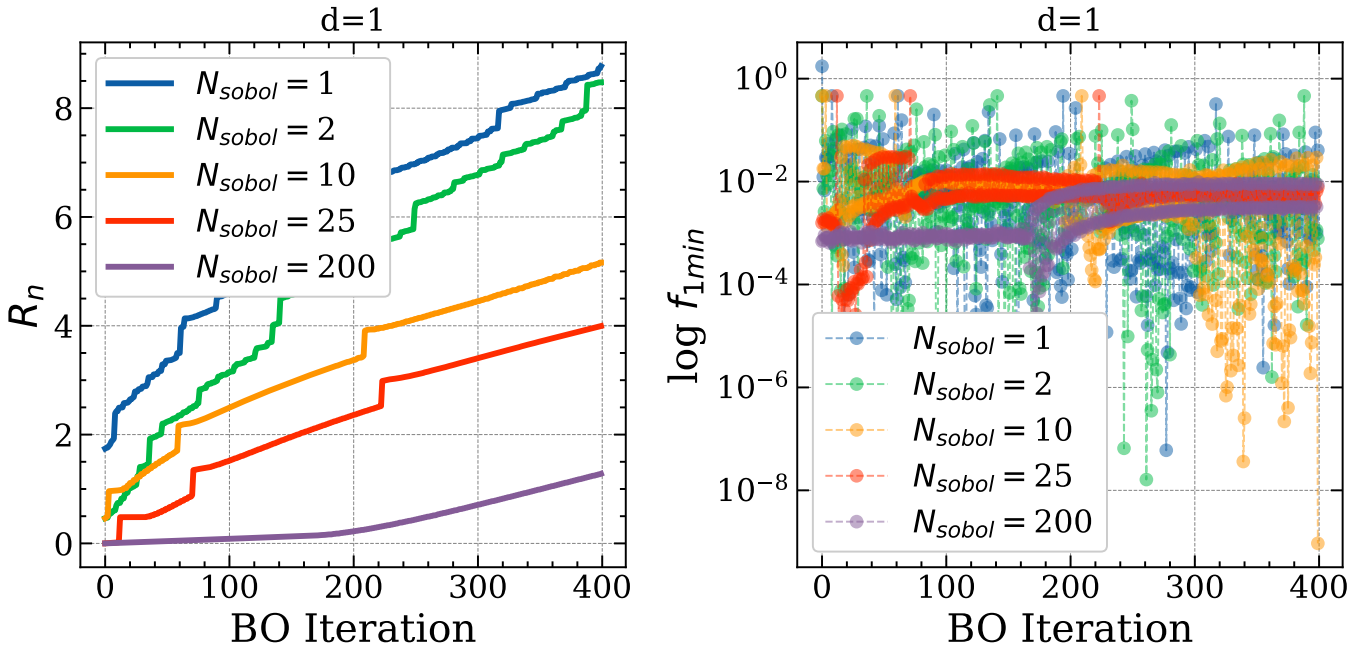

FIG. 13: The effect of N_{sobol} for $f_{1\text{min}}$ in $d = 1$. For all that is shown the following is fixed:

small N_{sobol} starts to have less of a negative impact. This can be seen in the sharp rise in R_n in Fig. 15 as compared with Fig. 16 for $N_{\text{sobol}} \leq 25$.

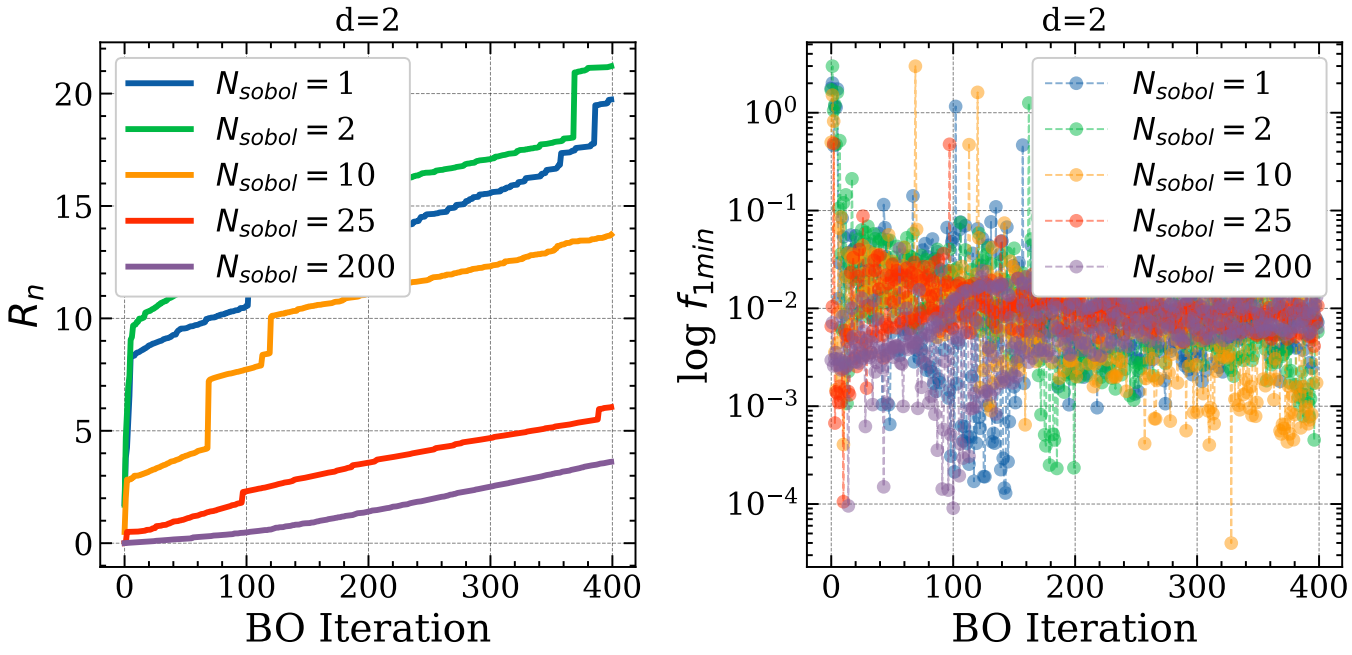

FIG. 14: The effect of N_{sobol} for $f_{1\text{min}}$ in $d = 1$. For all that is shown the following is fixed:

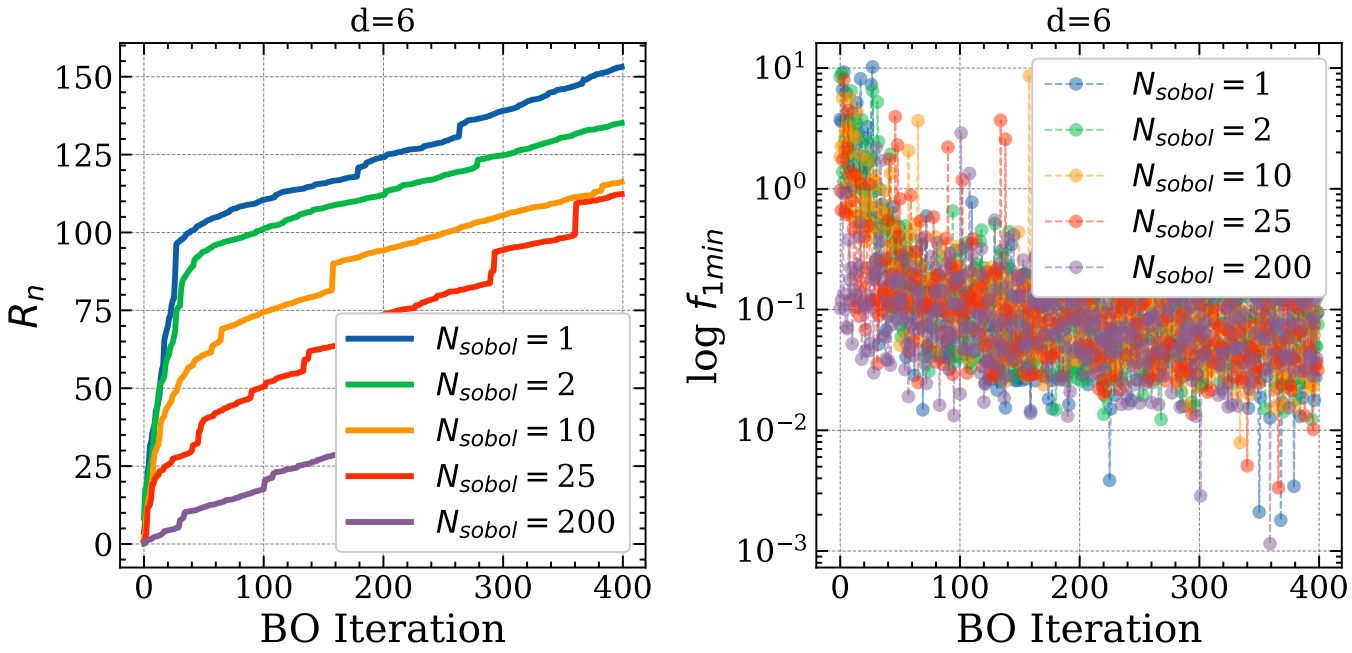

FIG. 15: The effect of N_{sobol} for $f_{1\text{min}}$ in $d = 6$. For all that is shown the following is fixed:

6. *The effect of the number of training epochs in fitting Kernel Hyperparameters*

As described in Sec. **??**, BO starts with fitting the kernel hyperparameters. In `GPytorch`, we tune the hyperparameters by maximizing the marginal log likelihood (MLL) (i.e. treating -MLL as the loss and minimizing it with gradient descent)

$$\text{MLL} = p_f(\mathbf{y} \mid \mathbf{x}) = \int p(\mathbf{y} \mid f(\mathbf{x}))p(f(\mathbf{x}) \mid \mathbf{x})df, \tag{10}$$

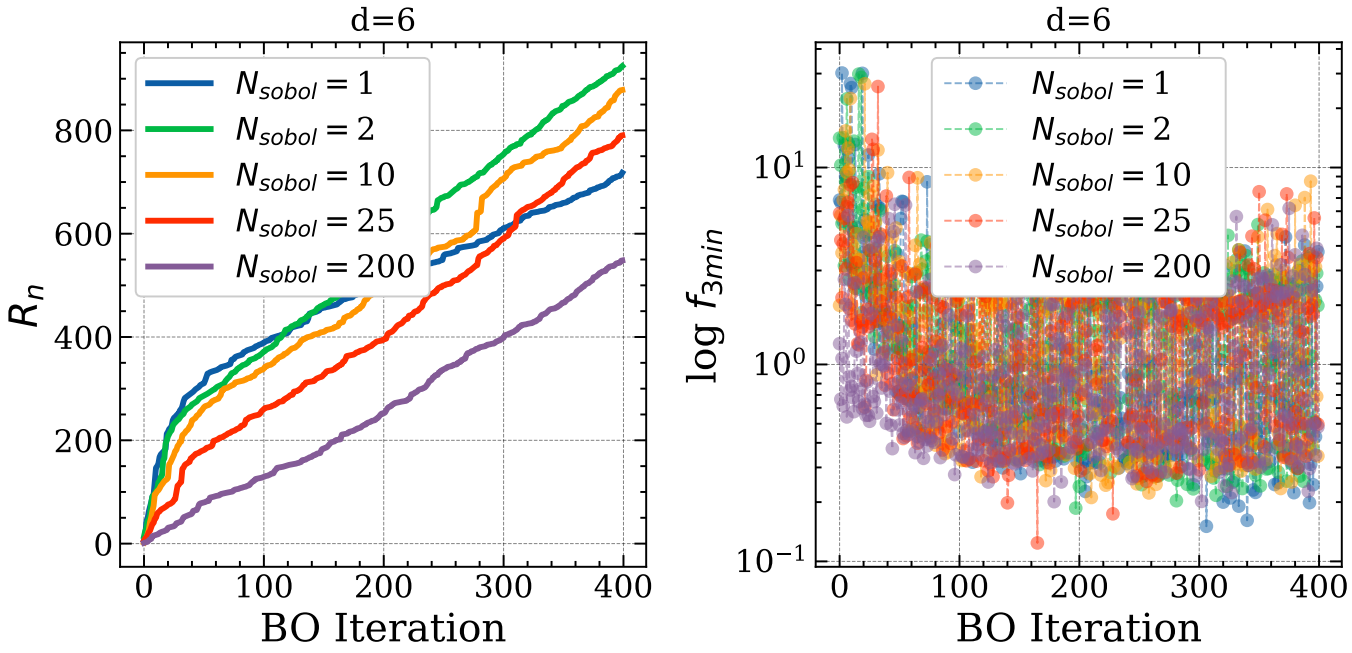

FIG. 16: The effect of N_{sobol} for $f_{3\text{min}}$ in $d = 6$. For all that is shown the following is fixed:

which, for the GP this equates to

$$p(\mathbf{y} \mid \mathbf{X}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{m}, \mathbf{K} + \sigma^2 \mathbf{I}\right) \tag{11}$$

A question remains: how many epochs should one train for? We denote the number of epochs in fitting $\hat{\phi}_{\text{MLE}}$ as E_ϕ In all of our experiments it was noted that the loss is a smooth and decreasing with the number of epochs, reaching a plateau. We see no evidence of overfitting or odd behaviour such as double descent, indicating that one can increase training epochs as desired. Therefore, a good number of training epochs was observed to be around 100. This number was set to 100 in all subsequent experiments.

Another question we consider is "should the kernel hyperparameters be optimized only once and for all on $\mathcal{D}_{\text{sobol}}$, or should they be re-optimized at every iteration in BO?" We expect that the confidence region gets narrower in regions where there is more data (that is, where there are more points sampled). However, if the kernel hyperparameters are not optimized at each BO iteration step, the confidence region of the predicted mean could potentially stay constant in width! We therefore recommend that the kernel hyperparameters are re-optimized on the marginal log-likelihood in each BO iteration step. In order to keep with the traditional Bayesian approach, *we did not re-optimize the kernel hyperparameters at each BO iteration. Instead, we optimized it once in the beginning on* $\mathcal{D}_{sobol}$.

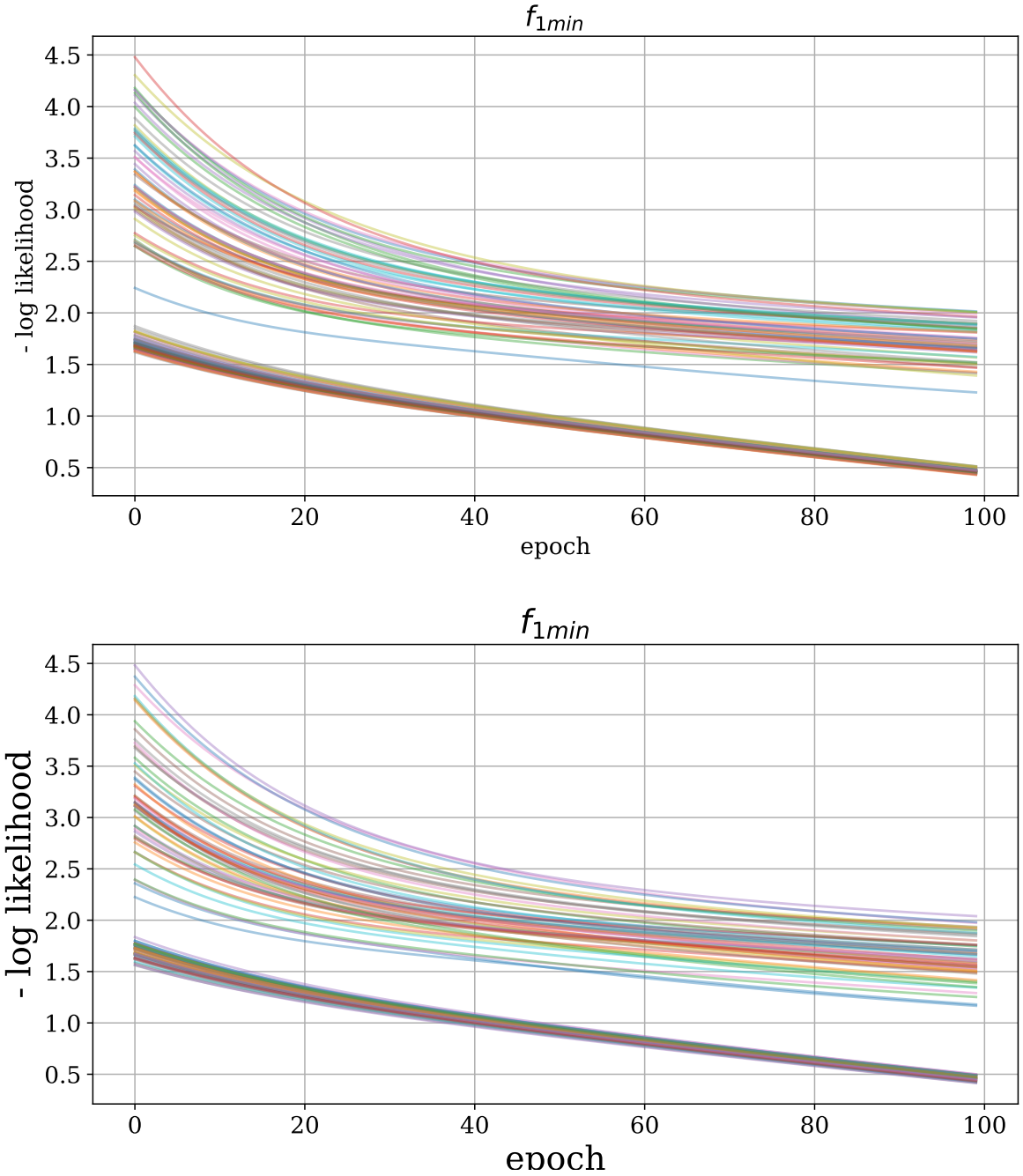

FIG. 17: Loss curves to fit $\hat{\phi}_{\text{MLE}}$. Top: $f_{1\text{min}}$, bottom: $f_{3\text{min}}$.

7. *The effect of $N_{restarts}$*

Fixing our algorithm to `Adam-R`, we use Cumulative regret curves to show the effect of the number of restarts in multi-start optimization in optimizing the acquisition function in 18.

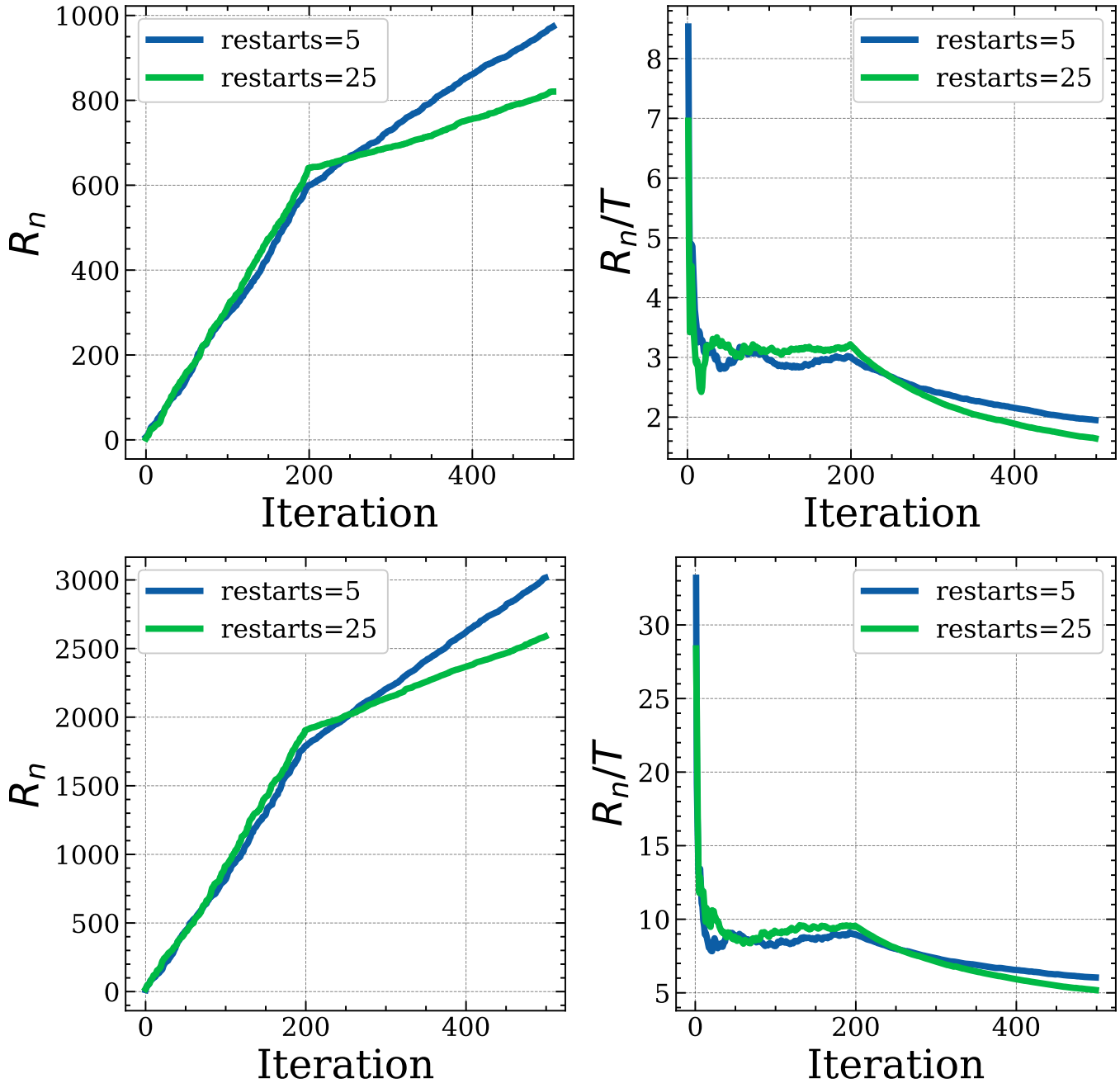

FIG. 18: Cumulative regret curves showing the effect of the number of restarts in multi-start optimization of the acquisition function. Top: $f_{1\text{min}}$, bottom: $f_{3\text{min}}$.

8. *Effect of Acquisition-Function Epochs*

See Fig. 19, which shows that E_{acq} has a very minor—almost negligible—effect on performance. **Surprisingly, lower E_{acq} leads to slightly better results.**

G. Potential for stopping criterion

In practice, Bayesian optimization proceeds by selecting a sequence of query points $\{x_n\}_{n=1}^{N}$ until a fixed evaluation budget M is exhausted. The choice of M that yields efficient convergence depends sensitively on the unknown objective

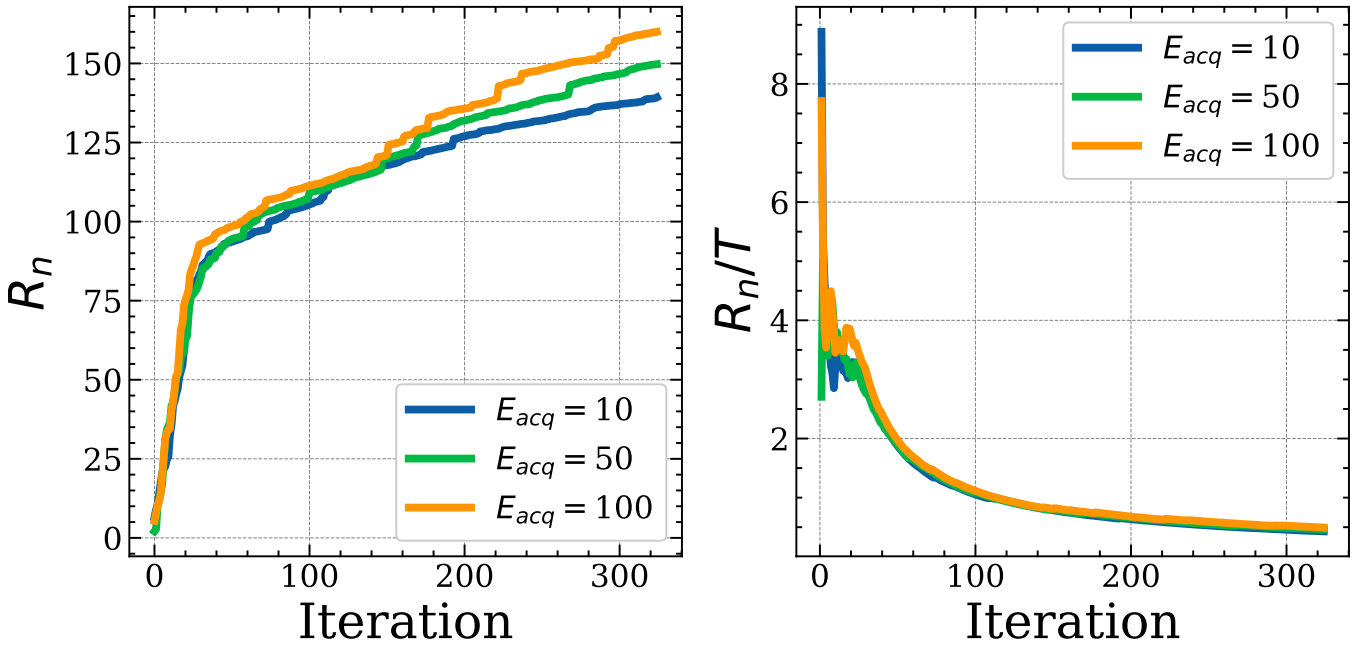

FIG. 19: The effect of the number of epochs in optimizing the acquisition function (per BO iteration), E_{acq}, for $f_{1\text{min}}$.

landscape, the surrogate model's fidelity, the initial dataset, and the particular acquisition rule in use. Alternatively, one might terminate as soon as the observed objective falls below some predefined threshold, but this requires prior knowledge of the global minimum—information we generally lack.

To sidestep these difficulties, we turn to **regret** as a performance metric. At iteration n, the *instantaneous regret* is

$$r_n = f(x^*) - f\big(x_n^+\big),$$

where $x^* = \arg\min_x f(x)$ denotes the true (unknown) minimizer and $x_n^+ = \arg\min_{i \le n} f(x_i)$ is the best point found so far. Summing over iterations gives the *cumulative regret*

$$R_n = \sum_{i=1}^{n} \big(f(x^*) - f(x_i^+)\big).$$

An algorithm is termed *no-regret* if

$$\lim_{n\to\infty} \frac{R_n}{n} = 0,$$

i.e. its average regret vanishes and R_n grows sublinearly with n. Sublinear growth of R_n guarantees that, on average, the algorithm spends an increasingly large fraction of its budget sampling near the global optimum.

Although we cannot compute $f(x^*)$ in practice, upper bounds on R_n provide rigorous convergence rates. In particular, one can show that the Expected Improvement (EI) acquisition yields $R_n = \mathcal{O}(n^{-\frac{1}{2}} \log n)$ (or other sublinear rates, depending on kernel smoothness), demonstrating that EI converges efficiently to the global minimum.

Since $\alpha^{\text{EI}}(x) \ge 0$, [4] consider a stopping criterion for EI based on a small predefined constant $\kappa > 0$, such that the algorithm is terminated if $\alpha_t^{EI}(x_t) \ge \kappa$ is violated. They consider $\kappa = 10^{-4}$ and $\kappa = 10^{-9}$ and observe that they lead to similar outcomes. However, we note that the choice of κ is problem-dependent. Furthermore, in practice $\alpha^{\text{EI}}(x)$ can be a small negative value due to numerical precision **We shed light on some of these heuristics with empirical experiments. One of the things we study is how big must this κ be? how problem-dependent**

is it? Is there a way to predict its minimum given a particular problem?

---

[1] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization (2020), arXiv:1910.06403.

[2] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, Constrained Bayesian Optimization with Noisy Experiments (2018), arXiv:1706.07094 [stat].

[3] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, GPyTorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration, in *Advances in neural information processing systems* (2018).

[4] H. Ishibashi, M. Karasuyama, I. Takeuchi, and H. Hino, enA stopping criterion for Bayesian optimization by the gap of expected minimum simple regrets, in en*Proceedings of The 26th International Conference on Artificial Intelligence and Statistics* (PMLR, 2023) pp. 6463–6497, iSSN: 2640-3498.