Inference on the Champagne Model using a Gaussian Process

TODO

- Change outputs

Setting up the Champagne Model

Imports

```
import pandas as pd
import numpy as np
from typing import Any
import matplotlib.pyplot as plt

from scipy.stats import qmc
from scipy.stats import norm

import tensorflow as tf
import tensorflow_probability as tfp
from tensorflow_probability.python.distributions import normal

tfb = tfp.bijectors
tfd = tfp.distributions
tfk = tfp.math.psd_kernels
tfp_acq = tfp.experimental.bayesopt.acquisition

gpu_devices = tf.config.experimental.list_physical_devices("GPU")
```

```
for device in gpu_devices:
    tf.config.experimental.set_memory_growth(device, True)
```

```
2024-05-06 17:03:30.934874: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorF
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with
2024-05-06 17:03:31.727360: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Wa
2024-05-06 17:03:34.519147: I external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:99
2024-05-06 17:03:34.663735: W tensorflow/core/common_runtime/gpu/gpu_device.cc:2251] Cannot d
Skipping registering GPU devices...
```

Model itself

```
np.random.seed(590154)

population = 1000
initial_infecteds = 10
epidemic_length = 1000
number_of_events = 15000

pv_champ_alpha = 0.4  # prop of effective care
pv_champ_beta = 0.4  # prop of radical cure
pv_champ_gamma_L = 1 / 223  # liver stage clearance rate
pv_champ_delta = 0.05  # prop of imported cases
pv_champ_lambda = 0.04  # transmission rate
pv_champ_f = 1 / 72  # relapse frequency
pv_champ_r = 1 / 60  # blood stage clearance rate

gamma_L_max = 1/30
lambda_max = 0.1
f_max = 1/14
r_max = 1/14


def champagne_stochastic(
    alpha_,
    beta_,
    gamma_L,
    lambda_,
    f,
    r,
```

```
    N=population,
    I_L=initial_infecteds,
    I_0=0,
    S_L=0,
    delta_=0,
    end_time=epidemic_length,
    num_events=number_of_events,
):
    if (0 > (alpha_ or beta_)) or (1 < (alpha_ or beta_)):
        return "Alpha or Beta out of bounds"
    if 0 > (gamma_L or lambda_ or f or r):
        return "Gamma, lambda, f or r out of bounds"

    t = 0
    S_0 = N - I_L - I_0 - S_L
    inc_counter = 0

    list_of_outcomes = [
        {"t": 0, "S_0": S_0, "S_L": S_L, "I_0": I_0, "I_L": I_L, "inc_counter": 0}
    ]

    prop_new = alpha_ * beta_ * f / (alpha_ * beta_ * f + gamma_L)
    i = 0

    while (i < num_events) or (t < 30):
        i += 1
        if S_0 == N:
            while t < 31:
                t += 1
                new_stages = {
                    "t": t,
                    "S_0": N,
                    "S_L": 0,
                    "I_0": 0,
                    "I_L": 0,
                    "inc_counter": inc_counter,
                }
                list_of_outcomes.append(new_stages)
            break

        S_0_to_I_L = (1 - alpha_) * lambda_ * (I_L + I_0) / N * S_0
        S_0_to_S_L = alpha_ * (1 - beta_) * lambda_ * (I_0 + I_L) / N * S_0
```

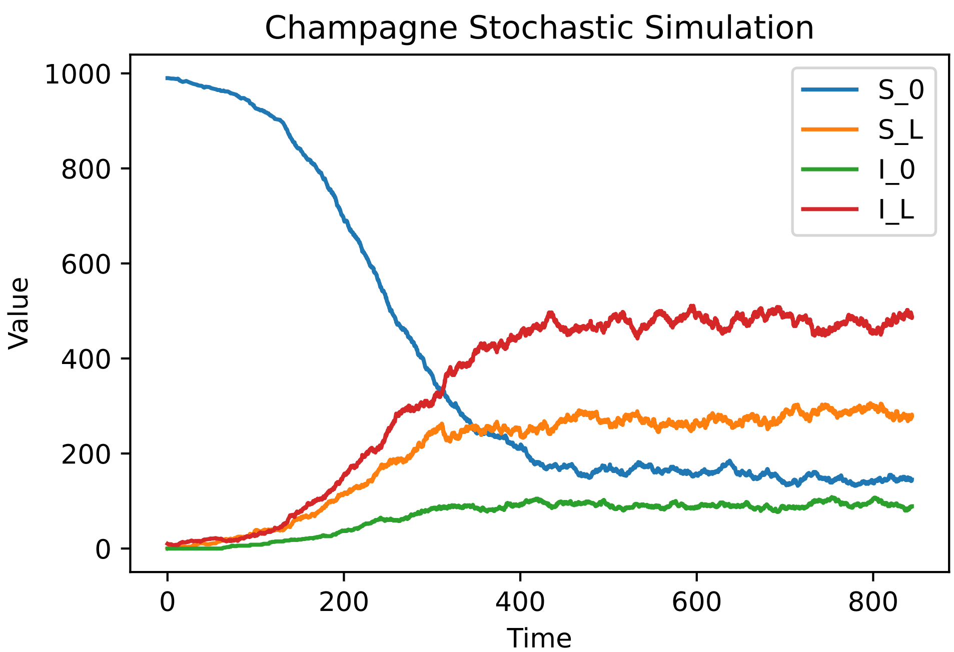
```
        I_0_to_S_0 = r * I_0 / N
        I_0_to_I_L = lambda_ * (I_L + I_0) / N * I_0
        I_L_to_I_0 = gamma_L * I_L
        I_L_to_S_L = r * I_L
        S_L_to_S_0 = (gamma_L + (f + lambda_ * (I_0 + I_L) / N) * alpha_ * beta_) * S_L
        S_L_to_I_L = (f + lambda_ * (I_0 + I_L) / N) * (1 - alpha_) * S_L

        total_rate = (
            S_0_to_I_L
            + S_0_to_S_L
            + I_0_to_S_0
            + I_0_to_I_L
            + I_L_to_I_0
            + I_L_to_S_L
            + S_L_to_S_0
            + S_L_to_I_L
        )

        delta_t = np.random.exponential(1 / total_rate)
        new_stages_prob = [
            S_0_to_I_L / total_rate,
            S_0_to_S_L / total_rate,
            I_0_to_S_0 / total_rate,
            I_0_to_I_L / total_rate,
            I_L_to_I_0 / total_rate,
            I_L_to_S_L / total_rate,
            S_L_to_S_0 / total_rate,
            S_L_to_I_L / total_rate,
        ]
        t += delta_t
        silent_incidences = np.random.poisson(
            delta_t * alpha_ * beta_ * lambda_ * (I_L + I_0) * S_0 / N
        )

        new_stages = np.random.choice(
            [
                {
                    "t": t,
                    "S_0": S_0 - 1,
                    "S_L": S_L,
                    "I_0": I_0,
                    "I_L": I_L + 1,
```

```
                "inc_counter": inc_counter + silent_incidences + 1,
            },
            {
                "t": t,
                "S_0": S_0 - 1,
                "S_L": S_L + 1,
                "I_0": I_0,
                "I_L": I_L,
                "inc_counter": inc_counter + silent_incidences + 1,
            },
            {
                "t": t,
                "S_0": S_0 + 1,
                "S_L": S_L,
                "I_0": I_0 - 1,
                "I_L": I_L,
                "inc_counter": inc_counter + silent_incidences,
            },
            {
                "t": t,
                "S_0": S_0,
                "S_L": S_L,
                "I_0": I_0 - 1,
                "I_L": I_L + 1,
                "inc_counter": inc_counter + silent_incidences,
            },
            {
                "t": t,
                "S_0": S_0,
                "S_L": S_L,
                "I_0": I_0 + 1,
                "I_L": I_L - 1,
                "inc_counter": inc_counter + silent_incidences,
            },
            {
                "t": t,
                "S_0": S_0,
                "S_L": S_L + 1,
                "I_0": I_0,
                "I_L": I_L - 1,
                "inc_counter": inc_counter + silent_incidences,
            },
```

```
                {
                    "t": t,
                    "S_0": S_0 + 1,
                    "S_L": S_L - 1,
                    "I_0": I_0,
                    "I_L": I_L,
                    "inc_counter": inc_counter
                    + silent_incidences
                    + np.random.binomial(1, prop_new),
                },
                {
                    "t": t,
                    "S_0": S_0,
                    "S_L": S_L - 1,
                    "I_0": I_0,
                    "I_L": I_L + 1,
                    "inc_counter": inc_counter + silent_incidences + 1,
                },
            ],
            p=new_stages_prob,
        )

        list_of_outcomes.append(new_stages)

        S_0 = new_stages["S_0"]
        I_0 = new_stages["I_0"]
        I_L = new_stages["I_L"]
        S_L = new_stages["S_L"]
        inc_counter = new_stages["inc_counter"]

    outcome_df = pd.DataFrame(list_of_outcomes)
    return outcome_df


champ_samp = champagne_stochastic(
    pv_champ_alpha,
    pv_champ_beta,
    pv_champ_gamma_L,
    pv_champ_lambda,
    pv_champ_f,
    pv_champ_r,
)  # .melt(id_vars='t')
```

Plotting outcome

```
champ_samp.drop("inc_counter", axis=1).plot(x="t", legend=True)
plt.xlabel("Time")
plt.ylabel("Value")
plt.title("Champagne Stochastic Simulation")
plt.savefig("champagne_GP_images/champagne_simulation.pdf")
plt.show()
```



Function that Outputs Final Prevalence

```
def incidence(df, start, days):
    start_ind = df[df["t"].le(start)].index[-1]
    end_ind = df[df["t"].le(start + days)].index[-1]
    incidence_week = df.iloc[end_ind]["inc_counter"] - df.iloc[start_ind]["inc_counter"]
    return incidence_week


def champ_sum_stats(alpha_, beta_, gamma_L, lambda_, f, r):
```

```
    champ_df_ = champagne_stochastic(alpha_, beta_, gamma_L, lambda_, f, r)
    fin_t = champ_df_.iloc[-1]["t"]
    first_month_inc = incidence(champ_df_, 0, 30)
    fin_t = champ_df_.iloc[-1]["t"]
    fin_week_inc = incidence(champ_df_, fin_t - 7, 7)
    fin_prev = champ_df_.iloc[-1]["I_0"] + champ_df_.iloc[-1]["I_L"]

    return np.array([fin_prev, first_month_inc, fin_week_inc])


observed_sum_stats = champ_sum_stats(
    pv_champ_alpha,
    pv_champ_beta,
    pv_champ_gamma_L,
    pv_champ_lambda,
    pv_champ_f,
    pv_champ_r,
)


def discrepency_fn(alpha_, beta_, gamma_L, lambda_, f, r): # best is L1 norm
    x = champ_sum_stats(alpha_, beta_, gamma_L, lambda_, f, r)
    # return np.sum(np.abs((x - observed_sum_stats) / observed_sum_stats))
    # return np.linalg.norm((x - observed_sum_stats) / observed_sum_stats)
    return np.log(np.linalg.norm((x - observed_sum_stats) / observed_sum_stats))
```

Testing the variances across different values of params etc.

```
# samples = 30
# cor_sums = np.zeros(samples)
# for i in range(samples):
#     cor_sums[i] = discrepency_fn(
#         pv_champ_alpha,
#         pv_champ_beta,
#         pv_champ_gamma_L,
#         pv_champ_lambda,
#         pv_champ_f,
#         pv_champ_r,
#     )

# cor_mean = np.mean(cor_sums)
# cor_s_2 = sum((cor_sums - cor_mean) ** 2) / (samples - 1)
```

```
# print(cor_mean, cor_s_2)

# doub_sums = np.zeros(samples)
# for i in range(samples):
#     doub_sums[i] = discrepency_fn(
#         2 * pv_champ_alpha,
#         2 * pv_champ_beta,
#         2 * pv_champ_gamma_L,
#         2 * pv_champ_lambda,
#         2 * pv_champ_f,
#         2 * pv_champ_r,
#     )

# doub_mean = np.mean(doub_sums)
# doub_s_2 = sum((doub_sums - doub_mean) ** 2) / (samples - 1)
# print(doub_mean, doub_s_2)

# half_sums = np.zeros(samples)
# for i in range(samples):
#     half_sums[i] = discrepency_fn(
#         pv_champ_alpha / 2,
#         pv_champ_beta / 2,
#         pv_champ_gamma_L / 2,
#         pv_champ_lambda / 2,
#         pv_champ_f / 2,
#         pv_champ_r / 2,
#     )

# half_mean = np.mean(half_sums)
# half_s_2 = sum((half_sums - half_mean) ** 2) / (samples - 1)
# print(half_mean, half_s_2)

# rogue_sums = np.zeros(samples)
# for i in range(samples):
#     rogue_sums[i] = discrepency_fn(
#         pv_champ_alpha / 2,
#         pv_champ_beta / 2,
#         pv_champ_gamma_L / 2,
#         pv_champ_lambda / 2,
#         pv_champ_f / 2,
#         pv_champ_r / 2,
#     )
```

```
# rogue_mean = np.mean(rogue_sums)
# rogue_s_2 = sum((rogue_sums - rogue_mean) ** 2) / (samples - 1)
# print(rogue_mean, rogue_s_2)

# plt.figure(figsize=(7, 4))
# plt.scatter(
#     np.array([half_mean, cor_mean, doub_mean, rogue_mean]),
#     np.array([half_s_2, cor_s_2, doub_s_2, rogue_s_2]),
# )
# plt.title("variance and mean")
# plt.xlabel("mean")
# plt.ylabel("variance")
# plt.show()
```

Gaussian Process Regression on Final Prevalence Discrepency

```
my_seed = np.random.default_rng(seed=1795)  # For replicability

num_samples = 12

variables_names = ["alpha", "beta", "gamma_L", "lambda", "f", "r"]

pv_champ_alpha = 0.4  # prop of effective care
pv_champ_beta = 0.4  # prop of radical cure
pv_champ_gamma_L = 1 / 223  # liver stage clearance rate
pv_champ_lambda = 0.04  # transmission rate
pv_champ_f = 1 / 72  # relapse frequency
pv_champ_r = 1 / 60  # blood stage clearance rate

samples = np.concatenate(
    (
        my_seed.uniform(low=0, high=1, size=(num_samples, 1)),  # alpha
        my_seed.uniform(low=0, high=1, size=(num_samples, 1)),  # beta
        my_seed.exponential(scale=pv_champ_gamma_L, size=(num_samples, 1)),  # gamma_L
        my_seed.exponential(scale=pv_champ_lambda, size=(num_samples, 1)),  # lambda
        my_seed.exponential(scale=pv_champ_f, size=(num_samples, 1)),  # f
        my_seed.exponential(scale=pv_champ_r, size=(num_samples, 1)),  # r
    ),
    axis=1,
```

```
)

LHC_sampler = qmc.LatinHypercube(d=6, seed=my_seed)
LHC_samples = LHC_sampler.random(n=num_samples)

# Using Champagne Initialisation table 2
LHC_samples[:, 2] = gamma_L_max * LHC_samples[:, 2]
LHC_samples[:, 3] = lambda_max * LHC_samples[:, 3]
LHC_samples[:, 4] = f_max * LHC_samples[:, 4]
LHC_samples[:, 5] = r_max * LHC_samples[:, 5]


# LHC_samples[:, 2] = 1/50* LHC_samples[:, 2]
# LHC_samples[:, 3] = 0.2 * LHC_samples[:, 3]
# LHC_samples[:, 4] = 1/10 * LHC_samples[:, 4]
# LHC_samples[:, 5] = 1/10 * LHC_samples[:, 5]
# LHC_samples[:, 2] = -pv_champ_gamma_L * np.log(LHC_samples[:, 2])
# LHC_samples[:, 3] = -pv_champ_lambda * np.log(LHC_samples[:, 3])
# LHC_samples[:, 4] = -pv_champ_f * np.log(LHC_samples[:, 4])
# LHC_samples[:, 5] = -pv_champ_r * np.log(LHC_samples[:, 5])

LHC_samples = np.repeat(LHC_samples, 10, axis = 0)

random_indices_df = pd.DataFrame(samples, columns=variables_names)
LHC_indices_df = pd.DataFrame(LHC_samples, columns=variables_names)

print(random_indices_df.head())
print(LHC_indices_df.head())
```

```
      alpha      beta   gamma_L    lambda         f         r
0  0.201552  0.059376  0.002013  0.034926  0.004799  0.017448
1  0.332324  0.694037  0.005082  0.029547  0.004978  0.011233
2  0.836050  0.859768  0.002921  0.035607  0.001421  0.007956
3  0.566773  0.561896  0.002327  0.012668  0.025850  0.007153
4  0.880603  0.481021  0.003977  0.025372  0.012134  0.001578
      alpha      beta   gamma_L  lambda         f         r
0  0.666699  0.759788  0.026395  0.0948  0.029288  0.023606
1  0.666699  0.759788  0.026395  0.0948  0.029288  0.023606
2  0.666699  0.759788  0.026395  0.0948  0.029288  0.023606
3  0.666699  0.759788  0.026395  0.0948  0.029288  0.023606
4  0.666699  0.759788  0.026395  0.0948  0.029288  0.023606
```

Generate Discrepencies

```
random_discrepencies = LHC_indices_df.apply(
    lambda x: discrepency_fn(
        x["alpha"], x["beta"], x["gamma_L"], x["lambda"], x["f"], x["r"]
    ),
    axis=1,
)

print(random_discrepencies.head())
```

```
0    0.369203
1    0.976110
2    0.964801
3    0.841432
4    0.253188
dtype: float64
```

Differing Methods to Iterate Function

```
# import timeit

# def function1():
#     np.vectorize(champ_sum_stats)(random_indices_df['alpha'],
#     random_indices_df['beta'], random_indices_df['gamma_L'],
#     random_indices_df['lambda'], random_indices_df['f'], random_indices_df['r'])
#     pass

# def function2():
#     random_indices_df.apply(
#         lambda x: champ_sum_stats(
#             x['alpha'], x['beta'], x['gamma_L'], x['lambda'], x['f'], x['r']),
#             axis = 1)
#     pass

# # Time function1
# time_taken_function1 = timeit.timeit(
#     "function1()", globals=globals(), number=100)
```

```
# # Time function2
# time_taken_function2 = timeit.timeit(
#     "function2()", globals=globals(), number=100)

# print("Time taken for function1:", time_taken_function1)
# print("Time taken for function2:", time_taken_function2)
```

Time taken for function1: 187.48960775700016 Time taken for function2: 204.06618941299985

Constrain Variables to be Positive

```
constrain_positive = tfb.Shift(np.finfo(np.float64).tiny)(tfb.Exp())
```

Custom Quadratic Mean Function

```
class quad_mean_fn(tf.Module):
    def __init__(self):
        super(quad_mean_fn, self).__init__()
        # self.amp_alpha_mean = tfp.util.TransformedVariable(
        #     bijector=constrain_positive,
        #     initial_value=1.0,
        #     dtype=np.float64,
        #     name="amp_alpha_mean",
        # )
        # self.alpha_tp = tf.Variable(pv_champ_alpha, dtype=np.float64, name="alpha_tp")
        # self.amp_beta_mean = tfp.util.TransformedVariable(
        #     bijector=constrain_positive,
        #     initial_value=0.5,
        #     dtype=np.float64,
        #     name="amp_beta_mean",
        # )
        # self.beta_tp = tf.Variable(pv_champ_beta, dtype=np.float64, name="beta_tp")
        self.amp_gamma_L_mean = tfp.util.TransformedVariable(
            bijector=constrain_positive,
            initial_value=1.0,
            dtype=np.float64,
            name="amp_gamma_L_mean",
        )
```

```
        # self.gamma_L_tp = tfp.util.TransformedVariable(
        #     bijector=constrain_positive,
        #     initial_value=1.0,
        #     dtype=np.float64,
        #     name="gamma_L_tp",
        # )
        self.amp_lambda_mean = tfp.util.TransformedVariable(
            bijector=constrain_positive,
            initial_value=1.0,
            dtype=np.float64,
            name="amp_lambda_mean",
        )
        # self.lambda_tp = tfp.util.TransformedVariable(
        #     bijector=constrain_positive,
        #     initial_value=1.0,
        #     dtype=np.float64,
        #     name="lambda_tp",
        # )
        self.amp_f_mean = tfp.util.TransformedVariable(
            bijector=constrain_positive,
            initial_value=1.0,
            dtype=np.float64,
            name="amp_f_mean",
        )
        # self.f_tp = tfp.util.TransformedVariable(
        #     bijector=constrain_positive,
        #     initial_value=1.0,
        #     dtype=np.float64,
        #     name="f_tp",
        # )
        self.amp_r_mean = tfp.util.TransformedVariable(
            bijector=constrain_positive,
            initial_value=1.0,
            dtype=np.float64,
            name="amp_r_mean",
        )
        # self.r_tp = tfp.util.TransformedVariable(
        #     bijector=constrain_positive,
        #     initial_value=1.0,
        #     dtype=np.float64,
        #     name="r_tp",
        # )
```

```
        # self.bias_mean = tfp.util.TransformedVariable(
        #     bijector=constrain_positive,
        #     initial_value=1.0,
        #     dtype=np.float64,
        #     name="bias_mean",
        # )
        self.bias_mean = tf.Variable(-1.5, dtype=np.float64, name="bias_mean")

    def __call__(self, x):
        return (
            self.bias_mean
            # + self.amp_alpha_mean * (x[..., 0] - self.alpha_tp) ** 2
            # + self.amp_beta_mean * (x[..., 1] - self.beta_tp) ** 2
            # + self.amp_gamma_L_mean * (x[..., 2] - self.gamma_L_tp) ** 2
            # + self.amp_lambda_mean * (x[..., 3] - self.lambda_tp) ** 2
            # + self.amp_f_mean * (x[..., 4] - self.f_tp) ** 2
            # + self.amp_r_mean * (x[..., 5] - self.r_tp) ** 2
            + self.amp_gamma_L_mean * (x[..., 2]) ** 2
            + self.amp_lambda_mean * (x[..., 3]) ** 2
            + self.amp_f_mean * (x[..., 4]) ** 2
            + self.amp_r_mean * (x[..., 5]) ** 2
        )


quad_mean_fn().__call__(x=np.array([[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]]))  # should return 1
```

```
<tf.Tensor: shape=(1,), dtype=float64, numpy=array([2.5])>
```

Custom Linear Mean Function

```
class lin_mean_fn(tf.Module):
    def __init__(self):
        super(lin_mean_fn, self).__init__()
        # self.amp_alpha_lin = tfp.util.TransformedVariable(
        #     bijector=constrain_positive,
        #     initial_value=1.0,
        #     dtype=np.float64,
        #     name="amp_alpha_lin",
        # )
        # self.amp_beta_lin = tfp.util.TransformedVariable(
```

```
    #     bijector=constrain_positive,
    #     initial_value=0.5,
    #     dtype=np.float64,
    #     name="amp_beta_lin",
    # )
    self.amp_gamma_L_lin = tfp.util.TransformedVariable(
        bijector=constrain_positive,
        initial_value=1.0,
        dtype=np.float64,
        name="amp_gamma_L_lin",
    )
    self.amp_lambda_lin = tfp.util.TransformedVariable(
        bijector=constrain_positive,
        initial_value=1.0,
        dtype=np.float64,
        name="amp_lambda_lin",
    )
    self.amp_f_lin = tfp.util.TransformedVariable(
        bijector=constrain_positive,
        initial_value=1.0,
        dtype=np.float64,
        name="amp_f_lin",
    )
    self.amp_r_lin = tfp.util.TransformedVariable(
        bijector=constrain_positive,
        initial_value=1.0,
        dtype=np.float64,
        name="amp_r_lin",
    )
    # self.bias_lin = tfp.util.TransformedVariable(
    #     bijector=constrain_positive,
    #     initial_value=1.0,
    #     dtype=np.float64,
    #     name="bias_lin",
    # )
    self.bias_lin = tf.Variable(0.0, dtype=np.float64, name="bias_mean")

def __call__(self, x):
    return (
        self.bias_lin
        # + self.amp_alpha_lin * (x[..., 0])
        # + self.amp_beta_lin * (x[..., 1])
```

```
            + self.amp_gamma_L_lin * (x[..., 2])
            + self.amp_lambda_lin * (x[..., 3])
            + self.amp_f_lin * (x[..., 4])
            + self.amp_r_lin * (x[..., 5])
        )
```

```
class const_mean_fn(tf.Module):
    def __init__(self):
        super(const_mean_fn, self).__init__()
        self.bias_lin = tf.Variable(0.0, dtype=np.float64, name="bias_mean")

    def __call__(self, x):
        return self.bias_lin
```

Making the ARD Kernel

```
index_vals = LHC_indices_df.values
obs_vals = random_discrepencies.values

amplitude_champ = tfp.util.TransformedVariable(
    bijector=constrain_positive,
    initial_value=1.0,
    dtype=np.float64,
    name="amplitude_champ",
)

observation_noise_variance_champ = tfp.util.TransformedVariable(
    bijector=constrain_positive,
    initial_value=1.,
    dtype=np.float64,
    name="observation_noise_variance_champ",
)
```

```
length_scales_champ = tfp.util.TransformedVariable(
    bijector=tfb.Sigmoid(np.float64(0.), [1./2, 1./2, gamma_L_max/2, lambda_max/2, f_max/2,
    initial_value=[1/4, 1/4, gamma_L_max/4, lambda_max/4, f_max/4, r_max/4],
    dtype=np.float64,
    name="length_scales_champ",
)
```

```
kernel_champ = tfk.FeatureScaled(
    tfk.MaternFiveHalves(amplitude=amplitude_champ),
    scale_diag=length_scales_champ,
)
```

Define the Gaussian Process with Quadratic Mean Function and ARD Kernel

```
# Define Gaussian Process with the custom kernel
champ_GP = tfd.GaussianProcess(
    kernel=kernel_champ,
    observation_noise_variance=observation_noise_variance_champ,
    index_points=index_vals,
    mean_fn=const_mean_fn(),
)

print(champ_GP.trainable_variables)

Adam_optim = tf.optimizers.Adam(learning_rate=0.01)
```

```
(<tf.Variable 'amplitude_champ:0' shape=() dtype=float64, numpy=0.0>, <tf.Variable 'length_s
```

Train the Hyperparameters

```
# predictive log stuff
@tf.function(autograph=False, jit_compile=False)
def optimize():
    with tf.GradientTape() as tape:
        K = (
            champ_GP.kernel.matrix(index_vals, index_vals)
            + tf.eye(index_vals.shape[0], dtype=np.float64)
            * observation_noise_variance_champ
        )
        means = champ_GP.mean_fn(index_vals)
        K_inv = tf.linalg.inv(K)
        K_inv_y = K_inv @ tf.reshape(obs_vals - means, shape=[obs_vals.shape[0], 1])
        K_inv_diag = tf.linalg.diag_part(K_inv)
        log_var = tf.math.log(K_inv_diag)
        log_mu = tf.reshape(K_inv_y, shape=[-1]) ** 2
```

```
        loss = -tf.math.reduce_sum(log_var - log_mu)
    grads = tape.gradient(loss, champ_GP.trainable_variables)
    Adam_optim.apply_gradients(zip(grads, champ_GP.trainable_variables))
    return loss


num_iters = 10000

lls_ = np.zeros(num_iters, np.float64)
tolerance = 1e-6  # Set your desired tolerance level
previous_loss = float("inf")

for i in range(num_iters):
    loss = optimize()
    lls_[i] = loss

    # Check if change in loss is less than tolerance
    if abs(loss - previous_loss) < tolerance:
        print(f"Hyperparameter convergence reached at iteration {i+1}.")
        lls_ = lls_[range(i + 1)]
        break

    previous_loss = loss
```

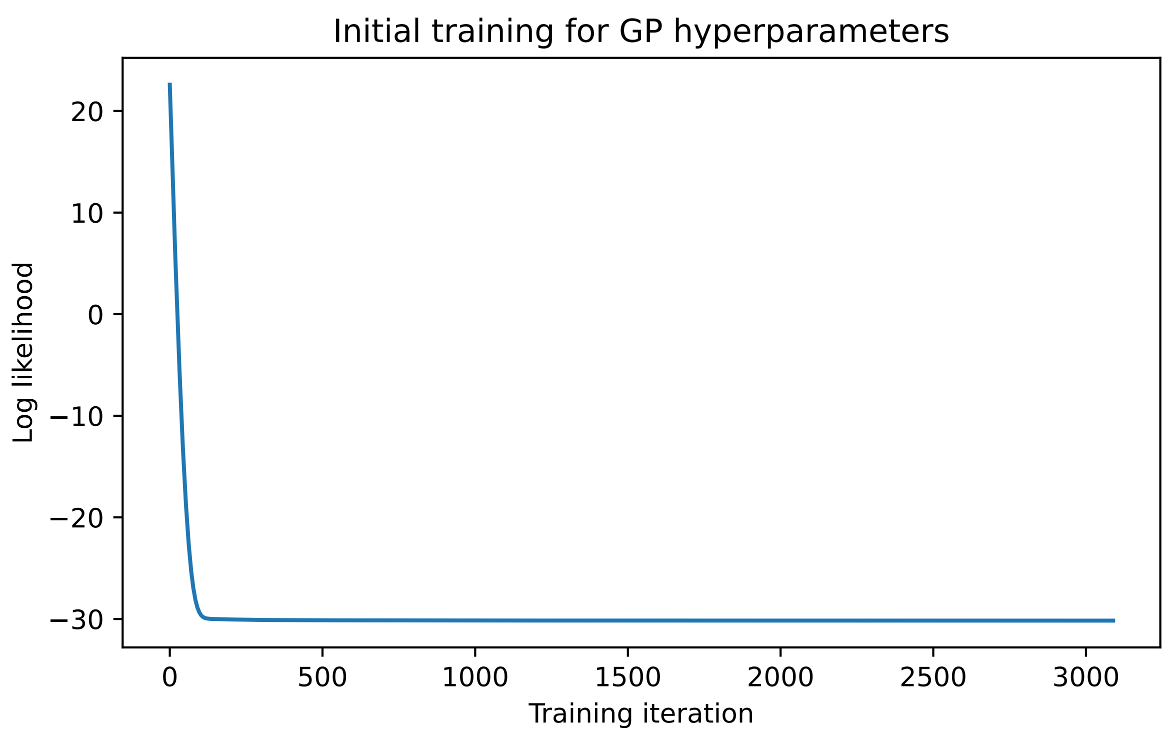
```
Hyperparameter convergence reached at iteration 3090.
```

```
print("Trained parameters:")
for var in champ_GP.trainable_variables:
    if "bias" in var.name:
        print("{} is {}\n".format(var.name, var.numpy().round(3)))
    else:
        if "length" in var.name:
            print(
                "{} is {}\n".format(
                    var.name,
                    tfb.Sigmoid(np.float64(0.0), 0.5).forward(var).numpy().round(3),
                )
            )
        else:
            print(
                "{} is {}\n".format(
```

```
                    var.name, constrain_positive.forward(var).numpy().round(3)
                )
            )
```

```
Trained parameters:
amplitude_champ:0 is 0.527

length_scales_champ:0 is [0.499 0.499 0.499 0.215 0.499 0.499]

observation_noise_variance_champ:0 is 0.432

bias_mean:0 is 0.281
```

```
plt.figure(figsize=(7, 4))
plt.plot(lls_)
plt.title("Initial training for GP hyperparameters")
plt.xlabel("Training iteration")
plt.ylabel("Log likelihood")
plt.savefig("champagne_GP_images/hyperparam_loss_log_discrep.pdf")
plt.show()
```



Creating slices across one variable dimension

```
plot_samp_no = 21
plot_gp_no = 100
gp_samp_no = 30
```

```
slice_samples_dict = {
    "alpha_slice_samples": np.repeat(np.concatenate(
        (
            np.linspace(0, 1, plot_samp_no, dtype=np.float64).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_samp_no).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_samp_no).reshape(-1, 1),  # gamma_L
            np.repeat(pv_champ_lambda, plot_samp_no).reshape(-1, 1),  # lambda
            np.repeat(pv_champ_f, plot_samp_no).reshape(-1, 1),  # f
            np.repeat(pv_champ_r, plot_samp_no).reshape(-1, 1),  # r
        ),
        axis=1,
    ), 5, axis = 0),
    "alpha_gp_samples": np.concatenate(
        (
            np.linspace(0, 1, plot_gp_no, dtype=np.float64).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_gp_no).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_gp_no).reshape(-1, 1),  # gamma_L
            np.repeat(pv_champ_lambda, plot_gp_no).reshape(-1, 1),  # lambda
            np.repeat(pv_champ_f, plot_gp_no).reshape(-1, 1),  # f
            np.repeat(pv_champ_r, plot_gp_no).reshape(-1, 1),  # r
        ),
        axis=1,
    ),
    "beta_slice_samples": np.repeat(np.concatenate(
        (
            np.repeat(pv_champ_alpha, plot_samp_no).reshape(-1, 1),  # alpha
            np.linspace(0, 1, plot_samp_no, dtype=np.float64).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_samp_no).reshape(-1, 1),  # gamma_L
            np.repeat(pv_champ_lambda, plot_samp_no).reshape(-1, 1),  # lambda
            np.repeat(pv_champ_f, plot_samp_no).reshape(-1, 1),  # f
            np.repeat(pv_champ_r, plot_samp_no).reshape(-1, 1),  # r
        ),
        axis=1,
    ), 5, axis = 0),
    "beta_gp_samples": np.concatenate(
```

```
        (
            np.repeat(pv_champ_alpha, plot_gp_no).reshape(-1, 1),  # alpha
            np.linspace(0, 1, plot_gp_no, dtype=np.float64).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_gp_no).reshape(-1, 1),  # gamma_L
            np.repeat(pv_champ_lambda, plot_gp_no).reshape(-1, 1),  # lambda
            np.repeat(pv_champ_f, plot_gp_no).reshape(-1, 1),  # f
            np.repeat(pv_champ_r, plot_gp_no).reshape(-1, 1),  # r
        ),
        axis=1,
    ),
    "gamma_L_slice_samples": np.repeat(np.concatenate(
        (
            np.repeat(pv_champ_alpha, plot_samp_no).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_samp_no).reshape(-1, 1),  # beta
            np.linspace(0, gamma_L_max, plot_samp_no, dtype=np.float64).reshape(-1, 1),  # ga
            np.repeat(pv_champ_lambda, plot_samp_no).reshape(-1, 1),  # lambda
            np.repeat(pv_champ_f, plot_samp_no).reshape(-1, 1),  # f
            np.repeat(pv_champ_r, plot_samp_no).reshape(-1, 1),  # r
        ),
        axis=1,
    ), 5, axis = 0),
    "gamma_L_gp_samples": np.concatenate(
        (
            np.repeat(pv_champ_alpha, plot_gp_no).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_gp_no).reshape(-1, 1),  # beta
            np.linspace(0, gamma_L_max, plot_gp_no, dtype=np.float64).reshape(-1, 1),  # gamm
            np.repeat(pv_champ_lambda, plot_gp_no).reshape(-1, 1),  # lambda
            np.repeat(pv_champ_f, plot_gp_no).reshape(-1, 1),  # f
            np.repeat(pv_champ_r, plot_gp_no).reshape(-1, 1),  # r
        ),
        axis=1,
    ),
    "lambda_slice_samples": np.repeat(np.concatenate(
        (
            np.repeat(pv_champ_alpha, plot_samp_no).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_samp_no).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_samp_no).reshape(-1, 1),  # gamma_L
            np.linspace(0, lambda_max, plot_samp_no, dtype=np.float64).reshape(-1, 1), # lamb
            np.repeat(pv_champ_f, plot_samp_no).reshape(-1, 1),  # f
            np.repeat(pv_champ_r, plot_samp_no).reshape(-1, 1),  # r
        ),
        axis=1,
```

```
    ), 5, axis = 0),
    "lambda_gp_samples": np.concatenate(
        (
            np.repeat(pv_champ_alpha, plot_gp_no).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_gp_no).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_gp_no).reshape(-1, 1),  # gamma_L
            np.linspace(0, lambda_max, plot_gp_no, dtype=np.float64).reshape(-1, 1), # lambda
            np.repeat(pv_champ_f, plot_gp_no).reshape(-1, 1),  # f
            np.repeat(pv_champ_r, plot_gp_no).reshape(-1, 1),  # r
        ),
        axis=1,
    ),
    "f_slice_samples": np.repeat(np.concatenate(
        (
            np.repeat(pv_champ_alpha, plot_samp_no).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_samp_no).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_samp_no).reshape(-1, 1),  # gamma_L
            np.repeat(pv_champ_lambda, plot_samp_no).reshape(-1, 1),  # lambda
            np.linspace(0, f_max, plot_samp_no, dtype=np.float64).reshape(-1, 1), # f
            np.repeat(pv_champ_r, plot_samp_no).reshape(-1, 1),  # r
        ),
        axis=1,
    ), 5, axis = 0),
    "f_gp_samples": np.concatenate(
        (
            np.repeat(pv_champ_alpha, plot_gp_no).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_gp_no).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_gp_no).reshape(-1, 1),  # gamma_L
            np.repeat(pv_champ_lambda, plot_gp_no).reshape(-1, 1),  # lambda
            np.linspace(0, f_max, plot_gp_no, dtype=np.float64).reshape(-1, 1), # f
            np.repeat(pv_champ_r, plot_gp_no).reshape(-1, 1),  # r
        ),
        axis=1,
    ),
    "r_slice_samples": np.repeat(np.concatenate(
        (
            np.repeat(pv_champ_alpha, plot_samp_no).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_samp_no).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_samp_no).reshape(-1, 1),  # gamma_L
            np.repeat(pv_champ_lambda, plot_samp_no).reshape(-1, 1),  # lambda
            np.repeat(pv_champ_f, plot_samp_no).reshape(-1, 1),  # f
            np.linspace(0, r_max, plot_samp_no, dtype=np.float64).reshape(-1, 1), # r
```

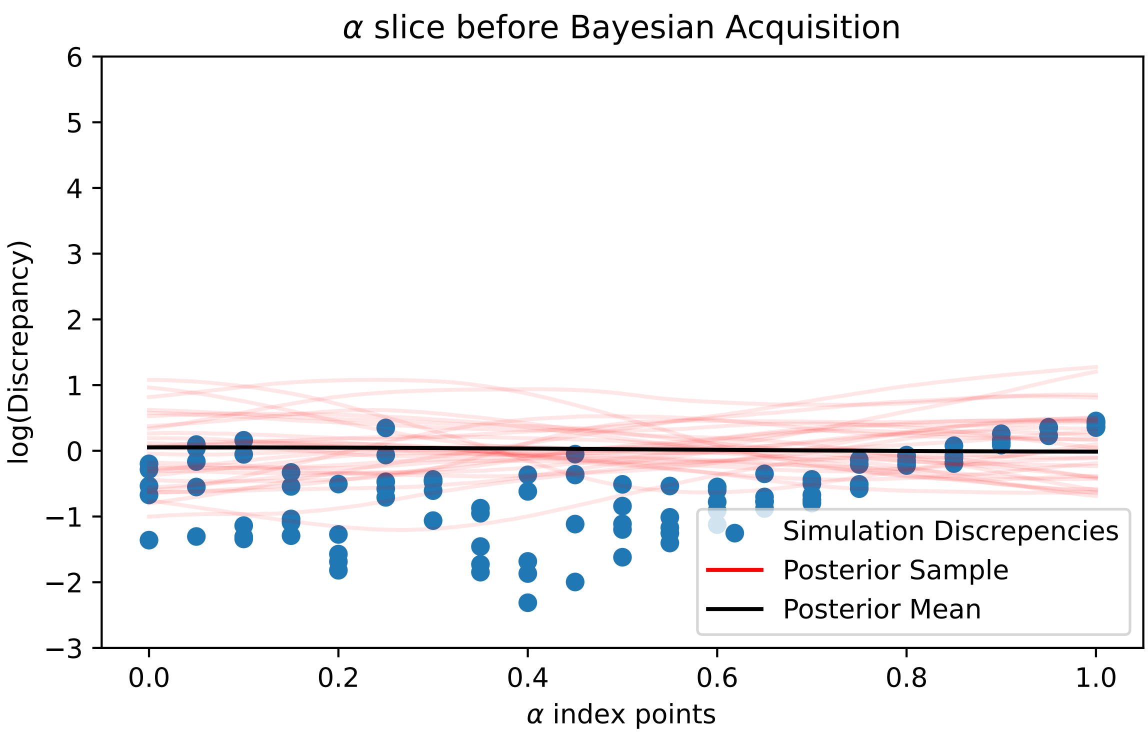
```
        ),
        axis=1,
    ), 5, axis = 0),
    "r_gp_samples": np.concatenate(
        (
            np.repeat(pv_champ_alpha, plot_gp_no).reshape(-1, 1),  # alpha
            np.repeat(pv_champ_beta, plot_gp_no).reshape(-1, 1),  # beta
            np.repeat(pv_champ_gamma_L, plot_gp_no).reshape(-1, 1),  # gamma_L
            np.repeat(pv_champ_lambda, plot_gp_no).reshape(-1, 1),  # lambda
            np.repeat(pv_champ_f, plot_gp_no).reshape(-1, 1),  # f
            np.linspace(0, r_max, plot_gp_no, dtype=np.float64).reshape(-1, 1), # r
        ),
        axis=1,
    ),
}
```

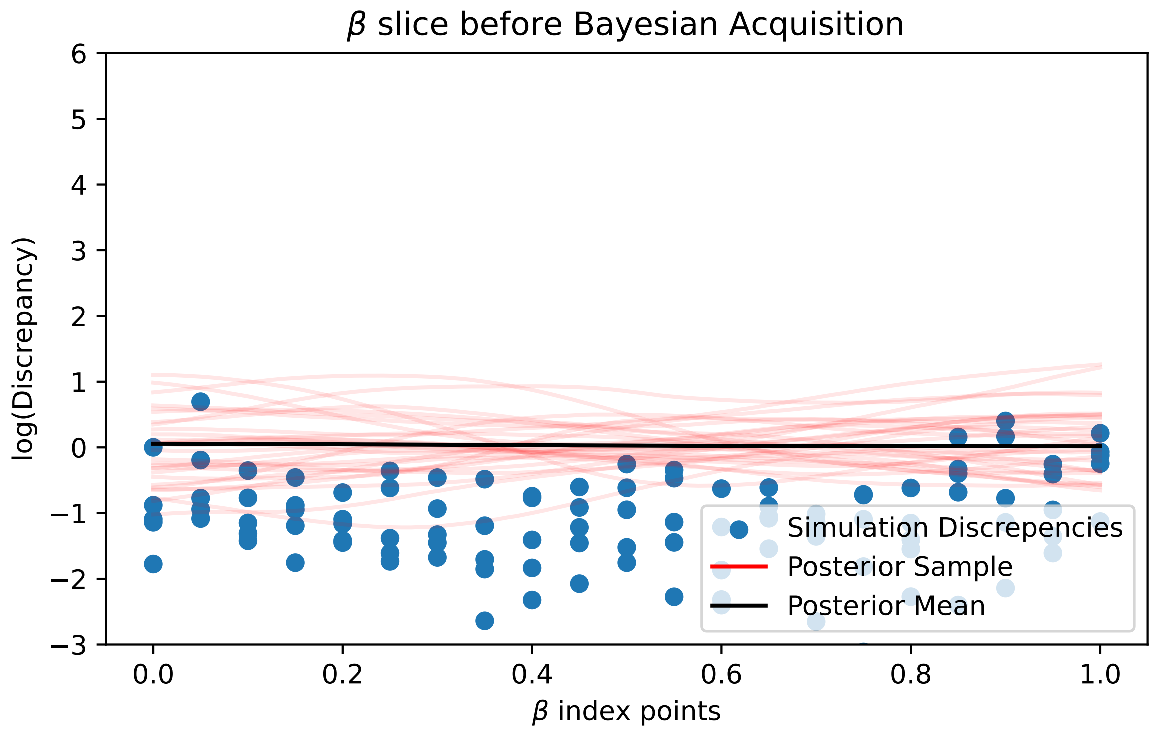
Plotting the GPs across different slices

```
GP_seed = tfp.random.sanitize_seed(4362)
vars = ["alpha", "beta", "gamma_L", "lambda", "f", "r"]
slice_indices_dfs_dict = {}
slice_index_vals_dict = {}
slice_discrepencies_dict = {}

for var in vars:
    val_df = pd.DataFrame(
        slice_samples_dict[var + "_slice_samples"], columns=variables_names
    )
    slice_indices_dfs_dict[var + "_slice_indices_df"] = val_df
    slice_index_vals_dict[var + "_slice_index_vals"] = val_df.values
    discreps = val_df.apply(
        lambda x: discrepency_fn(
            x["alpha"], x["beta"], x["gamma_L"], x["lambda"], x["f"], x["r"]
        ),
        axis=1,
    )
    slice_discrepencies_dict[var + "_slice_discrepencies"] = discreps


    gp_samples_df = pd.DataFrame(
```

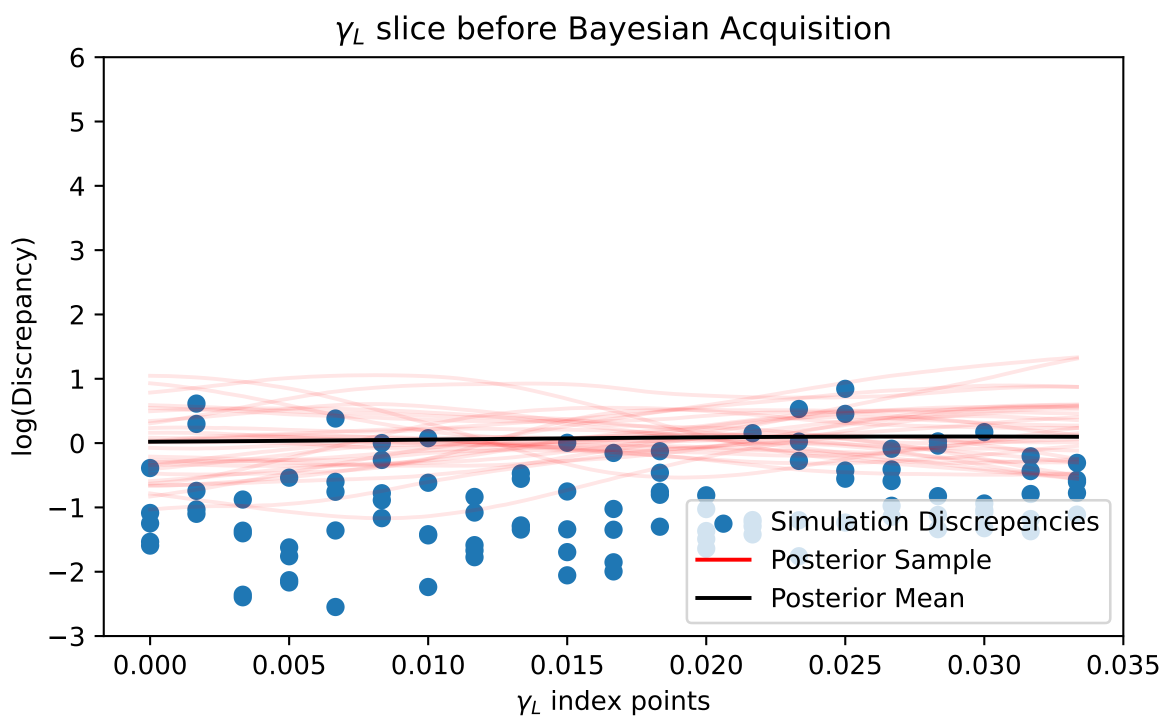
```
        slice_samples_dict[var + "_gp_samples"], columns=variables_names
    )
    slice_indices_dfs_dict[var + "_gp_indices_df"] = gp_samples_df
    slice_index_vals_dict[var + "_gp_index_vals"] = gp_samples_df.values

    champ_GP_reg = tfd.GaussianProcessRegressionModel(
        kernel=kernel_champ,
        index_points=gp_samples_df.values,
        observation_index_points=index_vals,
        observations=obs_vals,
        observation_noise_variance=observation_noise_variance_champ,
        predictive_noise_variance=0.0,
        mean_fn=const_mean_fn(),
    )
    GP_samples = champ_GP_reg.sample(gp_samp_no, seed=GP_seed)

    plt.figure(figsize=(7, 4))
    plt.scatter(
        val_df[var].values,
        discreps,
        label = "Simulation Discrepencies",
    )
    for i in range(gp_samp_no):
        plt.plot(
            gp_samples_df[var].values,
            GP_samples[i, :],
            c="r",
            alpha=0.1,
            label="Posterior Sample" if i == 0 else None,
        )
    plt.plot(
        slice_indices_dfs_dict[var + "_gp_indices_df"][var].values,
        champ_GP_reg.mean_fn(slice_indices_dfs_dict[var + "_gp_indices_df"].values),
        c="black",
        alpha=1,
        label="Posterior Mean",
    )
    leg = plt.legend(loc="lower right")
    for lh in leg.legend_handles:
        lh.set_alpha(1)
    if var in ["f", "r"]:
        plt.xlabel("$" + var + "$ index points")
```

```
        plt.title("$" + var + "$ slice before Bayesian Acquisition")
    else:
        plt.xlabel("$\\" + var + "$ index points")
        plt.title("$\\" + var + "$ slice before Bayesian Acquisition")
    # if var not in ["alpha", "beta"]:
    #     plt.xscale("log", base=np.e)
    plt.ylabel("log(Discrepancy)")
    plt.ylim((-3, 6))
    plt.savefig("champagne_GP_images/initial_" + var + "_slice_log_discrep.pdf")
    plt.show()
```

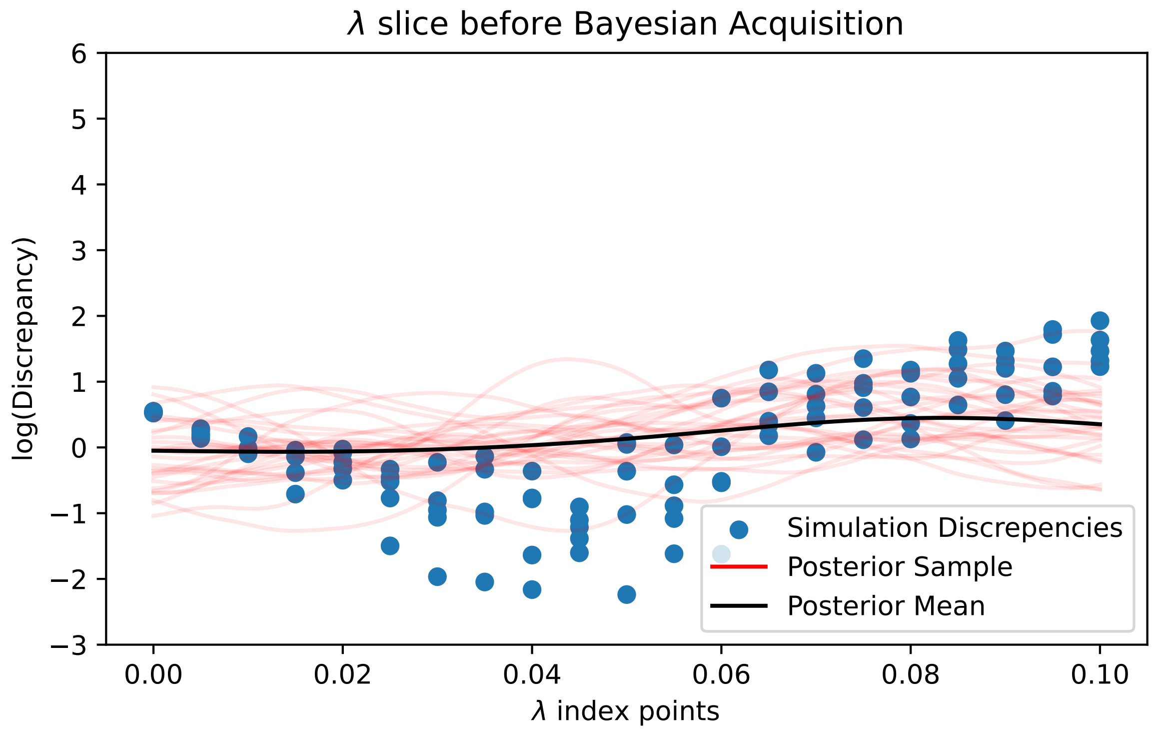


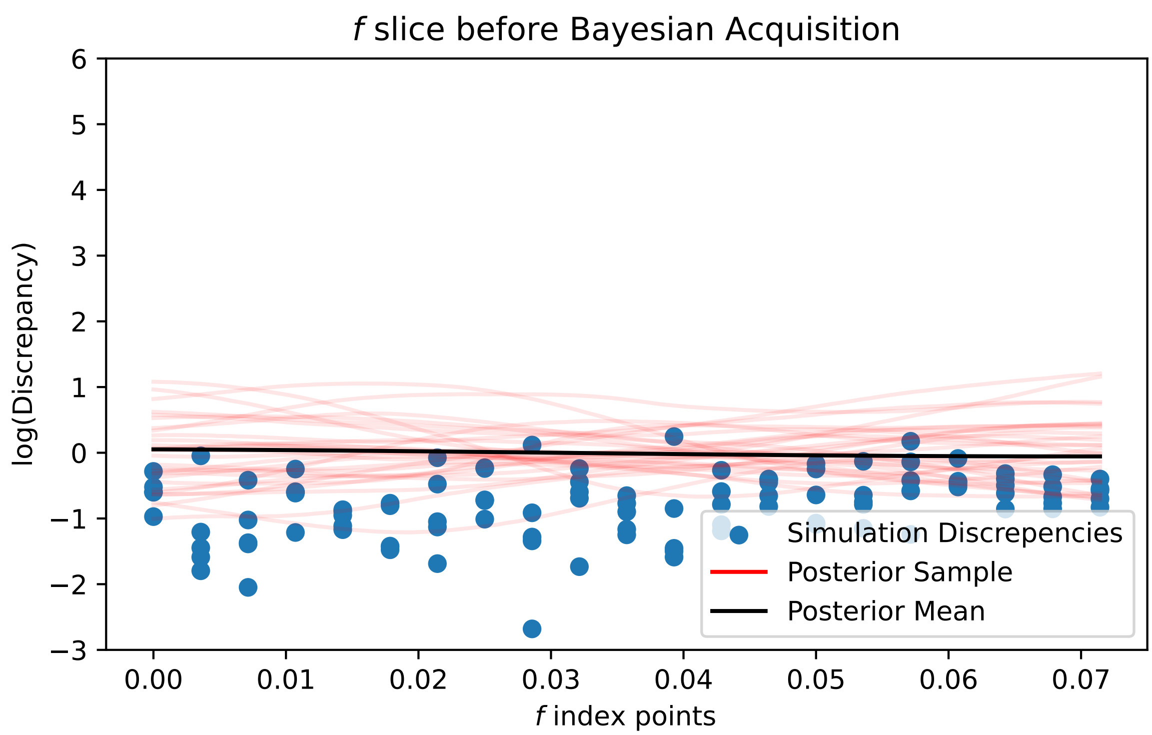


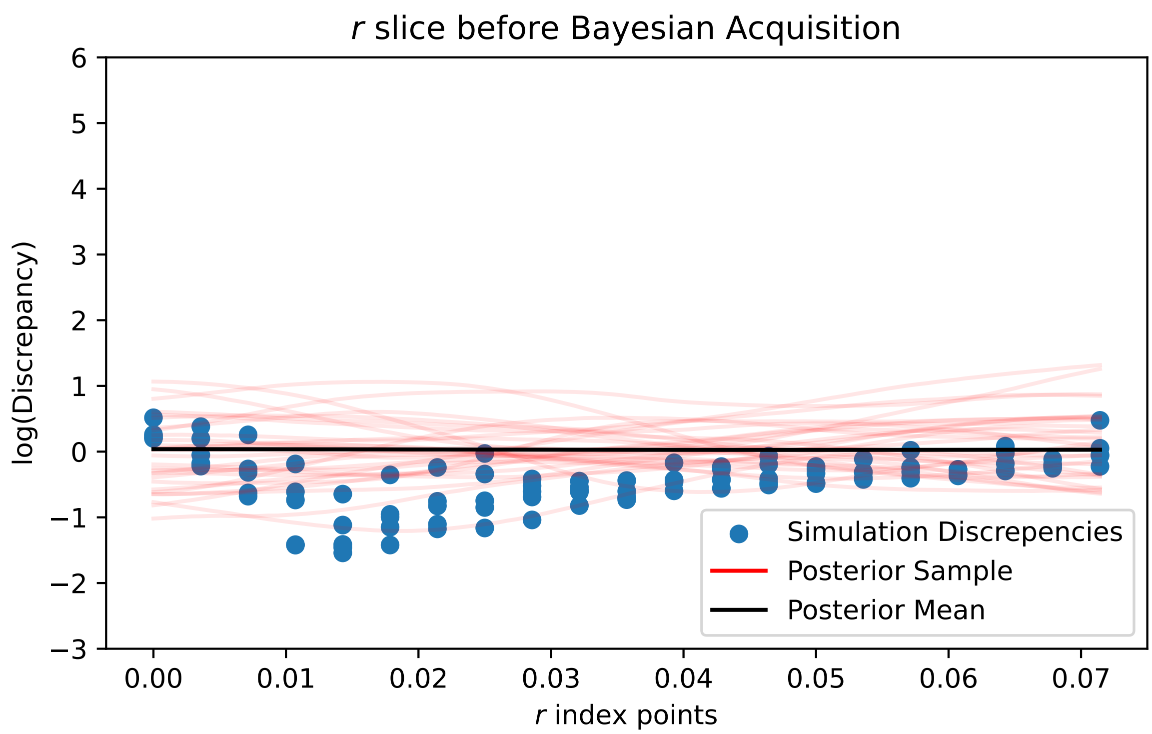
β slice before Bayesian Acquisition
log(Discrepancy)
β index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



γ_L slice before Bayesian Acquisition
log(Discrepancy)
γ_L index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


λ slice before Bayesian Acquisition
log(Discrepancy)
6
5
4
3
2
1
0
−1
−2
−3
0.00
0.02
0.04
0.06
0.08
0.10
λ index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


f slice before Bayesian Acquisition
log(Discrepancy)
6
5
4
3
2
1
0
−1
−2
−3
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
f index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



Acquiring the next datapoint to test

Proof that .variance returns what we need in acquisition function

```
new_guess = np.array([0.4, 0.4, 0.004, 0.04, 0.01, 0.17])
mean_t = champ_GP_reg.mean_fn(new_guess)
variance_t = champ_GP_reg.variance(index_points=[new_guess])

kernel_self = kernel_champ.apply(new_guess, new_guess)
kernel_others = kernel_champ.apply(new_guess, index_vals)
K = kernel_champ.matrix(
    index_vals, index_vals
) + observation_noise_variance_champ * np.identity(index_vals.shape[0])
inv_K = np.linalg.inv(K)
print("Self Kernel is {}".format(kernel_self.numpy().round(3)))
print("Others Kernel is {}".format(kernel_others.numpy().round(3)))
print(inv_K)
my_var_t = kernel_self - kernel_others.numpy() @ inv_K @ kernel_others.numpy()
```

```python
print("Variance function is {}".format(variance_t.numpy().round(3)))
print("Variance function is {}".format(my_var_t.numpy().round(3)))
```

```
Self Kernel is 0.278
Others Kernel is [0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.002 0.002
 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002
 0.002 0.002 0.002 0.002 0.002 0.002 0.001 0.001 0.001 0.001 0.001 0.001
 0.001 0.001 0.001 0.001 0.    0.    0.    0.    0.    0.    0.    0.
 0.    0.    0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002
 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002
 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.002 0.    0.    0.    0.
 0.    0.    0.    0.    0.    0.    0.001 0.001 0.001 0.001 0.001 0.001
 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001 0.001
 0.001 0.001 0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   ]
[[ 2.11643604e+00 -1.98547355e-01 -1.98547355e-01 ... -6.03519576e-04
  -6.03519576e-04 -6.03519576e-04]
 [-1.98547355e-01  2.11643604e+00 -1.98547355e-01 ... -6.03519576e-04
  -6.03519576e-04 -6.03519576e-04]
 [-1.98547355e-01 -1.98547355e-01  2.11643604e+00 ... -6.03519576e-04
  -6.03519576e-04 -6.03519576e-04]
 ...
 [-6.03519576e-04 -6.03519576e-04 -6.03519576e-04 ...  2.11581749e+00
  -1.99165906e-01 -1.99165906e-01]
 [-6.03519576e-04 -6.03519576e-04 -6.03519576e-04 ... -1.99165906e-01
   2.11581749e+00 -1.99165906e-01]
 [-6.03519576e-04 -6.03519576e-04 -6.03519576e-04 ... -1.99165906e-01
  -1.99165906e-01  2.11581749e+00]]
Variance function is [0.278]
Variance function is 0.278
```

Loss function

```python
next_alpha = tfp.util.TransformedVariable(
    initial_value=0.5,
    bijector=tfb.Sigmoid(),
    dtype=np.float64,
    name="next_alpha",
)
```

```
next_beta = tfp.util.TransformedVariable(
    initial_value=0.5,
    bijector=tfb.Sigmoid(),
    dtype=np.float64,
    name="next_beta",
)

next_gamma_L = tfp.util.TransformedVariable(
    initial_value=gamma_L_max/2,
    bijector=tfb.Sigmoid(np.float64(0.), gamma_L_max),
    dtype=np.float64,
    name="next_gamma_L",
)

next_lambda = tfp.util.TransformedVariable(
    initial_value=lambda_max/2,
    bijector=tfb.Sigmoid(np.float64(0.), lambda_max),
    dtype=np.float64,
    name="next_lambda",
)

next_f = tfp.util.TransformedVariable(
    initial_value=f_max/2,
    bijector=tfb.Sigmoid(np.float64(0.), f_max),
    dtype=np.float64,
    name="next_f",
)

next_r = tfp.util.TransformedVariable(
    initial_value=r_max/2,
    bijector=tfb.Sigmoid(np.float64(0.), r_max),
    dtype=np.float64,
    name="next_r",
)

next_vars = (
    (next_alpha.trainable_variables[0],
    next_beta.trainable_variables[0],
    next_gamma_L.trainable_variables[0],
    next_lambda.trainable_variables[0],
    next_f.trainable_variables[0],
    next_r.trainable_variables[0],)
```

```
)

next_vars
```

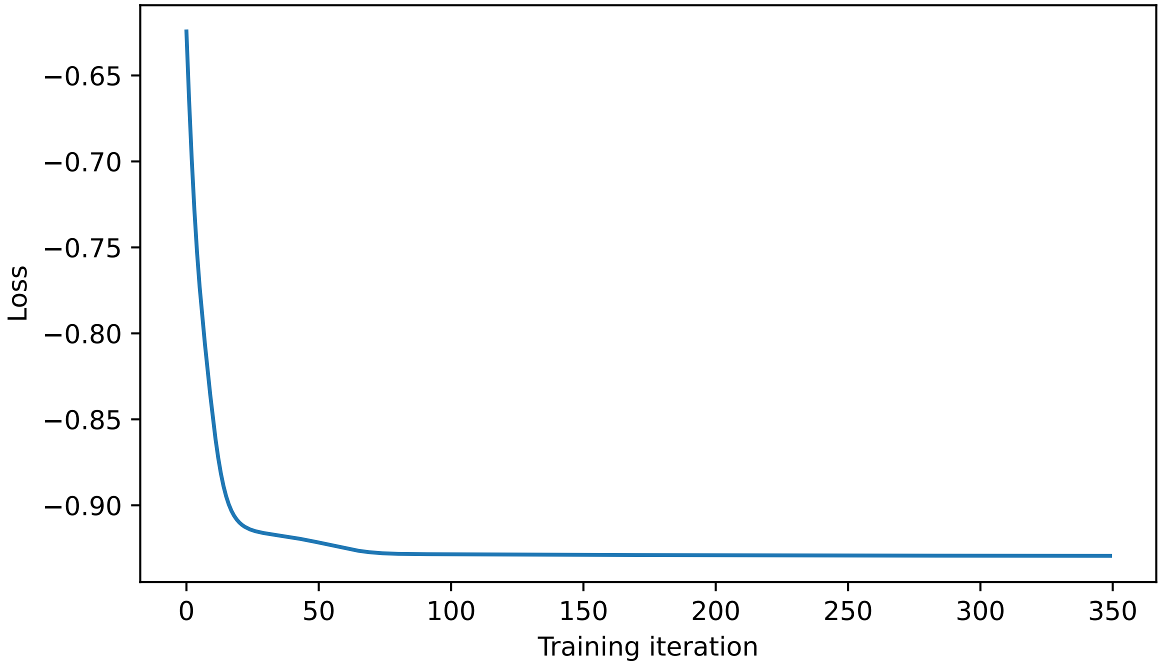
```
(<tf.Variable 'next_alpha:0' shape=() dtype=float64, numpy=0.0>,
 <tf.Variable 'next_beta:0' shape=() dtype=float64, numpy=0.0>,
 <tf.Variable 'next_gamma_L:0' shape=() dtype=float64, numpy=0.0>,
 <tf.Variable 'next_lambda:0' shape=() dtype=float64, numpy=0.0>,
 <tf.Variable 'next_f:0' shape=() dtype=float64, numpy=0.0>,
 <tf.Variable 'next_r:0' shape=() dtype=float64, numpy=0.0>)
```

```
Adam_optim = tf.optimizers.Adam(learning_rate=0.1)

@tf.function(autograph=False, jit_compile=False)
def optimize():
    with tf.GradientTape() as tape:
        next_guess = tf.reshape(
            tf.stack(
                [next_alpha, next_beta, next_gamma_L, next_lambda, next_f, next_r]
            ),
            [1, 6],
        )
        mean_t = champ_GP_reg.mean_fn(next_guess)
        std_t = champ_GP_reg.stddev(index_points=next_guess)
        loss = tf.squeeze(mean_t - 1.7 * std_t)
    grads = tape.gradient(loss, next_vars)
    Adam_optim.apply_gradients(zip(grads, next_vars))
    return loss

num_iters = 10000

lls_ = np.zeros(num_iters, np.float64)
tolerance = 1e-6  # Set your desired tolerance level
previous_loss = float("inf")

for i in range(num_iters):
    loss = optimize()
    lls_[i] = loss

    # Check if change in loss is less than tolerance
    if abs(loss - previous_loss) < tolerance:
```

```
        print(f"Acquisition function convergence reached at iteration {i+1}.")
        lls_ = lls_[range(i + 1)]
        break

    previous_loss = loss

print("Trained parameters:")
for var in [next_alpha, next_beta, next_gamma_L, next_lambda, next_f, next_r]:
    print("{} is {}".format(var.name, (var.bijector.forward(var).numpy().round(3))))
# if ("alpha" in var.name) | ("beta" in var.name):
#     print(
#         "{} is {}".format(var.name, (tfb.Sigmoid().forward(var).numpy().round(3)))
#     )
# else:
#     print(
#         "{} is {}".format(
#             var.name, constrain_positive.forward(var).numpy().round(3)
#         )
#     )
```

```
Acquisition function convergence reached at iteration 350.
Trained parameters:
next_alpha is 0.707
next_beta is 0.573
next_gamma_L is 0.017
next_lambda is 0.051
next_f is 0.037
next_r is 0.036
```

```
plt.figure(figsize=(7, 4))
plt.plot(lls_)
plt.xlabel("Training iteration")
plt.ylabel("Loss")
plt.savefig("champagne_GP_images/bolfi_optim_loss_log_discrep.pdf")
plt.show()
```




```
def update_GP():
    @tf.function(autograph=False, jit_compile=False)
    def opt_GP():
        with tf.GradientTape() as tape:
            K = (
                champ_GP.kernel.matrix(index_vals, index_vals)
                + tf.eye(index_vals.shape[0], dtype=np.float64)
                * observation_noise_variance_champ
            )
            means = champ_GP.mean_fn(index_vals)
            K_inv = tf.linalg.inv(K)
            K_inv_y = K_inv @ tf.reshape(obs_vals - means, shape=[obs_vals.shape[0], 1])
            K_inv_diag = tf.linalg.diag_part(K_inv)
            log_var = tf.math.log(K_inv_diag)
            log_mu = tf.reshape(K_inv_y, shape=[-1]) ** 2
            loss = -tf.math.reduce_sum(log_var - log_mu)
        grads = tape.gradient(loss, champ_GP.trainable_variables)
        optimizer_slow.apply_gradients(zip(grads, champ_GP.trainable_variables))
        return loss

    num_iters = 10000
```

```
    lls_ = np.zeros(num_iters, np.float64)
    tolerance = 1e-6  # Set your desired tolerance level
    previous_loss = float("inf")

    for i in range(num_iters):
        loss = opt_GP()
        lls_[i] = loss.numpy()

        # Check if change in loss is less than tolerance
        if abs(loss - previous_loss) < tolerance:
            print(f"Hyperparameter convergence reached at iteration {i+1}.")
            lls_ = lls_[range(i + 1)]
            break

        previous_loss = loss
    for var in optimizer_slow.variables:
        var.assign(tf.zeros_like(var))


def update_var_UCB():
    optimizer_fast = tf.optimizers.Adam(learning_rate=1.0)

    @tf.function(autograph=False, jit_compile=False)
    def opt_var():
        with tf.GradientTape() as tape:
            next_guess = tf.reshape(
                tf.stack(
                    [next_alpha, next_beta, next_gamma_L, next_lambda, next_f, next_r]
                ),
                [1, 6],
            )
            mean_t = champ_GP_reg.mean_fn(next_guess)
            std_t = champ_GP_reg.stddev(index_points=next_guess)
            loss = tf.squeeze(mean_t - eta_t * std_t)
        grads = tape.gradient(loss, next_vars)
        optimizer_fast.apply_gradients(zip(grads, next_vars))
        return loss

    num_iters = 10000

    lls_ = np.zeros(num_iters, np.float64)
    tolerance = 1e-6  # Set your desired tolerance level
```

```
    previous_loss = float("inf")

    for i in range(num_iters):
        loss = opt_var()
        lls_[i] = loss

        # Check if change in loss is less than tolerance
        if abs(loss - previous_loss) < tolerance:
            print(f"Acquisition function convergence reached at iteration {i+1}.")
            lls_ = lls_[range(i + 1)]
            break

        previous_loss = loss

    next_guess = tf.reshape(
        tf.stack([next_alpha, next_beta, next_gamma_L, next_lambda, next_f, next_r]),
        [1, 6],
    )
    print(
        "The final UCB loss was {}".format(loss.numpy().round(3))
        + " with predicted mean of {}".format(
            champ_GP_reg.mean_fn(next_guess).numpy().round(3)
        )
    )
    for var in optimizer_fast.variables:
        var.assign(tf.zeros_like(var))


def update_var_EI():
    optimizer_fast = tf.optimizers.Adam(learning_rate=1.0)

    @tf.function(autograph=False, jit_compile=False)
    def opt_var():
        with tf.GradientTape() as tape:
            next_guess = tf.reshape(
                tf.stack(
                    [next_alpha, next_beta, next_gamma_L, next_lambda, next_f, next_r]
                ),
                [1, 6],
            )
            mean_t = champ_GP_reg.mean_fn(next_guess)
            std_t = champ_GP_reg.stddev(index_points=next_guess)
```

```
            delt = min_obs - mean_t
            loss = -tf.squeeze(
                delt * tfd.Normal(0, std_t).cdf(delt)
                + std_t * champ_GP_reg.prob(delt, index_points=next_guess)
            )
        grads = tape.gradient(loss, next_vars)
        optimizer_fast.apply_gradients(zip(grads, next_vars))
        return loss

    num_iters = 10000

    lls_ = np.zeros(num_iters, np.float64)
    tolerance = 1e-9  # Set your desired tolerance level
    previous_loss = np.float64("inf")

    for i in range(num_iters):
        loss = opt_var()
        lls_[i] = loss

        # Check if change in loss is less than tolerance
        if (i > 200) and (abs(loss - previous_loss) < tolerance):
            print(f"Acquisition function convergence reached at iteration {i+1}.")
            lls_ = lls_[range(i + 1)]
            break

        previous_loss = loss
    print(loss)
    for var in optimizer_fast.variables:
        var.assign(tf.zeros_like(var))


# EI = tfp_acq.GaussianProcessExpectedImprovement(champ_GP_reg, obs_vals)


def new_eta_t(t, d, exploration_rate):
    # return np.log((t + 1) ** (d * 2 + 2) * np.pi**2 / (3 * exploration_rate))
    return np.sqrt(np.log((t + 1) ** (d * 2 + 2) * np.pi**2 / (3 * exploration_rate)))
```

```
# optimizer_fast = tf.optimizers.Adam(learning_rate=1.)
# update_var_EI()
# plt.figure(figsize=(7, 4))
# plt.plot(lls_)
```

```
# plt.xlabel("Training iteration")
# plt.ylabel("Loss")
# plt.show()
```

```
exploration_rate = 0.1
d = 6
update_freq = 20  # how many iterations before updating GP hyperparams
eta_t = tf.Variable(0, dtype=np.float64, name="eta_t")
min_obs = tf.Variable(100, dtype=np.float64, name="min_obs", shape=())
min_index = index_vals[
    champ_GP_reg.mean_fn(index_vals) == min(champ_GP_reg.mean_fn(index_vals))
][
    0,
]

for t in range(401):
    # min_index = index_vals[
    #     champ_GP_reg.mean_fn(index_vals) == min(champ_GP_reg.mean_fn(index_vals))
    # ][
    #     0,
    # ]
    optimizer_slow = tf.optimizers.Adam()
    eta_t.assign(new_eta_t(t, d, exploration_rate))
    # min_obs.assign(min(champ_GP_reg.mean_fn(index_vals)))
    print("Iteration " + str(t))
    # print(eta_t)

    ######################################################################
    var_num = 0

    for var in [next_alpha, next_beta, next_gamma_L, next_lambda, next_f, next_r]:
        var.assign(
            var.bijector.forward(
                (var.bijector.forward(np.float64(100000000.0))
                * np.float64(np.random.uniform()))
            )
        )
        # if ("alpha" in var.name) or ("beta" in var.name):
        #     var.assign(tfb.Sigmoid().inverse(np.float64(np.random.uniform())))
        # else:
        #     var.assign(tfb.Sigmoid().inverse(np.float64(np.random.uniform())))
        var_num += 1
```

```
update_var_UCB()
# update_var_EI()
# print(next_vars)

new_params = np.array(
    [
        next_alpha.numpy(),
        next_beta.numpy(),
        next_gamma_L.numpy(),
        next_lambda.numpy(),
        next_f.numpy(),
        next_r.numpy(),
    ]
).reshape(1, -1)
print("The next parameters to simulate from are {}".format(new_params.round(3)))

for repeats in range(5):
    new_discrepency = discrepency_fn(
        next_alpha.numpy(),
        next_beta.numpy(),
        next_gamma_L.numpy(),
        next_lambda.numpy(),
        next_f.numpy(),
        next_r.numpy(),
    )

    index_vals = np.append(
        index_vals,
        new_params,
        axis=0,
    )
    obs_vals = np.append(obs_vals, new_discrepency)
######################################################################
# var_num = 0

# for var in next_vars:
#     if ('alpha' in var.name) or ('beta' in var.name):
#         var.assign(tfb.Sigmoid().inverse(min_index[var_num]))
#     else:
#         var.assign(constrain_positive.inverse(min_index[var_num]))
#     var_num += 1
```

```
# # for var in next_vars:
# #     if ('alpha' in var.name) or ('beta' in var.name):
# #         var.assign(tfb.Sigmoid().inverse(np.float64(np.random.uniform())))
# #     else:
# #         var.assign(constrain_positive.inverse(np.float64(np.random.uniform())))
# #     var_num += 1

# update_var_UCB()
# # update_var_EI()
# # print(next_vars)

# new_params = np.array(
#     [
#         next_alpha.numpy(),
#         next_beta.numpy(),
#         next_gamma_L.numpy(),
#         next_lambda.numpy(),
#         next_f.numpy(),
#         next_r.numpy(),
#     ]
# ).reshape(1, -1)
# print(new_params)

# for repeats in range(2):
#     new_discrepency = discrepency_fn(
#         next_alpha.numpy(),
#         next_beta.numpy(),
#         next_gamma_L.numpy(),
#         next_lambda.numpy(),
#         next_f.numpy(),
#         next_r.numpy(),
#     )

#     index_vals = np.append(
#         index_vals,
#         new_params,
#         axis=0,
#     )
#     obs_vals = np.append(obs_vals, new_discrepency)
######################################################################

print(
```

```
        "The mean of the samples was {}".format(
            ((obs_vals[-1] + obs_vals[-2]) / 2).round(3)
        )
    )

    if (t + 1) % update_freq == 0:
        champ_GP = tfd.GaussianProcess(
            kernel=kernel_champ,
            observation_noise_variance=observation_noise_variance_champ,
            index_points=index_vals,
            mean_fn=const_mean_fn(),
        )
        update_GP()
        min_value = min(champ_GP_reg.mean_fn(index_vals))
        min_index = index_vals[champ_GP_reg.mean_fn(index_vals) == min_value][0,]
        print(
            "The minimum predicted mean of the observed indices is {}".format(
                min_value.numpy().round(3)
            )
            + " at the point {}".format(min_index.round(3))
        )

    champ_GP_reg = tfd.GaussianProcessRegressionModel(
        kernel=kernel_champ,
        observation_index_points=index_vals,
        observations=obs_vals,
        observation_noise_variance=observation_noise_variance_champ,
        predictive_noise_variance=0.0,
        mean_fn=const_mean_fn(),
    )

    if (t > 0) & (t % 50 == 0):
        print("Trained parameters:")
        for train_var in champ_GP.trainable_variables:
            if "bias" in train_var.name:
                print("{} is {}\n".format(train_var.name, train_var.numpy().round(3)))
            else:
                if "length" in train_var.name:
                    print(
                        "{} is {}\n".format(
                            train_var.name,
                            tfb.Sigmoid().forward(train_var).numpy().round(3),
```

```
                    )
                )
            else:
                print(
                    "{} is {}\n".format(
                        train_var.name,
                        constrain_positive.forward(train_var).numpy().round(3),
                    )
                )
    # if "length" in train_var.name:
    #     print(
    #         "{} is {}\n".format(
    #             train_var.name,
    #             tfb.Sigmoid().forward(train_var).numpy().round(3),
    #         )
    #     )
    # else:
    #     if "tp" in train_var.name:  # or "bias" in var.name:
    #         print(
    #             "{} is {}\n".format(train_var.name, train_var.numpy().round(3))
    #         )
    #     else:
    #         print(
    #             "{} is {}\n".format(
    #                 train_var.name,
    #                 constrain_positive.forward(train_var).numpy().round(3),
    #             )
    #         )
    for var in vars:
        champ_GP_reg = tfd.GaussianProcessRegressionModel(
            kernel=kernel_champ,
            index_points=slice_indices_dfs_dict[var + "_gp_indices_df"].values,
            observation_index_points=index_vals,
            observations=obs_vals,
            observation_noise_variance=observation_noise_variance_champ,
            predictive_noise_variance=0.0,
            mean_fn=const_mean_fn(),
        )
        GP_samples = champ_GP_reg.sample(gp_samp_no, seed=GP_seed)

        plt.figure(figsize=(7, 4))
        plt.scatter(
```

```
                slice_indices_dfs_dict[var + "_slice_indices_df"][var].values,
                slice_discrepencies_dict[var + "_slice_discrepencies"],
                label="Simulation Discrepencies",
            )
            for i in range(gp_samp_no):
                plt.plot(
                    slice_indices_dfs_dict[var + "_gp_indices_df"][var].values,
                    GP_samples[i, :],
                    c="r",
                    alpha=0.1,
                    label="Posterior Sample" if i == 0 else None,
                )
            plt.plot(
                slice_indices_dfs_dict[var + "_gp_indices_df"][var].values,
                champ_GP_reg.mean_fn(
                    slice_indices_dfs_dict[var + "_gp_indices_df"].values
                ),
                c="black",
                alpha=1,
                label="Posterior Mean",
            )
            leg = plt.legend(loc="lower right")
            for lh in leg.legend_handles:
                lh.set_alpha(1)
            if var in ["f", "r"]:
                plt.xlabel("$" + var + "$ index points")
                plt.title(
                    "$" + var + "$ slice after " + str(t) + " Bayesian acquisitions"
                )
            else:
                plt.xlabel("$\\" + var + "$ index points")
                plt.title(
                    "$\\" + var + "$ slice after " + str(t) + " Bayesian acquisitions"
                )
            plt.ylabel("log(Discrepancy)")
            plt.ylim((-3, 6))
            plt.savefig(
                "champagne_GP_images/"
                + var
                + "_slice_"
                + str(t)
                + "_bolfi_updates_log_discrep.pdf"
```

```
    )
    plt.show()
```

```
Iteration 0
Acquisition function convergence reached at iteration 28.
The final UCB loss was -0.986 with predicted mean of [-0.009]
The next parameters to simulate from are [[1.    0.999 0.    0.001 0.    0.   ]]
The mean of the samples was 0.545
Iteration 1
Acquisition function convergence reached at iteration 84.
The final UCB loss was -1.871 with predicted mean of [0.027]
The next parameters to simulate from are [[1.    1.    0.    0.036 0.    0.071]]
The mean of the samples was 0.449
Iteration 2
Acquisition function convergence reached at iteration 240.
The final UCB loss was -2.197 with predicted mean of [0.04]
The next parameters to simulate from are [[0. 1. 0. 0. 0. 0.]]
The mean of the samples was 0.393
Iteration 3
Acquisition function convergence reached at iteration 247.
The final UCB loss was -2.491 with predicted mean of [0.009]
The next parameters to simulate from are [[0.    1.    0.033 0.    0.    0.071]]
The mean of the samples was 0.549
Iteration 4
Acquisition function convergence reached at iteration 70.
The final UCB loss was -2.562 with predicted mean of [0.098]
The next parameters to simulate from are [[0.    1.    0.033 0.055 0.    0.071]]
The mean of the samples was -0.731
Iteration 5
Acquisition function convergence reached at iteration 85.
The final UCB loss was -2.738 with predicted mean of [0.033]
The next parameters to simulate from are [[0.999 1.    0.033 0.059 0.    0.071]]
The mean of the samples was 0.102
Iteration 6
Acquisition function convergence reached at iteration 68.
The final UCB loss was -2.849 with predicted mean of [0.015]
The next parameters to simulate from are [[0.    1.    0.    0.047 0.    0.071]]
The mean of the samples was 0.414
Iteration 7
Acquisition function convergence reached at iteration 41.
The final UCB loss was -2.976 with predicted mean of [-0.011]
The next parameters to simulate from are [[0.    1.    0.033 0.043 0.071 0.071]]
```

```
The mean of the samples was 0.372
Iteration 8
Acquisition function convergence reached at iteration 60.
The final UCB loss was -2.774 with predicted mean of [-0.116]
The next parameters to simulate from are [[0.    0.999 0.033 0.032 0.    0.071]]
The mean of the samples was -0.821
Iteration 9
Acquisition function convergence reached at iteration 176.
The final UCB loss was -3.025 with predicted mean of [0.082]
The next parameters to simulate from are [[1.    1.    0.033 0.    0.    0.071]]
The mean of the samples was 0.549
Iteration 10
Acquisition function convergence reached at iteration 214.
The final UCB loss was -3.072 with predicted mean of [0.109]
The next parameters to simulate from are [[0.  1.  0.  0.1 0.  0. ]]
The mean of the samples was 2.408
Iteration 11
Acquisition function convergence reached at iteration 83.
The final UCB loss was -3.076 with predicted mean of [0.144]
The next parameters to simulate from are [[1.    1.    0.    0.053 0.    0.   ]]
The mean of the samples was 0.545
Iteration 12
Acquisition function convergence reached at iteration 91.
The final UCB loss was -3.303 with predicted mean of [-0.035]
The next parameters to simulate from are [[0.002 0.    0.033 0.033 0.    0.071]]
The mean of the samples was -0.753
Iteration 13
Acquisition function convergence reached at iteration 358.
The final UCB loss was -3.303 with predicted mean of [-0.043]
The next parameters to simulate from are [[0.38  0.    0.033 0.    0.    0.071]]
The mean of the samples was 0.549
Iteration 14
Acquisition function convergence reached at iteration 256.
The final UCB loss was -3.344 with predicted mean of [0.028]
The next parameters to simulate from are [[0.997 0.999 0.033 0.    0.071 0.   ]]
The mean of the samples was 0.43
Iteration 15
Acquisition function convergence reached at iteration 79.
The final UCB loss was -3.297 with predicted mean of [0.05]
The next parameters to simulate from are [[0.973 0.    0.033 0.047 0.    0.071]]
The mean of the samples was 0.404
Iteration 16
Acquisition function convergence reached at iteration 272.
```

```
The final UCB loss was -3.251 with predicted mean of [0.112]
The next parameters to simulate from are [[1.    1.    0.033 0.029 0.    0.   ]]
The mean of the samples was 0.122
Iteration 17
Acquisition function convergence reached at iteration 81.
The final UCB loss was -3.366 with predicted mean of [0.063]
The next parameters to simulate from are [[0.998 0.999 0.033 0.058 0.071 0.   ]]
The mean of the samples was -0.193
Iteration 18
Acquisition function convergence reached at iteration 86.
The final UCB loss was -3.465 with predicted mean of [-0.]
The next parameters to simulate from are [[0.001 0.001 0.033 0.022 0.071 0.071]]
The mean of the samples was -0.374
Iteration 19
Acquisition function convergence reached at iteration 318.
The final UCB loss was -3.446 with predicted mean of [0.085]
The next parameters to simulate from are [[0.    0.999 0.033 0.1   0.071 0.071]]
The mean of the samples was 1.641
Hyperparameter convergence reached at iteration 4524.
The minimum predicted mean of the observed indices is -0.722 at the point [0.    1.    0.033
Iteration 20
Acquisition function convergence reached at iteration 524.
The final UCB loss was -7.684 with predicted mean of [0.312]
The next parameters to simulate from are [[1.    1.    0.033 0.1   0.071 0.071]]
The mean of the samples was 2.333
Iteration 21
Acquisition function convergence reached at iteration 65.
The final UCB loss was -8.107 with predicted mean of [0.045]
The next parameters to simulate from are [[1.    1.    0.    0.    0.071 0.071]]
The mean of the samples was 0.389
Iteration 22
Acquisition function convergence reached at iteration 131.
The final UCB loss was -7.941 with predicted mean of [0.133]
The next parameters to simulate from are [[1.    1.    0.    0.    0.071 0.   ]]
The mean of the samples was 0.513
Iteration 23
Acquisition function convergence reached at iteration 74.
The final UCB loss was -7.739 with predicted mean of [0.153]
The next parameters to simulate from are [[0.441 1.    0.033 0.    0.071 0.071]]
The mean of the samples was 0.5
Iteration 24
Acquisition function convergence reached at iteration 548.
The final UCB loss was -7.622 with predicted mean of [0.179]
```

```
The next parameters to simulate from are [[0.    1.    0.033 0.1   0.    0.071]]
The mean of the samples was 0.844
Iteration 25
Acquisition function convergence reached at iteration 550.
The final UCB loss was -8.23 with predicted mean of [0.131]
The next parameters to simulate from are [[1.    1.    0.    0.1   0.071 0.   ]]
The mean of the samples was 1.67
Iteration 26
Acquisition function convergence reached at iteration 478.
The final UCB loss was -6.962 with predicted mean of [0.392]
The next parameters to simulate from are [[1.    1.    0.033 0.049 0.071 0.071]]
The mean of the samples was 1.616
Iteration 27
Acquisition function convergence reached at iteration 652.
The final UCB loss was -7.799 with predicted mean of [0.179]
The next parameters to simulate from are [[0.    1.    0.    0.    0.    0.071]]
The mean of the samples was 0.549
Iteration 28
Acquisition function convergence reached at iteration 600.
The final UCB loss was -8.171 with predicted mean of [0.102]
The next parameters to simulate from are [[0.    1.    0.    0.    0.071 0.071]]
The mean of the samples was 0.375
Iteration 29
Acquisition function convergence reached at iteration 604.
The final UCB loss was -8.24 with predicted mean of [0.257]
The next parameters to simulate from are [[0.    0.    0.033 0.1   0.071 0.071]]
The mean of the samples was 1.802
Iteration 30
Acquisition function convergence reached at iteration 76.
The final UCB loss was -7.274 with predicted mean of [0.121]
The next parameters to simulate from are [[0.002 0.997 0.015 0.036 0.04  0.071]]
The mean of the samples was -0.02
Iteration 31
Acquisition function convergence reached at iteration 397.
The final UCB loss was -8.261 with predicted mean of [0.266]
The next parameters to simulate from are [[1.    1.    0.    0.1   0.    0.071]]
The mean of the samples was 0.373
Iteration 32
Acquisition function convergence reached at iteration 111.
The final UCB loss was -7.33 with predicted mean of [-0.037]
The next parameters to simulate from are [[0.443 0.999 0.033 0.045 0.    0.035]]
The mean of the samples was -0.862
Iteration 33
```

```
Acquisition function convergence reached at iteration 320.
The final UCB loss was -8.757 with predicted mean of [-0.068]
The next parameters to simulate from are [[0.    0.    0.033 0.    0.071 0.   ]]
The mean of the samples was 0.452
Iteration 34
Acquisition function convergence reached at iteration 99.
The final UCB loss was -8.421 with predicted mean of [0.034]
The next parameters to simulate from are [[0.    0.    0.    0.047 0.    0.071]]
The mean of the samples was 0.367
Iteration 35
Acquisition function convergence reached at iteration 75.
The final UCB loss was -7.411 with predicted mean of [0.364]
The next parameters to simulate from are [[1.    1.    0.018 0.    0.036 0.039]]
The mean of the samples was 0.513
Iteration 36
Acquisition function convergence reached at iteration 104.
The final UCB loss was -7.032 with predicted mean of [-0.008]
The next parameters to simulate from are [[0.463 0.999 0.033 0.035 0.027 0.071]]
The mean of the samples was -0.391
Iteration 37
Acquisition function convergence reached at iteration 372.
The final UCB loss was -8.667 with predicted mean of [0.112]
The next parameters to simulate from are [[1.    0.    0.    0.    0.    0.071]]
The mean of the samples was 0.549
Iteration 38
Acquisition function convergence reached at iteration 98.
The final UCB loss was -7.272 with predicted mean of [0.126]
The next parameters to simulate from are [[0.493 0.999 0.016 0.062 0.    0.071]]
The mean of the samples was -0.145
Iteration 39
Acquisition function convergence reached at iteration 380.
The final UCB loss was -8.685 with predicted mean of [0.093]
The next parameters to simulate from are [[0. 0. 0. 0. 0. 0.]]
The mean of the samples was 0.444
Hyperparameter convergence reached at iteration 1475.
The minimum predicted mean of the observed indices is -0.807 at the point [0.443 0.999 0.033
Iteration 40
Acquisition function convergence reached at iteration 107.
The final UCB loss was -6.816 with predicted mean of [0.06]
The next parameters to simulate from are [[0.    0.001 0.033 0.024 0.    0.   ]]
The mean of the samples was 0.218
Iteration 41
Acquisition function convergence reached at iteration 325.
```

```
The final UCB loss was -5.812 with predicted mean of [0.033]
The next parameters to simulate from are [[0.511 1.    0.017 0.022 0.    0.071]]
The mean of the samples was 0.246
Iteration 42
Acquisition function convergence reached at iteration 70.
The final UCB loss was -6.847 with predicted mean of [0.07]
The next parameters to simulate from are [[0.001 1.    0.033 0.03  0.071 0.   ]]
The mean of the samples was 0.21
Iteration 43
Acquisition function convergence reached at iteration 85.
The final UCB loss was -6.71 with predicted mean of [0.104]
The next parameters to simulate from are [[0.    1.    0.033 0.    0.    0.   ]]
The mean of the samples was 0.447
Iteration 44
Acquisition function convergence reached at iteration 405.
The final UCB loss was -6.643 with predicted mean of [0.333]
The next parameters to simulate from are [[0.    1.    0.033 0.081 0.071 0.   ]]
The mean of the samples was 1.679
Iteration 45
Acquisition function convergence reached at iteration 344.
The final UCB loss was -6.617 with predicted mean of [0.151]
The next parameters to simulate from are [[1.    1.    0.033 0.077 0.    0.   ]]
The mean of the samples was 0.137
Iteration 46
Acquisition function convergence reached at iteration 109.
The final UCB loss was -6.535 with predicted mean of [-0.122]
The next parameters to simulate from are [[0.    0.244 0.033 0.052 0.039 0.071]]
The mean of the samples was 0.213
Iteration 47
Acquisition function convergence reached at iteration 103.
The final UCB loss was -6.053 with predicted mean of [0.004]
The next parameters to simulate from are [[0.615 1.    0.033 0.034 0.043 0.   ]]
The mean of the samples was 0.202
Iteration 48
Acquisition function convergence reached at iteration 50.
The final UCB loss was -6.288 with predicted mean of [0.366]
The next parameters to simulate from are [[0.    1.    0.    0.044 0.    0.   ]]
The mean of the samples was 0.521
Iteration 49
Acquisition function convergence reached at iteration 366.
The final UCB loss was -6.483 with predicted mean of [-0.012]
The next parameters to simulate from are [[0.    0.447 0.033 0.    0.071 0.049]]
The mean of the samples was 0.5
```

```
Iteration 50
Acquisition function convergence reached at iteration 100.
The final UCB loss was -6.356 with predicted mean of [-0.346]
The next parameters to simulate from are [[0.    0.496 0.033 0.042 0.    0.038]]
The mean of the samples was -1.461
Trained parameters:
amplitude_champ:0 is 0.951

length_scales_champ:0 is [1.    1.    1.    0.516 1.    1.   ]

observation_noise_variance_champ:0 is 0.466

bias_mean:0 is 0.865

Iteration 51
Acquisition function convergence reached at iteration 77.
The final UCB loss was -6.24 with predicted mean of [0.158]
The next parameters to simulate from are [[0.    0.001 0.033 0.    0.026 0.037]]
The mean of the samples was 0.498
Iteration 52
Acquisition function convergence reached at iteration 98.
The final UCB loss was -6.887 with predicted mean of [0.202]
The next parameters to simulate from are [[0.    0.    0.033 0.052 0.071 0.   ]]
The mean of the samples was 0.983
Iteration 53
Acquisition function convergence reached at iteration 600.
The final UCB loss was -6.589 with predicted mean of [0.157]
The next parameters to simulate from are [[0.    0.    0.    0.    0.    0.071]]
The mean of the samples was 0.549
Iteration 54
Acquisition function convergence reached at iteration 79.
The final UCB loss was -5.86 with predicted mean of [-0.252]
The next parameters to simulate from are [[0.    0.526 0.033 0.015 0.    0.041]]
The mean of the samples was -0.197
Iteration 55
Acquisition function convergence reached at iteration 99.
The final UCB loss was -6.992 with predicted mean of [0.151]
The next parameters to simulate from are [[0.001 1.    0.    0.027 0.071 0.   ]]
The mean of the samples was 0.193
Iteration 56
Acquisition function convergence reached at iteration 120.
The final UCB loss was -6.119 with predicted mean of [-0.33]
The next parameters to simulate from are [[0.    1.    0.033 0.033 0.031 0.037]]
```

```
The mean of the samples was -0.427
Iteration 57
Acquisition function convergence reached at iteration 106.
The final UCB loss was -6.788 with predicted mean of [0.29]
The next parameters to simulate from are [[0.   0.   0.   0.05 0.   0.  ]]
The mean of the samples was 0.687
Iteration 58
Acquisition function convergence reached at iteration 470.
The final UCB loss was -6.55 with predicted mean of [0.537]
The next parameters to simulate from are [[1.  1.  0.  0.1 0.  0. ]]
The mean of the samples was 0.545
Iteration 59
Acquisition function convergence reached at iteration 475.
The final UCB loss was -7.151 with predicted mean of [0.133]
The next parameters to simulate from are [[1.    0.    0.    0.1   0.    0.071]]
The mean of the samples was 0.353
Hyperparameter convergence reached at iteration 1513.
The minimum predicted mean of the observed indices is -1.113 at the point [0.    0.496 0.033
Iteration 60
Acquisition function convergence reached at iteration 458.
The final UCB loss was -6.549 with predicted mean of [0.129]
The next parameters to simulate from are [[1.    0.    0.033 0.    0.    0.   ]]
The mean of the samples was 0.541
Iteration 61
Acquisition function convergence reached at iteration 362.
The final UCB loss was -6.518 with predicted mean of [0.134]
The next parameters to simulate from are [[1.    0.    0.    0.031 0.    0.   ]]
The mean of the samples was 0.373
Iteration 62
Acquisition function convergence reached at iteration 94.
The final UCB loss was -6.396 with predicted mean of [0.18]
The next parameters to simulate from are [[1.    0.001 0.    0.056 0.    0.071]]
The mean of the samples was 0.434
Iteration 63
Acquisition function convergence reached at iteration 65.
The final UCB loss was -6.444 with predicted mean of [0.143]
The next parameters to simulate from are [[1.    0.001 0.033 0.062 0.    0.   ]]
The mean of the samples was 0.331
Iteration 64
Acquisition function convergence reached at iteration 66.
The final UCB loss was -6.095 with predicted mean of [0.228]
The next parameters to simulate from are [[0.999 0.005 0.033 0.017 0.    0.071]]
The mean of the samples was 0.423
```

```
Iteration 65
Acquisition function convergence reached at iteration 122.
The final UCB loss was -5.706 with predicted mean of [0.252]
The next parameters to simulate from are [[0.    1.    0.    0.023 0.    0.036]]
The mean of the samples was 0.418
Iteration 66
Acquisition function convergence reached at iteration 93.
The final UCB loss was -6.531 with predicted mean of [0.108]
The next parameters to simulate from are [[1.    1.    0.    0.044 0.071 0.   ]]
The mean of the samples was 1.114
Iteration 67
Acquisition function convergence reached at iteration 110.
The final UCB loss was -6.067 with predicted mean of [0.031]
The next parameters to simulate from are [[0.431 0.001 0.017 0.033 0.    0.039]]
The mean of the samples was -0.099
Iteration 68
Acquisition function convergence reached at iteration 327.
The final UCB loss was -6.687 with predicted mean of [-0.033]
The next parameters to simulate from are [[0.001 0.    0.    0.037 0.071 0.071]]
The mean of the samples was 1.155
Iteration 69
Acquisition function convergence reached at iteration 285.
The final UCB loss was -5.974 with predicted mean of [-0.062]
The next parameters to simulate from are [[0.445 0.382 0.033 0.046 0.    0.   ]]
The mean of the samples was 0.31
Iteration 70
Acquisition function convergence reached at iteration 95.
The final UCB loss was -5.96 with predicted mean of [-0.434]
The next parameters to simulate from are [[0.001 0.494 0.02  0.04  0.    0.071]]
The mean of the samples was -0.119
Iteration 71
Acquisition function convergence reached at iteration 507.
The final UCB loss was -6.136 with predicted mean of [0.278]
The next parameters to simulate from are [[1.    1.    0.033 0.    0.    0.   ]]
The mean of the samples was 1.298
Iteration 72
Acquisition function convergence reached at iteration 73.
The final UCB loss was -5.865 with predicted mean of [0.078]
The next parameters to simulate from are [[0.999 0.682 0.033 0.031 0.    0.049]]
The mean of the samples was 0.457
Iteration 73
Acquisition function convergence reached at iteration 113.
The final UCB loss was -6.129 with predicted mean of [0.27]
```

```
The next parameters to simulate from are [[1.    1.    0.    0.07  0.026 0.071]]
The mean of the samples was 0.341
Iteration 74
Acquisition function convergence reached at iteration 311.
The final UCB loss was -5.937 with predicted mean of [-0.094]
The next parameters to simulate from are [[0.    1.    0.033 0.027 0.    0.   ]]
The mean of the samples was 0.267
Iteration 75
Acquisition function convergence reached at iteration 442.
The final UCB loss was -6.368 with predicted mean of [0.233]
The next parameters to simulate from are [[0.    0.    0.033 0.074 0.    0.071]]
The mean of the samples was 0.445
Iteration 76
Acquisition function convergence reached at iteration 460.
The final UCB loss was -6.721 with predicted mean of [0.1]
The next parameters to simulate from are [[1.    0.    0.033 0.    0.071 0.071]]
The mean of the samples was 0.549
Iteration 77
Acquisition function convergence reached at iteration 465.
The final UCB loss was -6.606 with predicted mean of [0.216]
The next parameters to simulate from are [[1.  0.  0.  0.1 0.  0. ]]
The mean of the samples was 0.599
Iteration 78
Acquisition function convergence reached at iteration 109.
The final UCB loss was -6.334 with predicted mean of [0.264]
The next parameters to simulate from are [[0.    1.    0.033 0.07  0.    0.   ]]
The mean of the samples was 1.458
Iteration 79
Acquisition function convergence reached at iteration 97.
The final UCB loss was -5.484 with predicted mean of [-0.141]
The next parameters to simulate from are [[0.438 1.    0.033 0.022 0.    0.037]]
The mean of the samples was -0.221
Hyperparameter convergence reached at iteration 1853.
The minimum predicted mean of the observed indices is -1.095 at the point [0.    0.496 0.033
Iteration 80
Acquisition function convergence reached at iteration 446.
The final UCB loss was -6.499 with predicted mean of [0.262]
The next parameters to simulate from are [[0.    0.    0.    0.1   0.071 0.   ]]
The mean of the samples was 2.454
Iteration 81
Acquisition function convergence reached at iteration 501.
The final UCB loss was -6.104 with predicted mean of [0.555]
The next parameters to simulate from are [[0.    0.    0.    0.1   0.071 0.071]]
```

```
The mean of the samples was 1.776
Iteration 82
Acquisition function convergence reached at iteration 96.
The final UCB loss was -6.148 with predicted mean of [0.445]
The next parameters to simulate from are [[0.    0.998 0.    0.063 0.071 0.   ]]
The mean of the samples was 1.068
Iteration 83
Acquisition function convergence reached at iteration 372.
The final UCB loss was -6.034 with predicted mean of [0.441]
The next parameters to simulate from are [[0.    1.    0.    0.086 0.    0.071]]
The mean of the samples was 0.58
Iteration 84
Acquisition function convergence reached at iteration 487.
The final UCB loss was -6.514 with predicted mean of [0.096]
The next parameters to simulate from are [[0.    0.    0.    0.    0.071 0.   ]]
The mean of the samples was 0.447
Iteration 85
Acquisition function convergence reached at iteration 106.
The final UCB loss was -6.512 with predicted mean of [0.166]
The next parameters to simulate from are [[1.    0.    0.    0.025 0.071 0.071]]
The mean of the samples was 0.476
Iteration 86
Acquisition function convergence reached at iteration 469.
The final UCB loss was -5.923 with predicted mean of [0.294]
The next parameters to simulate from are [[0.    1.    0.    0.046 0.071 0.071]]
The mean of the samples was 1.152
Iteration 87
Acquisition function convergence reached at iteration 108.
The final UCB loss was -5.966 with predicted mean of [-0.402]
The next parameters to simulate from are [[0.001 0.486 0.033 0.024 0.038 0.071]]
The mean of the samples was -0.465
Iteration 88
Acquisition function convergence reached at iteration 66.
The final UCB loss was -5.533 with predicted mean of [0.508]
The next parameters to simulate from are [[0.001 0.478 0.    0.064 0.    0.039]]
The mean of the samples was 0.238
Iteration 89
Acquisition function convergence reached at iteration 100.
The final UCB loss was -6.23 with predicted mean of [0.369]
The next parameters to simulate from are [[0.002 0.001 0.    0.037 0.071 0.   ]]
The mean of the samples was 0.452
Iteration 90
Acquisition function convergence reached at iteration 82.
```

```
The final UCB loss was -6.008 with predicted mean of [-0.217]
The next parameters to simulate from are [[0.002 0.26  0.023 0.03  0.045 0.033]]
The mean of the samples was -0.724
Iteration 91
Acquisition function convergence reached at iteration 89.
The final UCB loss was -5.593 with predicted mean of [-0.07]
The next parameters to simulate from are [[0.001 0.587 0.019 0.051 0.038 0.038]]
The mean of the samples was 0.232
Iteration 92
Acquisition function convergence reached at iteration 253.
The final UCB loss was -5.676 with predicted mean of [0.401]
The next parameters to simulate from are [[1.    1.    0.009 0.026 0.028 0.   ]]
The mean of the samples was 0.404
Iteration 93
Acquisition function convergence reached at iteration 561.
The final UCB loss was -6.034 with predicted mean of [0.28]
The next parameters to simulate from are [[1.    0.    0.    0.    0.071 0.071]]
The mean of the samples was 0.547
Iteration 94
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.783 with predicted mean of [0.408]
The next parameters to simulate from are [[1.    1.    0.015 0.071 0.043 0.   ]]
The mean of the samples was 0.177
Iteration 95
Acquisition function convergence reached at iteration 109.
The final UCB loss was -6.15 with predicted mean of [0.397]
The next parameters to simulate from are [[1.    1.    0.    0.037 0.071 0.071]]
The mean of the samples was 0.343
Iteration 96
Acquisition function convergence reached at iteration 504.
The final UCB loss was -6.161 with predicted mean of [0.329]
The next parameters to simulate from are [[1.    1.    0.    0.    0.    0.071]]
The mean of the samples was 0.547
Iteration 97
Acquisition function convergence reached at iteration 84.
The final UCB loss was -6.233 with predicted mean of [0.299]
The next parameters to simulate from are [[1.    0.001 0.    0.065 0.    0.   ]]
The mean of the samples was 0.375
Iteration 98
Acquisition function convergence reached at iteration 326.
The final UCB loss was -5.68 with predicted mean of [0.065]
The next parameters to simulate from are [[1.    1.    0.033 0.054 0.024 0.02 ]]
The mean of the samples was 1.815
```

```
Iteration 99
Acquisition function convergence reached at iteration 100.
The final UCB loss was -5.571 with predicted mean of [0.498]
The next parameters to simulate from are [[0.668 1.    0.    0.073 0.    0.031]]
The mean of the samples was 0.457
Hyperparameter convergence reached at iteration 1929.
The minimum predicted mean of the observed indices is -1.11 at the point [0.    0.496 0.033
Iteration 100
Acquisition function convergence reached at iteration 90.
The final UCB loss was -6.382 with predicted mean of [0.271]
The next parameters to simulate from are [[1.    0.    0.    0.028 0.    0.071]]
The mean of the samples was 0.424
Trained parameters:
amplitude_champ:0 is 0.853

length_scales_champ:0 is [1.    1.    1.    0.288 1.    1.   ]

observation_noise_variance_champ:0 is 0.448

bias_mean:0 is 0.858

Iteration 101
Acquisition function convergence reached at iteration 106.
The final UCB loss was -6.144 with predicted mean of [0.279]
The next parameters to simulate from are [[0.354 1.    0.    0.02  0.071 0.071]]
The mean of the samples was 0.354
Iteration 102
Acquisition function convergence reached at iteration 90.
The final UCB loss was -5.838 with predicted mean of [0.369]
The next parameters to simulate from are [[0.998 0.389 0.012 0.048 0.    0.032]]
The mean of the samples was 0.354
Iteration 103
Acquisition function convergence reached at iteration 120.
The final UCB loss was -6.712 with predicted mean of [0.171]
The next parameters to simulate from are [[1.    0.    0.    0.018 0.071 0.   ]]
The mean of the samples was 0.206
Iteration 104
Acquisition function convergence reached at iteration 66.
The final UCB loss was -6.284 with predicted mean of [0.266]
The next parameters to simulate from are [[1.    1.    0.033 0.016 0.071 0.071]]
The mean of the samples was 0.707
Iteration 105
Acquisition function convergence reached at iteration 410.
```

```
The final UCB loss was -6.414 with predicted mean of [0.186]
The next parameters to simulate from are [[0.    0.    0.    0.024 0.    0.   ]]
The mean of the samples was 0.233
Iteration 106
Acquisition function convergence reached at iteration 69.
The final UCB loss was -5.846 with predicted mean of [0.129]
The next parameters to simulate from are [[0.    0.57  0.014 0.024 0.    0.   ]]
The mean of the samples was 0.213
Iteration 107
Acquisition function convergence reached at iteration 80.
The final UCB loss was -6.016 with predicted mean of [0.338]
The next parameters to simulate from are [[1.    1.    0.    0.02  0.071 0.029]]
The mean of the samples was 0.418
Iteration 108
Acquisition function convergence reached at iteration 97.
The final UCB loss was -6.321 with predicted mean of [-0.015]
The next parameters to simulate from are [[0.    0.001 0.012 0.021 0.    0.071]]
The mean of the samples was 0.488
Iteration 109
Acquisition function convergence reached at iteration 106.
The final UCB loss was -5.75 with predicted mean of [-0.031]
The next parameters to simulate from are [[0.478 1.    0.014 0.039 0.    0.039]]
The mean of the samples was -0.129
Iteration 110
Acquisition function convergence reached at iteration 96.
The final UCB loss was -5.798 with predicted mean of [0.375]
The next parameters to simulate from are [[1.    1.    0.    0.018 0.033 0.071]]
The mean of the samples was 0.417
Iteration 111
Acquisition function convergence reached at iteration 98.
The final UCB loss was -6.723 with predicted mean of [0.113]
The next parameters to simulate from are [[1.    0.    0.033 0.016 0.071 0.   ]]
The mean of the samples was 0.403
Iteration 112
Acquisition function convergence reached at iteration 263.
The final UCB loss was -6.535 with predicted mean of [0.142]
The next parameters to simulate from are [[1.    0.    0.    0.054 0.071 0.071]]
The mean of the samples was 0.438
Iteration 113
Acquisition function convergence reached at iteration 426.
The final UCB loss was -6.426 with predicted mean of [-0.046]
The next parameters to simulate from are [[0.    1.    0.033 0.019 0.071 0.071]]
The mean of the samples was -2.124
```

```
Iteration 114
Acquisition function convergence reached at iteration 112.
The final UCB loss was -5.815 with predicted mean of [0.226]
The next parameters to simulate from are [[0.    0.534 0.    0.025 0.03  0.071]]
The mean of the samples was 0.427
Iteration 115
Acquisition function convergence reached at iteration 101.
The final UCB loss was -5.796 with predicted mean of [0.375]
The next parameters to simulate from are [[0.999 0.61  0.    0.051 0.038 0.071]]
The mean of the samples was 0.377
Iteration 116
Acquisition function convergence reached at iteration 495.
The final UCB loss was -6.626 with predicted mean of [0.193]
The next parameters to simulate from are [[0.    1.    0.033 0.    0.071 0.   ]]
The mean of the samples was 0.452
Iteration 117
Acquisition function convergence reached at iteration 109.
The final UCB loss was -6.457 with predicted mean of [-0.407]
The next parameters to simulate from are [[0.    0.998 0.019 0.017 0.071 0.04 ]]
The mean of the samples was -1.022
Iteration 118
Acquisition function convergence reached at iteration 108.
The final UCB loss was -6.455 with predicted mean of [-0.621]
The next parameters to simulate from are [[0.    0.503 0.018 0.018 0.071 0.071]]
The mean of the samples was -0.663
Iteration 119
Acquisition function convergence reached at iteration 102.
The final UCB loss was -6.217 with predicted mean of [-0.718]
The next parameters to simulate from are [[0.    1.    0.033 0.015 0.036 0.071]]
The mean of the samples was -0.261
Hyperparameter convergence reached at iteration 1892.
The minimum predicted mean of the observed indices is -1.539 at the point [0.    1.    0.033
Iteration 120
Acquisition function convergence reached at iteration 898.
The final UCB loss was -5.807 with predicted mean of [-0.621]
The next parameters to simulate from are [[0.    0.569 0.023 0.029 0.    0.042]]
The mean of the samples was -0.495
Iteration 121
Acquisition function convergence reached at iteration 125.
The final UCB loss was -6.612 with predicted mean of [0.387]
The next parameters to simulate from are [[1.    1.    0.    0.091 0.071 0.071]]
The mean of the samples was 0.357
Iteration 122
```

```
Acquisition function convergence reached at iteration 100.
The final UCB loss was -6.675 with predicted mean of [-0.511]
The next parameters to simulate from are [[0.326 0.477 0.033 0.022 0.071 0.034]]
The mean of the samples was -0.87
Iteration 123
Acquisition function convergence reached at iteration 105.
The final UCB loss was -6.287 with predicted mean of [0.088]
The next parameters to simulate from are [[0.534 1.    0.02  0.016 0.071 0.   ]]
The mean of the samples was 0.346
Iteration 124
Acquisition function convergence reached at iteration 107.
The final UCB loss was -6.203 with predicted mean of [-0.504]
The next parameters to simulate from are [[0.394 1.    0.025 0.028 0.071 0.052]]
The mean of the samples was -0.893
Iteration 125
Acquisition function convergence reached at iteration 123.
The final UCB loss was -6.94 with predicted mean of [0.218]
The next parameters to simulate from are [[1.    0.    0.033 0.075 0.071 0.071]]
The mean of the samples was 0.461
Iteration 126
Acquisition function convergence reached at iteration 70.
The final UCB loss was -5.859 with predicted mean of [0.499]
The next parameters to simulate from are [[1.    0.    0.022 0.    0.023 0.071]]
The mean of the samples was 0.549
Iteration 127
Acquisition function convergence reached at iteration 103.
The final UCB loss was -6.751 with predicted mean of [-0.104]
The next parameters to simulate from are [[0.    0.    0.033 0.025 0.071 0.   ]]
The mean of the samples was 0.242
Iteration 128
Acquisition function convergence reached at iteration 126.
The final UCB loss was -6.705 with predicted mean of [-0.066]
The next parameters to simulate from are [[0.462 0.    0.033 0.039 0.071 0.071]]
The mean of the samples was -0.506
Iteration 129
Acquisition function convergence reached at iteration 107.
The final UCB loss was -6.481 with predicted mean of [-0.386]
The next parameters to simulate from are [[0.479 0.    0.033 0.024 0.038 0.047]]
The mean of the samples was -0.816
Iteration 130
Acquisition function convergence reached at iteration 86.
The final UCB loss was -5.852 with predicted mean of [0.411]
The next parameters to simulate from are [[0.435 0.999 0.    0.036 0.043 0.028]]
```

```
The mean of the samples was -0.192
Iteration 131
Acquisition function convergence reached at iteration 547.
The final UCB loss was -6.484 with predicted mean of [0.039]
The next parameters to simulate from are [[1.    0.    0.033 0.023 0.071 0.071]]
The mean of the samples was 0.512
Iteration 132
Acquisition function convergence reached at iteration 113.
The final UCB loss was -6.545 with predicted mean of [0.509]
The next parameters to simulate from are [[0.    0.999 0.033 0.07  0.071 0.071]]
The mean of the samples was 1.309
Iteration 133
Acquisition function convergence reached at iteration 478.
The final UCB loss was -6.547 with predicted mean of [0.166]
The next parameters to simulate from are [[1.    0.    0.033 0.024 0.    0.   ]]
The mean of the samples was 0.412
Iteration 134
Acquisition function convergence reached at iteration 105.
The final UCB loss was -6.959 with predicted mean of [0.172]
The next parameters to simulate from are [[1.    0.    0.    0.071 0.071 0.   ]]
The mean of the samples was 0.103
Iteration 135
Acquisition function convergence reached at iteration 74.
The final UCB loss was -6.8 with predicted mean of [0.128]
The next parameters to simulate from are [[1.    0.    0.    0.044 0.071 0.   ]]
The mean of the samples was 0.218
Iteration 136
Acquisition function convergence reached at iteration 86.
The final UCB loss was -6.168 with predicted mean of [0.388]
The next parameters to simulate from are [[0.    1.    0.    0.068 0.032 0.071]]
The mean of the samples was 1.029
Iteration 137
Acquisition function convergence reached at iteration 483.
The final UCB loss was -7.009 with predicted mean of [0.258]
The next parameters to simulate from are [[1.    0.    0.033 0.1   0.071 0.   ]]
The mean of the samples was 0.558
Iteration 138
Acquisition function convergence reached at iteration 478.
The final UCB loss was -6.564 with predicted mean of [0.289]
The next parameters to simulate from are [[1.    0.    0.013 0.    0.071 0.   ]]
The mean of the samples was 0.549
Iteration 139
Acquisition function convergence reached at iteration 597.
```

```
The final UCB loss was -6.512 with predicted mean of [0.134]
The next parameters to simulate from are [[0.    0.    0.014 0.    0.071 0.071]]
The mean of the samples was 0.441
Hyperparameter convergence reached at iteration 1916.
The minimum predicted mean of the observed indices is -1.529 at the point [0.    1.    0.033
Iteration 140
Acquisition function convergence reached at iteration 98.
The final UCB loss was -6.797 with predicted mean of [0.123]
The next parameters to simulate from are [[1.    0.    0.033 0.064 0.071 0.   ]]
The mean of the samples was 0.332
Iteration 141
Acquisition function convergence reached at iteration 104.
The final UCB loss was -6.199 with predicted mean of [-0.353]
The next parameters to simulate from are [[0.408 0.474 0.033 0.05  0.    0.071]]
The mean of the samples was -0.213
Iteration 142
Acquisition function convergence reached at iteration 96.
The final UCB loss was -5.939 with predicted mean of [-0.24]
The next parameters to simulate from are [[0.498 0.491 0.033 0.015 0.047 0.071]]
The mean of the samples was -0.081
Iteration 143
Acquisition function convergence reached at iteration 394.
The final UCB loss was -6.321 with predicted mean of [0.049]
The next parameters to simulate from are [[1.    0.    0.033 0.044 0.071 0.031]]
The mean of the samples was 0.398
Iteration 144
Acquisition function convergence reached at iteration 79.
The final UCB loss was -6.361 with predicted mean of [-0.067]
The next parameters to simulate from are [[0.001 0.    0.033 0.053 0.    0.031]]
The mean of the samples was 0.264
Iteration 145
Acquisition function convergence reached at iteration 108.
The final UCB loss was -6.147 with predicted mean of [-0.647]
The next parameters to simulate from are [[0.001 0.    0.033 0.038 0.037 0.042]]
The mean of the samples was -0.411
Iteration 146
Acquisition function convergence reached at iteration 97.
The final UCB loss was -6.031 with predicted mean of [0.044]
The next parameters to simulate from are [[0.001 0.54  0.033 0.016 0.04  0.   ]]
The mean of the samples was 0.384
Iteration 147
Acquisition function convergence reached at iteration 93.
The final UCB loss was -6.045 with predicted mean of [-0.408]
```

```
The next parameters to simulate from are [[0.001 0.638 0.017 0.033 0.071 0.033]]
The mean of the samples was -0.2
Iteration 148
Acquisition function convergence reached at iteration 82.
The final UCB loss was -6.046 with predicted mean of [-0.816]
The next parameters to simulate from are [[0.379 0.487 0.033 0.036 0.033 0.04 ]]
The mean of the samples was -0.841
Iteration 149
Acquisition function convergence reached at iteration 34.
The final UCB loss was -6.666 with predicted mean of [0.349]
The next parameters to simulate from are [[0.999 0.997 0.033 0.095 0.    0.071]]
The mean of the samples was 0.202
Iteration 150
Acquisition function convergence reached at iteration 70.
The final UCB loss was -5.434 with predicted mean of [0.374]
The next parameters to simulate from are [[0.545 1.    0.    0.048 0.031 0.071]]
The mean of the samples was 0.38
Trained parameters:
amplitude_champ:0 is 0.849

length_scales_champ:0 is [1.    1.    1.    0.293 1.    1.   ]

observation_noise_variance_champ:0 is 0.412

bias_mean:0 is 0.853

Iteration 151
Acquisition function convergence reached at iteration 90.
The final UCB loss was -6.122 with predicted mean of [0.177]
The next parameters to simulate from are [[1.    0.702 0.033 0.029 0.071 0.   ]]
The mean of the samples was 0.314
Iteration 152
Acquisition function convergence reached at iteration 97.
The final UCB loss was -5.873 with predicted mean of [-0.454]
The next parameters to simulate from are [[0.358 1.    0.033 0.016 0.051 0.036]]
The mean of the samples was -0.221
Iteration 153
Acquisition function convergence reached at iteration 97.
The final UCB loss was -5.654 with predicted mean of [0.312]
The next parameters to simulate from are [[1.    1.    0.017 0.033 0.04  0.043]]
The mean of the samples was 1.551
Iteration 154
Acquisition function convergence reached at iteration 115.
```

```
The final UCB loss was -6.14 with predicted mean of [-0.029]
The next parameters to simulate from are [[0.532 0.    0.017 0.038 0.037 0.071]]
The mean of the samples was -0.592
Iteration 155
Acquisition function convergence reached at iteration 378.
The final UCB loss was -6.417 with predicted mean of [0.682]
The next parameters to simulate from are [[0.    1.    0.    0.1   0.071 0.071]]
The mean of the samples was 2.08
Iteration 156
Acquisition function convergence reached at iteration 390.
The final UCB loss was -6.724 with predicted mean of [0.285]
The next parameters to simulate from are [[1. 0. 0. 0. 0. 0.]]
The mean of the samples was 0.549
Iteration 157
Acquisition function convergence reached at iteration 420.
The final UCB loss was -6.172 with predicted mean of [-0.797]
The next parameters to simulate from are [[0.    0.486 0.033 0.032 0.071 0.071]]
The mean of the samples was -0.094
Iteration 158
Acquisition function convergence reached at iteration 109.
The final UCB loss was -6.007 with predicted mean of [-0.417]
The next parameters to simulate from are [[0.437 0.411 0.033 0.029 0.    0.071]]
The mean of the samples was 0.107
Iteration 159
Acquisition function convergence reached at iteration 102.
The final UCB loss was -6.187 with predicted mean of [0.383]
The next parameters to simulate from are [[0.281 1.    0.033 0.075 0.    0.071]]
The mean of the samples was 0.25
Hyperparameter convergence reached at iteration 6340.
The minimum predicted mean of the observed indices is -1.49 at the point [0.    1.    0.033
Iteration 160
Acquisition function convergence reached at iteration 100.
The final UCB loss was -6.02 with predicted mean of [0.351]
The next parameters to simulate from are [[0.548 0.    0.033 0.    0.042 0.   ]]
The mean of the samples was 0.449
Iteration 161
Acquisition function convergence reached at iteration 79.
The final UCB loss was -6.202 with predicted mean of [0.12]
The next parameters to simulate from are [[0.48  0.    0.    0.025 0.042 0.   ]]
The mean of the samples was 0.195
Iteration 162
Acquisition function convergence reached at iteration 116.
The final UCB loss was -6.141 with predicted mean of [0.05]
```

```
The next parameters to simulate from are [[0.537 0.001 0.022 0.036 0.071 0.   ]]
The mean of the samples was 0.541
Iteration 163
Acquisition function convergence reached at iteration 643.
The final UCB loss was -6.408 with predicted mean of [0.161]
The next parameters to simulate from are [[0.    1.    0.    0.    0.071 0.   ]]
The mean of the samples was 0.449
Iteration 164
Acquisition function convergence reached at iteration 99.
The final UCB loss was -6.152 with predicted mean of [0.42]
The next parameters to simulate from are [[0.334 0.    0.    0.063 0.071 0.   ]]
The mean of the samples was 0.608
Iteration 165
Acquisition function convergence reached at iteration 91.
The final UCB loss was -5.968 with predicted mean of [0.261]
The next parameters to simulate from are [[0.546 0.424 0.    0.048 0.027 0.   ]]
The mean of the samples was 0.486
Iteration 166
Acquisition function convergence reached at iteration 115.
The final UCB loss was -6.135 with predicted mean of [-0.092]
The next parameters to simulate from are [[0.425 0.    0.014 0.019 0.071 0.047]]
The mean of the samples was -1.391
Iteration 167
Acquisition function convergence reached at iteration 98.
The final UCB loss was -6.07 with predicted mean of [0.012]
The next parameters to simulate from are [[0.499 0.    0.    0.015 0.042 0.071]]
The mean of the samples was 0.067
Iteration 168
Acquisition function convergence reached at iteration 439.
The final UCB loss was -6.122 with predicted mean of [0.184]
The next parameters to simulate from are [[0.495 0.376 0.    0.    0.071 0.034]]
The mean of the samples was 0.473
Iteration 169
Acquisition function convergence reached at iteration 75.
The final UCB loss was -6.03 with predicted mean of [-0.395]
The next parameters to simulate from are [[0.54  0.411 0.012 0.033 0.071 0.047]]
The mean of the samples was -0.348
Iteration 170
Acquisition function convergence reached at iteration 94.
The final UCB loss was -5.804 with predicted mean of [0.009]
The next parameters to simulate from are [[0.24  0.325 0.012 0.017 0.071 0.001]]
The mean of the samples was 0.19
Iteration 171
```

```
Acquisition function convergence reached at iteration 119.
The final UCB loss was -5.895 with predicted mean of [-0.673]
The next parameters to simulate from are [[0.196 0.    0.02  0.022 0.044 0.071]]
The mean of the samples was -0.467
Iteration 172
Acquisition function convergence reached at iteration 100.
The final UCB loss was -5.767 with predicted mean of [-0.168]
The next parameters to simulate from are [[0.576 0.446 0.015 0.014 0.071 0.071]]
The mean of the samples was -0.045
Iteration 173
Acquisition function convergence reached at iteration 317.
The final UCB loss was -5.571 with predicted mean of [-0.568]
The next parameters to simulate from are [[0.442 0.584 0.033 0.029 0.071 0.071]]
The mean of the samples was -0.53
Iteration 174
Acquisition function convergence reached at iteration 91.
The final UCB loss was -5.743 with predicted mean of [-0.194]
The next parameters to simulate from are [[0.001 0.    0.014 0.016 0.04  0.025]]
The mean of the samples was -1.056
Iteration 175
Acquisition function convergence reached at iteration 533.
The final UCB loss was -5.74 with predicted mean of [0.099]
The next parameters to simulate from are [[0.    1.    0.021 0.    0.052 0.071]]
The mean of the samples was 0.429
Iteration 176
Acquisition function convergence reached at iteration 93.
The final UCB loss was -6.103 with predicted mean of [0.345]
The next parameters to simulate from are [[0.396 1.    0.033 0.054 0.071 0.   ]]
The mean of the samples was 0.425
Iteration 177
Acquisition function convergence reached at iteration 115.
The final UCB loss was -5.603 with predicted mean of [-0.96]
The next parameters to simulate from are [[0.3   0.576 0.02  0.029 0.041 0.056]]
The mean of the samples was -0.922
Iteration 178
Acquisition function convergence reached at iteration 116.
The final UCB loss was -5.677 with predicted mean of [-0.588]
The next parameters to simulate from are [[0.    1.    0.033 0.048 0.    0.034]]
The mean of the samples was -0.621
Iteration 179
Acquisition function convergence reached at iteration 97.
The final UCB loss was -5.876 with predicted mean of [0.64]
The next parameters to simulate from are [[0.997 1.    0.033 0.076 0.049 0.071]]
```

```
The mean of the samples was 0.644
Hyperparameter convergence reached at iteration 5052.
The minimum predicted mean of the observed indices is -1.478 at the point [0.    1.    0.033
Iteration 180
Acquisition function convergence reached at iteration 112.
The final UCB loss was -5.633 with predicted mean of [0.652]
The next parameters to simulate from are [[0.    1.    0.    0.069 0.    0.   ]]
The mean of the samples was 1.362
Iteration 181
Acquisition function convergence reached at iteration 257.
The final UCB loss was -5.26 with predicted mean of [-0.5]
The next parameters to simulate from are [[0.    1.    0.033 0.045 0.027 0.071]]
The mean of the samples was -0.415
Iteration 182
Acquisition function convergence reached at iteration 97.
The final UCB loss was -5.228 with predicted mean of [0.094]
The next parameters to simulate from are [[0.    1.    0.017 0.058 0.    0.046]]
The mean of the samples was -0.046
Iteration 183
Acquisition function convergence reached at iteration 91.
The final UCB loss was -5.4 with predicted mean of [0.3]
The next parameters to simulate from are [[0.412 0.999 0.013 0.041 0.071 0.   ]]
The mean of the samples was 0.251
Iteration 184
Acquisition function convergence reached at iteration 555.
The final UCB loss was -6.452 with predicted mean of [0.223]
The next parameters to simulate from are [[1.    0.    0.033 0.1   0.    0.071]]
The mean of the samples was 0.38
Iteration 185
Acquisition function convergence reached at iteration 120.
The final UCB loss was -5.493 with predicted mean of [0.234]
The next parameters to simulate from are [[0.53  0.478 0.033 0.02  0.    0.   ]]
The mean of the samples was 0.276
Iteration 186
Acquisition function convergence reached at iteration 432.
The final UCB loss was -6.355 with predicted mean of [0.363]
The next parameters to simulate from are [[0.    0.    0.033 0.1   0.    0.   ]]
The mean of the samples was 2.022
Iteration 187
Acquisition function convergence reached at iteration 451.
The final UCB loss was -5.914 with predicted mean of [0.535]
The next parameters to simulate from are [[0.    0.    0.033 0.069 0.071 0.071]]
The mean of the samples was 1.093
```

```
Iteration 188
Acquisition function convergence reached at iteration 400.
The final UCB loss was -5.348 with predicted mean of [0.364]
The next parameters to simulate from are [[1.    1.    0.033 0.078 0.    0.047]]
The mean of the samples was 1.31
Iteration 189
Acquisition function convergence reached at iteration 354.
The final UCB loss was -5.407 with predicted mean of [-0.1]
The next parameters to simulate from are [[0.    0.516 0.033 0.062 0.    0.049]]
The mean of the samples was -0.009
Iteration 190
Acquisition function convergence reached at iteration 130.
The final UCB loss was -6.148 with predicted mean of [0.325]
The next parameters to simulate from are [[0.999 0.    0.033 0.074 0.    0.071]]
The mean of the samples was 0.398
Iteration 191
Acquisition function convergence reached at iteration 67.
The final UCB loss was -5.288 with predicted mean of [0.549]
The next parameters to simulate from are [[0.585 0.481 0.033 0.    0.    0.035]]
The mean of the samples was 0.549
Iteration 192
Acquisition function convergence reached at iteration 294.
The final UCB loss was -5.72 with predicted mean of [0.858]
The next parameters to simulate from are [[0.    1.    0.033 0.1   0.    0.   ]]
The mean of the samples was 2.44
Iteration 193
Acquisition function convergence reached at iteration 118.
The final UCB loss was -5.293 with predicted mean of [-0.475]
The next parameters to simulate from are [[0.412 1.    0.025 0.044 0.    0.071]]
The mean of the samples was -0.121
Iteration 194
Acquisition function convergence reached at iteration 118.
The final UCB loss was -5.541 with predicted mean of [-0.226]
The next parameters to simulate from are [[0.505 0.    0.033 0.045 0.    0.039]]
The mean of the samples was -0.873
Iteration 195
Acquisition function convergence reached at iteration 249.
The final UCB loss was -5.518 with predicted mean of [0.377]
The next parameters to simulate from are [[1.    0.332 0.    0.016 0.027 0.029]]
The mean of the samples was 0.431
Iteration 196
Acquisition function convergence reached at iteration 271.
The final UCB loss was -5.274 with predicted mean of [0.171]
```

```
The next parameters to simulate from are [[0.    1.    0.016 0.018 0.    0.071]]
The mean of the samples was 0.24
Iteration 197
Acquisition function convergence reached at iteration 89.
The final UCB loss was -5.05 with predicted mean of [-0.416]
The next parameters to simulate from are [[0.    1.    0.02  0.038 0.    0.044]]
The mean of the samples was -0.685
Iteration 198
Acquisition function convergence reached at iteration 98.
The final UCB loss was -4.839 with predicted mean of [-0.275]
The next parameters to simulate from are [[0.    1.    0.033 0.022 0.    0.042]]
The mean of the samples was -0.335
Iteration 199
Acquisition function convergence reached at iteration 86.
The final UCB loss was -6.204 with predicted mean of [0.286]
The next parameters to simulate from are [[1.    1.    0.033 0.086 0.071 0.   ]]
The mean of the samples was 0.039
Hyperparameter convergence reached at iteration 2003.
The minimum predicted mean of the observed indices is -1.481 at the point [0.    1.    0.033
Iteration 200
Acquisition function convergence reached at iteration 482.
The final UCB loss was -6.409 with predicted mean of [0.399]
The next parameters to simulate from are [[1.    0.    0.033 0.1   0.    0.   ]]
The mean of the samples was 0.875
Trained parameters:
amplitude_champ:0 is 0.802

length_scales_champ:0 is [0.95  1.    1.    0.311 0.996 1.   ]

observation_noise_variance_champ:0 is 0.433

bias_mean:0 is 0.883

Iteration 201
Acquisition function convergence reached at iteration 372.
The final UCB loss was -5.648 with predicted mean of [0.422]
The next parameters to simulate from are [[0.459 0.    0.013 0.    0.    0.03 ]]
The mean of the samples was 0.549
Iteration 202
Acquisition function convergence reached at iteration 420.
The final UCB loss was -5.812 with predicted mean of [0.432]
The next parameters to simulate from are [[1.    0.    0.033 0.081 0.035 0.   ]]
The mean of the samples was 0.618
```

```
Iteration 203
Acquisition function convergence reached at iteration 507.
The final UCB loss was -6.243 with predicted mean of [0.453]
The next parameters to simulate from are [[1.    0.    0.033 0.1   0.071 0.071]]
The mean of the samples was 0.354
Iteration 204
Acquisition function convergence reached at iteration 92.
The final UCB loss was -5.307 with predicted mean of [0.403]
The next parameters to simulate from are [[0.378 1.    0.033 0.058 0.035 0.046]]
The mean of the samples was 0.309
Iteration 205
Acquisition function convergence reached at iteration 351.
The final UCB loss was -5.455 with predicted mean of [0.513]
The next parameters to simulate from are [[0.501 0.512 0.    0.    0.    0.071]]
The mean of the samples was 0.547
Iteration 206
Acquisition function convergence reached at iteration 81.
The final UCB loss was -6.366 with predicted mean of [0.316]
The next parameters to simulate from are [[0.    0.    0.    0.081 0.    0.071]]
The mean of the samples was 0.246
Iteration 207
Acquisition function convergence reached at iteration 116.
The final UCB loss was -5.36 with predicted mean of [0.46]
The next parameters to simulate from are [[0.491 1.    0.    0.    0.038 0.071]]
The mean of the samples was 0.438
Iteration 208
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.791 with predicted mean of [0.167]
The next parameters to simulate from are [[0.713 0.    0.033 0.056 0.044 0.071]]
The mean of the samples was -0.443
Iteration 209
Acquisition function convergence reached at iteration 102.
The final UCB loss was -6.073 with predicted mean of [0.289]
The next parameters to simulate from are [[0.999 0.328 0.    0.078 0.    0.071]]
The mean of the samples was 0.362
Iteration 210
Acquisition function convergence reached at iteration 97.
The final UCB loss was -5.579 with predicted mean of [-0.244]
The next parameters to simulate from are [[0.    0.    0.033 0.014 0.027 0.071]]
The mean of the samples was -0.146
Iteration 211
Acquisition function convergence reached at iteration 375.
The final UCB loss was -5.61 with predicted mean of [-0.434]
```

```
The next parameters to simulate from are [[0.    0.    0.033 0.015 0.071 0.038]]
The mean of the samples was -0.399
Iteration 212
Acquisition function convergence reached at iteration 79.
The final UCB loss was -5.742 with predicted mean of [0.125]
The next parameters to simulate from are [[0.001 0.    0.021 0.041 0.023 0.   ]]
The mean of the samples was 0.408
Iteration 213
Acquisition function convergence reached at iteration 68.
The final UCB loss was -5.937 with predicted mean of [0.073]
The next parameters to simulate from are [[0.398 0.    0.017 0.06  0.    0.071]]
The mean of the samples was -0.254
Iteration 214
Acquisition function convergence reached at iteration 68.
The final UCB loss was -5.361 with predicted mean of [-0.363]
The next parameters to simulate from are [[0.001 0.    0.033 0.025 0.    0.04 ]]
The mean of the samples was -0.421
Iteration 215
Acquisition function convergence reached at iteration 96.
The final UCB loss was -5.776 with predicted mean of [0.329]
The next parameters to simulate from are [[0.999 0.546 0.033 0.074 0.071 0.031]]
The mean of the samples was 0.412
Iteration 216
Acquisition function convergence reached at iteration 87.
The final UCB loss was -6.045 with predicted mean of [0.655]
The next parameters to simulate from are [[0.    0.    0.    0.082 0.    0.   ]]
The mean of the samples was 1.447
Iteration 217
Acquisition function convergence reached at iteration 296.
The final UCB loss was -5.586 with predicted mean of [0.106]
The next parameters to simulate from are [[0.506 0.    0.013 0.056 0.    0.026]]
The mean of the samples was 0.642
Iteration 218
Acquisition function convergence reached at iteration 92.
The final UCB loss was -5.612 with predicted mean of [0.285]
The next parameters to simulate from are [[0.516 0.417 0.    0.04  0.    0.071]]
The mean of the samples was 0.442
Iteration 219
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.387 with predicted mean of [0.553]
The next parameters to simulate from are [[1.    0.65  0.033 0.    0.048 0.071]]
The mean of the samples was 0.512
Hyperparameter convergence reached at iteration 2178.
```

```
The minimum predicted mean of the observed indices is -1.465 at the point [0.    1.    0.033
Iteration 220
Acquisition function convergence reached at iteration 134.
The final UCB loss was -5.385 with predicted mean of [-1.058]
The next parameters to simulate from are [[0.    0.814 0.033 0.023 0.071 0.043]]
The mean of the samples was -1.171
Iteration 221
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.339 with predicted mean of [0.096]
The next parameters to simulate from are [[0.524 0.    0.02  0.019 0.029 0.   ]]
The mean of the samples was 0.231
Iteration 222
Acquisition function convergence reached at iteration 397.
The final UCB loss was -5.82 with predicted mean of [0.701]
The next parameters to simulate from are [[0.    0.    0.    0.067 0.071 0.071]]
The mean of the samples was 1.424
Iteration 223
Acquisition function convergence reached at iteration 95.
The final UCB loss was -5.568 with predicted mean of [-0.04]
The next parameters to simulate from are [[0.506 0.001 0.033 0.056 0.039 0.028]]
The mean of the samples was -0.791
Iteration 224
Acquisition function convergence reached at iteration 406.
The final UCB loss was -6.357 with predicted mean of [0.393]
The next parameters to simulate from are [[1.    0.    0.    0.1   0.071 0.071]]
The mean of the samples was 0.448
Iteration 225
Acquisition function convergence reached at iteration 106.
The final UCB loss was -5.801 with predicted mean of [0.32]
The next parameters to simulate from are [[1.    0.    0.    0.077 0.071 0.071]]
The mean of the samples was 0.377
Iteration 226
Acquisition function convergence reached at iteration 115.
The final UCB loss was -5.211 with predicted mean of [-0.329]
The next parameters to simulate from are [[0.462 0.    0.033 0.036 0.034 0.014]]
The mean of the samples was -0.355
Iteration 227
Acquisition function convergence reached at iteration 117.
The final UCB loss was -5.566 with predicted mean of [0.058]
The next parameters to simulate from are [[1.    0.    0.011 0.038 0.041 0.039]]
The mean of the samples was 0.426
Iteration 228
Acquisition function convergence reached at iteration 117.
```

```
The final UCB loss was -5.007 with predicted mean of [-0.647]
The next parameters to simulate from are [[0.427 0.528 0.033 0.034 0.    0.031]]
The mean of the samples was -0.491
Iteration 229
Acquisition function convergence reached at iteration 118.
The final UCB loss was -5.299 with predicted mean of [-0.313]
The next parameters to simulate from are [[0.    0.    0.019 0.047 0.    0.051]]
The mean of the samples was -0.516
Iteration 230
Acquisition function convergence reached at iteration 86.
The final UCB loss was -5.69 with predicted mean of [-0.163]
The next parameters to simulate from are [[0.    0.    0.    0.019 0.071 0.035]]
The mean of the samples was 0.372
Iteration 231
Acquisition function convergence reached at iteration 219.
The final UCB loss was -4.993 with predicted mean of [-0.756]
The next parameters to simulate from are [[0.    0.472 0.033 0.038 0.019 0.071]]
The mean of the samples was -0.605
Iteration 232
Acquisition function convergence reached at iteration 56.
The final UCB loss was -4.872 with predicted mean of [0.32]
The next parameters to simulate from are [[0.541 0.998 0.033 0.013 0.    0.071]]
The mean of the samples was 0.382
Iteration 233
Acquisition function convergence reached at iteration 638.
The final UCB loss was -6.118 with predicted mean of [0.514]
The next parameters to simulate from are [[1.    0.    0.    0.1   0.071 0.   ]]
The mean of the samples was 0.585
Iteration 234
Acquisition function convergence reached at iteration 88.
The final UCB loss was -5.242 with predicted mean of [0.564]
The next parameters to simulate from are [[1.    0.512 0.015 0.016 0.    0.071]]
The mean of the samples was 0.524
Iteration 235
Acquisition function convergence reached at iteration 109.
The final UCB loss was -5.279 with predicted mean of [0.182]
The next parameters to simulate from are [[0.    0.545 0.033 0.042 0.04  0.   ]]
The mean of the samples was 0.715
Iteration 236
Acquisition function convergence reached at iteration 79.
The final UCB loss was -5.365 with predicted mean of [0.579]
The next parameters to simulate from are [[0.999 1.    0.014 0.1   0.036 0.071]]
The mean of the samples was 0.373
```

```
Iteration 237
Acquisition function convergence reached at iteration 108.
The final UCB loss was -5.373 with predicted mean of [0.333]
The next parameters to simulate from are [[0.    1.    0.    0.    0.037 0.033]]
The mean of the samples was 0.473
Iteration 238
Acquisition function convergence reached at iteration 380.
The final UCB loss was -5.478 with predicted mean of [0.638]
The next parameters to simulate from are [[0.521 0.    0.033 0.086 0.029 0.071]]
The mean of the samples was 0.429
Iteration 239
Acquisition function convergence reached at iteration 133.
The final UCB loss was -5.348 with predicted mean of [0.222]
The next parameters to simulate from are [[0.643 0.356 0.033 0.065 0.    0.039]]
The mean of the samples was -0.482
Hyperparameter convergence reached at iteration 1983.
The minimum predicted mean of the observed indices is -1.435 at the point [0.    1.    0.033
Iteration 240
Acquisition function convergence reached at iteration 85.
The final UCB loss was -4.945 with predicted mean of [0.067]
The next parameters to simulate from are [[0.539 0.    0.033 0.019 0.    0.033]]
The mean of the samples was 0.066
Iteration 241
Acquisition function convergence reached at iteration 93.
The final UCB loss was -5.563 with predicted mean of [0.131]
The next parameters to simulate from are [[0.479 0.    0.033 0.    0.071 0.047]]
The mean of the samples was 0.512
Iteration 242
Acquisition function convergence reached at iteration 117.
The final UCB loss was -5.236 with predicted mean of [0.177]
The next parameters to simulate from are [[0.001 0.    0.    0.034 0.028 0.036]]
The mean of the samples was -0.159
Iteration 243
Acquisition function convergence reached at iteration 93.
The final UCB loss was -5.752 with predicted mean of [0.188]
The next parameters to simulate from are [[0.536 0.999 0.    0.024 0.    0.   ]]
The mean of the samples was 0.283
Iteration 244
Acquisition function convergence reached at iteration 98.
The final UCB loss was -4.903 with predicted mean of [0.218]
The next parameters to simulate from are [[0.    0.    0.    0.016 0.    0.038]]
The mean of the samples was 0.454
Iteration 245
```

```
Acquisition function convergence reached at iteration 272.
The final UCB loss was -5.344 with predicted mean of [0.368]
The next parameters to simulate from are [[0.542 0.456 0.033 0.061 0.071 0.071]]
The mean of the samples was 0.166
Iteration 246
Acquisition function convergence reached at iteration 116.
The final UCB loss was -5.071 with predicted mean of [-0.436]
The next parameters to simulate from are [[0.694 0.399 0.033 0.049 0.03  0.046]]
The mean of the samples was -0.623
Iteration 247
Acquisition function convergence reached at iteration 395.
The final UCB loss was -5.706 with predicted mean of [0.407]
The next parameters to simulate from are [[0.383 0.    0.    0.1   0.    0.071]]
The mean of the samples was 0.318
Iteration 248
Acquisition function convergence reached at iteration 45.
The final UCB loss was -5.471 with predicted mean of [0.485]
The next parameters to simulate from are [[0.435 1.    0.    0.1   0.    0.071]]
The mean of the samples was 0.666
Iteration 249
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.159 with predicted mean of [0.397]
The next parameters to simulate from are [[0.999 1.    0.    0.023 0.    0.03 ]]
The mean of the samples was 0.46
Iteration 250
Acquisition function convergence reached at iteration 120.
The final UCB loss was -5.114 with predicted mean of [0.217]
The next parameters to simulate from are [[0.458 0.396 0.    0.031 0.    0.021]]
The mean of the samples was 0.398
Trained parameters:
amplitude_champ:0 is 0.756

length_scales_champ:0 is [0.831 1.    1.    0.356 1.    1.   ]

observation_noise_variance_champ:0 is 0.429

bias_mean:0 is 0.9

Iteration 251
Acquisition function convergence reached at iteration 85.
The final UCB loss was -5.328 with predicted mean of [0.121]
The next parameters to simulate from are [[0.    1.    0.014 0.02  0.037 0.    ]]
The mean of the samples was 0.326
```

```
Iteration 252
Acquisition function convergence reached at iteration 439.
The final UCB loss was -5.242 with predicted mean of [0.429]
The next parameters to simulate from are [[0.513 1.    0.    0.    0.    0.028]]
The mean of the samples was 0.545
Iteration 253
Acquisition function convergence reached at iteration 116.
The final UCB loss was -5.349 with predicted mean of [0.165]
The next parameters to simulate from are [[0.534 0.    0.    0.041 0.071 0.036]]
The mean of the samples was 0.061
Iteration 254
Acquisition function convergence reached at iteration 412.
The final UCB loss was -5.572 with predicted mean of [0.538]
The next parameters to simulate from are [[0.702 1.    0.    0.071 0.071 0.   ]]
The mean of the samples was 0.307
Iteration 255
Acquisition function convergence reached at iteration 82.
The final UCB loss was -4.904 with predicted mean of [0.374]
The next parameters to simulate from are [[0.4   1.    0.    0.025 0.    0.071]]
The mean of the samples was 0.438
Iteration 256
Acquisition function convergence reached at iteration 112.
The final UCB loss was -5.383 with predicted mean of [0.336]
The next parameters to simulate from are [[1.    0.613 0.013 0.07  0.071 0.071]]
The mean of the samples was 0.364
Iteration 257
Acquisition function convergence reached at iteration 369.
The final UCB loss was -5.337 with predicted mean of [0.212]
The next parameters to simulate from are [[0.522 0.    0.    0.072 0.    0.071]]
The mean of the samples was 0.55
Iteration 258
Acquisition function convergence reached at iteration 77.
The final UCB loss was -5.679 with predicted mean of [0.767]
The next parameters to simulate from are [[0.345 0.    0.033 0.086 0.071 0.   ]]
The mean of the samples was 1.678
Iteration 259
Acquisition function convergence reached at iteration 93.
The final UCB loss was -5.285 with predicted mean of [0.485]
The next parameters to simulate from are [[1.    0.241 0.016 0.1   0.034 0.071]]
The mean of the samples was 0.418
Hyperparameter convergence reached at iteration 2034.
The minimum predicted mean of the observed indices is -1.433 at the point [0.    1.    0.033
Iteration 260
```

```
Acquisition function convergence reached at iteration 110.
The final UCB loss was -5.001 with predicted mean of [0.116]
The next parameters to simulate from are [[0.585 0.481 0.033 0.07  0.025 0.071]]
The mean of the samples was 0.021
Iteration 261
Acquisition function convergence reached at iteration 104.
The final UCB loss was -5.393 with predicted mean of [0.414]
The next parameters to simulate from are [[1.    0.502 0.014 0.08  0.    0.   ]]
The mean of the samples was 0.443
Iteration 262
Acquisition function convergence reached at iteration 98.
The final UCB loss was -4.998 with predicted mean of [0.063]
The next parameters to simulate from are [[0.    1.    0.    0.021 0.071 0.041]]
The mean of the samples was 0.462
Iteration 263
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.077 with predicted mean of [0.616]
The next parameters to simulate from are [[0.609 0.516 0.    0.086 0.041 0.071]]
The mean of the samples was 0.447
Iteration 264
Acquisition function convergence reached at iteration 324.
The final UCB loss was -5.145 with predicted mean of [0.351]
The next parameters to simulate from are [[1.    0.418 0.019 0.014 0.071 0.034]]
The mean of the samples was 0.442
Iteration 265
Acquisition function convergence reached at iteration 101.
The final UCB loss was -4.99 with predicted mean of [-0.776]
The next parameters to simulate from are [[0.2   0.349 0.02  0.013 0.054 0.04 ]]
The mean of the samples was -0.303
Iteration 266
Acquisition function convergence reached at iteration 89.
The final UCB loss was -5.112 with predicted mean of [0.1]
The next parameters to simulate from are [[0.001 0.    0.022 0.041 0.071 0.071]]
The mean of the samples was 0.363
Iteration 267
Acquisition function convergence reached at iteration 93.
The final UCB loss was -4.852 with predicted mean of [0.571]
The next parameters to simulate from are [[0.54  1.    0.017 0.082 0.031 0.071]]
The mean of the samples was 0.293
Iteration 268
Acquisition function convergence reached at iteration 113.
The final UCB loss was -4.879 with predicted mean of [0.442]
The next parameters to simulate from are [[0.    0.    0.015 0.064 0.026 0.071]]
```

```
The mean of the samples was 0.628
Iteration 269
Acquisition function convergence reached at iteration 458.
The final UCB loss was -5.37 with predicted mean of [0.666]
The next parameters to simulate from are [[0.549 1.    0.033 0.1   0.071 0.   ]]
The mean of the samples was 1.342
Iteration 270
Acquisition function convergence reached at iteration 96.
The final UCB loss was -5.377 with predicted mean of [0.274]
The next parameters to simulate from are [[1.    1.    0.    0.064 0.071 0.041]]
The mean of the samples was 0.362
Iteration 271
Acquisition function convergence reached at iteration 101.
The final UCB loss was -4.918 with predicted mean of [0.362]
The next parameters to simulate from are [[0.538 1.    0.    0.058 0.037 0.025]]
The mean of the samples was 0.145
Iteration 272
Acquisition function convergence reached at iteration 442.
The final UCB loss was -5.559 with predicted mean of [0.655]
The next parameters to simulate from are [[0.    0.    0.033 0.1   0.    0.071]]
The mean of the samples was 1.572
Iteration 273
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.203 with predicted mean of [-0.651]
The next parameters to simulate from are [[0.371 0.    0.033 0.041 0.026 0.071]]
The mean of the samples was -0.579
Iteration 274
Acquisition function convergence reached at iteration 98.
The final UCB loss was -5.223 with predicted mean of [0.335]
The next parameters to simulate from are [[1.    0.474 0.015 0.078 0.071 0.   ]]
The mean of the samples was 0.402
Iteration 275
Acquisition function convergence reached at iteration 90.
The final UCB loss was -5.325 with predicted mean of [0.416]
The next parameters to simulate from are [[0.999 0.001 0.    0.082 0.032 0.033]]
The mean of the samples was 0.365
Iteration 276
Acquisition function convergence reached at iteration 376.
The final UCB loss was -5.419 with predicted mean of [1.056]
The next parameters to simulate from are [[0.    1.    0.    0.1   0.071 0.   ]]
The mean of the samples was 2.493
Iteration 277
Acquisition function convergence reached at iteration 115.
```

```
The final UCB loss was -5.073 with predicted mean of [-0.423]
The next parameters to simulate from are [[0.449 0.473 0.018 0.05  0.019 0.043]]
The mean of the samples was -0.788
Iteration 278
Acquisition function convergence reached at iteration 101.
The final UCB loss was -5.214 with predicted mean of [0.399]
The next parameters to simulate from are [[0.603 1.    0.019 0.058 0.    0.   ]]
The mean of the samples was 0.198
Iteration 279
Acquisition function convergence reached at iteration 83.
The final UCB loss was -5.184 with predicted mean of [0.318]
The next parameters to simulate from are [[0.459 1.    0.015 0.    0.071 0.034]]
The mean of the samples was 0.457
Hyperparameter convergence reached at iteration 1990.
The minimum predicted mean of the observed indices is -1.422 at the point [0.    1.    0.033
Iteration 280
Acquisition function convergence reached at iteration 80.
The final UCB loss was -4.766 with predicted mean of [0.853]
The next parameters to simulate from are [[0.698 1.    0.    0.1   0.036 0.037]]
The mean of the samples was 0.775
Iteration 281
Acquisition function convergence reached at iteration 100.
The final UCB loss was -5.02 with predicted mean of [0.221]
The next parameters to simulate from are [[0.    0.45  0.    0.015 0.037 0.   ]]
The mean of the samples was 0.334
Iteration 282
Acquisition function convergence reached at iteration 96.
The final UCB loss was -4.897 with predicted mean of [0.237]
The next parameters to simulate from are [[0.504 1.    0.016 0.048 0.071 0.046]]
The mean of the samples was -0.076
Iteration 283
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.081 with predicted mean of [0.251]
The next parameters to simulate from are [[0.62  0.578 0.    0.027 0.071 0.   ]]
The mean of the samples was 0.216
Iteration 284
Acquisition function convergence reached at iteration 106.
The final UCB loss was -5.064 with predicted mean of [0.441]
The next parameters to simulate from are [[0.542 0.546 0.033 0.    0.071 0.   ]]
The mean of the samples was 0.447
Iteration 285
Acquisition function convergence reached at iteration 57.
The final UCB loss was -5.097 with predicted mean of [0.233]
```

```
The next parameters to simulate from are [[0.997 0.001 0.016 0.056 0.035 0.   ]]
The mean of the samples was 0.195
Iteration 286
Acquisition function convergence reached at iteration 98.
The final UCB loss was -4.945 with predicted mean of [0.452]
The next parameters to simulate from are [[1.    0.498 0.016 0.018 0.    0.   ]]
The mean of the samples was 0.417
Iteration 287
Acquisition function convergence reached at iteration 121.
The final UCB loss was -4.956 with predicted mean of [0.133]
The next parameters to simulate from are [[0.839 0.462 0.017 0.063 0.    0.071]]
The mean of the samples was 0.139
Iteration 288
Acquisition function convergence reached at iteration 100.
The final UCB loss was -5.01 with predicted mean of [0.204]
The next parameters to simulate from are [[0.66  0.428 0.033 0.05  0.071 0.   ]]
The mean of the samples was 0.466
Iteration 289
Acquisition function convergence reached at iteration 351.
The final UCB loss was -5.005 with predicted mean of [-0.799]
The next parameters to simulate from are [[0.    1.    0.019 0.023 0.071 0.071]]
The mean of the samples was -0.457
Iteration 290
Acquisition function convergence reached at iteration 117.
The final UCB loss was -4.929 with predicted mean of [-0.438]
The next parameters to simulate from are [[0.459 0.515 0.019 0.033 0.039 0.018]]
The mean of the samples was -1.1
Iteration 291
Acquisition function convergence reached at iteration 73.
The final UCB loss was -4.848 with predicted mean of [0.513]
The next parameters to simulate from are [[0.659 1.    0.022 0.073 0.071 0.029]]
The mean of the samples was 0.488
Iteration 292
Acquisition function convergence reached at iteration 83.
The final UCB loss was -4.967 with predicted mean of [0.323]
The next parameters to simulate from are [[1.    0.532 0.    0.065 0.038 0.   ]]
The mean of the samples was 0.534
Iteration 293
Acquisition function convergence reached at iteration 110.
The final UCB loss was -5.153 with predicted mean of [-0.794]
The next parameters to simulate from are [[0.313 0.    0.026 0.031 0.071 0.039]]
The mean of the samples was -0.467
Iteration 294
```

```
Acquisition function convergence reached at iteration 98.
The final UCB loss was -5.056 with predicted mean of [0.448]
The next parameters to simulate from are [[0.52  1.    0.    0.    0.041 0.   ]]
The mean of the samples was 0.407
Iteration 295
Acquisition function convergence reached at iteration 94.
The final UCB loss was -4.938 with predicted mean of [-0.003]
The next parameters to simulate from are [[0.537 1.    0.021 0.03  0.    0.   ]]
The mean of the samples was 0.226
Iteration 296
Acquisition function convergence reached at iteration 104.
The final UCB loss was -4.899 with predicted mean of [0.495]
The next parameters to simulate from are [[0.644 0.564 0.033 0.069 0.032 0.   ]]
The mean of the samples was 0.487
Iteration 297
Acquisition function convergence reached at iteration 103.
The final UCB loss was -4.901 with predicted mean of [0.253]
The next parameters to simulate from are [[0.463 0.569 0.    0.053 0.071 0.036]]
The mean of the samples was 0.698
Iteration 298
Acquisition function convergence reached at iteration 77.
The final UCB loss was -4.811 with predicted mean of [0.27]
The next parameters to simulate from are [[1.    0.497 0.016 0.043 0.071 0.   ]]
The mean of the samples was 0.368
Iteration 299
Acquisition function convergence reached at iteration 88.
The final UCB loss was -4.961 with predicted mean of [-0.31]
The next parameters to simulate from are [[0.39  0.578 0.033 0.041 0.071 0.036]]
The mean of the samples was -0.274
Hyperparameter convergence reached at iteration 2016.
The minimum predicted mean of the observed indices is -1.385 at the point [0.    1.    0.033
Iteration 300
Acquisition function convergence reached at iteration 86.
The final UCB loss was -4.968 with predicted mean of [-0.2]
The next parameters to simulate from are [[0.647 0.    0.033 0.022 0.071 0.033]]
The mean of the samples was -0.688
Trained parameters:
amplitude_champ:0 is 0.731

length_scales_champ:0 is [0.784 1.    1.    0.356 1.    1.   ]

observation_noise_variance_champ:0 is 0.411
```

```
bias_mean:0 is 0.871

Iteration 301
Acquisition function convergence reached at iteration 98.
The final UCB loss was -5.007 with predicted mean of [-0.296]
The next parameters to simulate from are [[0.674 0.499 0.033 0.021 0.041 0.027]]
The mean of the samples was -0.132
Iteration 302
Acquisition function convergence reached at iteration 99.
The final UCB loss was -5.281 with predicted mean of [0.399]
The next parameters to simulate from are [[0.532 0.    0.    0.    0.04  0.   ]]
The mean of the samples was 0.452
Iteration 303
Acquisition function convergence reached at iteration 110.
The final UCB loss was -4.732 with predicted mean of [-0.597]
The next parameters to simulate from are [[0.386 0.686 0.016 0.021 0.071 0.033]]
The mean of the samples was -1.225
Iteration 304
Acquisition function convergence reached at iteration 98.
The final UCB loss was -4.839 with predicted mean of [0.156]
The next parameters to simulate from are [[0.686 1.    0.016 0.023 0.071 0.071]]
The mean of the samples was 0.164
Iteration 305
Acquisition function convergence reached at iteration 93.
The final UCB loss was -4.88 with predicted mean of [0.012]
The next parameters to simulate from are [[0.689 1.    0.033 0.028 0.071 0.032]]
The mean of the samples was -0.735
Iteration 306
Acquisition function convergence reached at iteration 264.
The final UCB loss was -5.723 with predicted mean of [0.31]
The next parameters to simulate from are [[1.    1.    0.033 0.1   0.    0.   ]]
The mean of the samples was 0.352
Iteration 307
Acquisition function convergence reached at iteration 275.
The final UCB loss was -4.662 with predicted mean of [0.519]
The next parameters to simulate from are [[0.673 0.514 0.011 0.    0.071 0.   ]]
The mean of the samples was 0.437
Iteration 308
Acquisition function convergence reached at iteration 109.
The final UCB loss was -4.874 with predicted mean of [0.732]
The next parameters to simulate from are [[0.615 1.    0.016 0.085 0.    0.   ]]
The mean of the samples was 0.694
Iteration 309
```

```
Acquisition function convergence reached at iteration 301.
The final UCB loss was -4.619 with predicted mean of [0.221]
The next parameters to simulate from are [[0.782 1.    0.017 0.056 0.071 0.   ]]
The mean of the samples was 0.197
Iteration 310
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.057 with predicted mean of [0.568]
The next parameters to simulate from are [[0.48  1.    0.033 0.    0.029 0.   ]]
The mean of the samples was 0.445
Iteration 311
Acquisition function convergence reached at iteration 81.
The final UCB loss was -4.55 with predicted mean of [0.647]
The next parameters to simulate from are [[0.38  1.    0.016 0.062 0.039 0.   ]]
The mean of the samples was 0.688
Iteration 312
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.132 with predicted mean of [0.179]
The next parameters to simulate from are [[1.    0.    0.022 0.067 0.032 0.046]]
The mean of the samples was 0.347
Iteration 313
Acquisition function convergence reached at iteration 92.
The final UCB loss was -4.832 with predicted mean of [0.46]
The next parameters to simulate from are [[1.    1.    0.    0.082 0.032 0.028]]
The mean of the samples was 0.432
Iteration 314
Acquisition function convergence reached at iteration 111.
The final UCB loss was -5.024 with predicted mean of [-0.677]
The next parameters to simulate from are [[0.324 1.    0.02  0.033 0.039 0.032]]
The mean of the samples was -0.806
Iteration 315
Acquisition function convergence reached at iteration 94.
The final UCB loss was -4.635 with predicted mean of [-0.46]
The next parameters to simulate from are [[0.6   0.788 0.021 0.044 0.036 0.027]]
The mean of the samples was -0.537
Iteration 316
Acquisition function convergence reached at iteration 106.
The final UCB loss was -4.997 with predicted mean of [0.61]
The next parameters to simulate from are [[1.    0.49  0.    0.1   0.071 0.041]]
The mean of the samples was 0.448
Iteration 317
Acquisition function convergence reached at iteration 305.
The final UCB loss was -5.073 with predicted mean of [0.451]
The next parameters to simulate from are [[1.    0.495 0.    0.1   0.    0.037]]
```

```
The mean of the samples was 0.407
Iteration 318
Acquisition function convergence reached at iteration 85.
The final UCB loss was -5.032 with predicted mean of [0.561]
The next parameters to simulate from are [[1.    0.    0.018 0.1   0.037 0.027]]
The mean of the samples was 0.387
Iteration 319
Acquisition function convergence reached at iteration 111.
The final UCB loss was -5.121 with predicted mean of [0.619]
The next parameters to simulate from are [[0.621 0.    0.021 0.1   0.    0.04 ]]
The mean of the samples was 0.47
Hyperparameter convergence reached at iteration 1933.
The minimum predicted mean of the observed indices is -1.382 at the point [0.    1.    0.033
Iteration 320
Acquisition function convergence reached at iteration 89.
The final UCB loss was -4.887 with predicted mean of [0.406]
The next parameters to simulate from are [[1.    0.    0.018 0.082 0.    0.036]]
The mean of the samples was 0.418
Iteration 321
Acquisition function convergence reached at iteration 341.
The final UCB loss was -4.946 with predicted mean of [0.886]
The next parameters to simulate from are [[0.582 0.    0.    0.1   0.071 0.038]]
The mean of the samples was 1.177
Iteration 322
Acquisition function convergence reached at iteration 372.
The final UCB loss was -4.86 with predicted mean of [0.56]
The next parameters to simulate from are [[1.    0.458 0.033 0.1   0.036 0.039]]
The mean of the samples was 0.464
Iteration 323
Acquisition function convergence reached at iteration 94.
The final UCB loss was -4.781 with predicted mean of [0.75]
The next parameters to simulate from are [[1.    1.    0.021 0.1   0.071 0.026]]
The mean of the samples was 0.366
Iteration 324
Acquisition function convergence reached at iteration 78.
The final UCB loss was -4.606 with predicted mean of [0.358]
The next parameters to simulate from are [[1.    0.538 0.    0.1   0.036 0.071]]
The mean of the samples was 0.351
Iteration 325
Acquisition function convergence reached at iteration 108.
The final UCB loss was -4.598 with predicted mean of [0.449]
The next parameters to simulate from are [[1.    0.565 0.016 0.079 0.034 0.046]]
The mean of the samples was 0.38
```

```
Iteration 326
Acquisition function convergence reached at iteration 481.
The final UCB loss was -5.208 with predicted mean of [0.196]
The next parameters to simulate from are [[0.4   0.    0.    0.    0.071 0.071]]
The mean of the samples was 0.462
Iteration 327
Acquisition function convergence reached at iteration 98.
The final UCB loss was -5.126 with predicted mean of [0.861]
The next parameters to simulate from are [[0.463 0.    0.    0.1   0.    0.   ]]
The mean of the samples was 1.673
Iteration 328
Acquisition function convergence reached at iteration 99.
The final UCB loss was -4.44 with predicted mean of [1.229]
The next parameters to simulate from are [[0.    0.705 0.    0.075 0.071 0.071]]
The mean of the samples was 1.467
Iteration 329
Acquisition function convergence reached at iteration 62.
The final UCB loss was -4.891 with predicted mean of [0.712]
The next parameters to simulate from are [[0.576 1.    0.    0.1   0.071 0.071]]
The mean of the samples was 0.867
Iteration 330
Acquisition function convergence reached at iteration 130.
The final UCB loss was -4.814 with predicted mean of [0.275]
The next parameters to simulate from are [[0.642 0.    0.    0.046 0.    0.04 ]]
The mean of the samples was 0.426
Iteration 331
Acquisition function convergence reached at iteration 113.
The final UCB loss was -4.792 with predicted mean of [0.621]
The next parameters to simulate from are [[0.561 0.    0.    0.073 0.    0.019]]
The mean of the samples was -0.174
Iteration 332
Acquisition function convergence reached at iteration 535.
The final UCB loss was -4.83 with predicted mean of [0.768]
The next parameters to simulate from are [[0.545 0.    0.033 0.1   0.071 0.071]]
The mean of the samples was 0.768
Iteration 333
Acquisition function convergence reached at iteration 321.
The final UCB loss was -4.743 with predicted mean of [0.285]
The next parameters to simulate from are [[1.    0.328 0.    0.066 0.071 0.036]]
The mean of the samples was 0.428
Iteration 334
Acquisition function convergence reached at iteration 112.
The final UCB loss was -4.485 with predicted mean of [0.346]
```

```
The next parameters to simulate from are [[0.693 0.    0.    0.088 0.    0.042]]
The mean of the samples was 0.381
Iteration 335
Acquisition function convergence reached at iteration 99.
The final UCB loss was -4.821 with predicted mean of [0.627]
The next parameters to simulate from are [[0.583 0.    0.018 0.078 0.071 0.071]]
The mean of the samples was 0.651
Iteration 336
Acquisition function convergence reached at iteration 105.
The final UCB loss was -4.797 with predicted mean of [0.284]
The next parameters to simulate from are [[0.788 0.999 0.    0.058 0.    0.071]]
The mean of the samples was 0.421
Iteration 337
Acquisition function convergence reached at iteration 92.
The final UCB loss was -4.867 with predicted mean of [0.519]
The next parameters to simulate from are [[0.62  0.    0.    0.077 0.038 0.   ]]
The mean of the samples was 0.983
Iteration 338
Acquisition function convergence reached at iteration 111.
The final UCB loss was -4.726 with predicted mean of [0.822]
The next parameters to simulate from are [[0.    0.    0.013 0.055 0.071 0.026]]
The mean of the samples was 0.848
Iteration 339
Acquisition function convergence reached at iteration 80.
The final UCB loss was -5.225 with predicted mean of [0.54]
The next parameters to simulate from are [[0.562 0.54  0.033 0.1   0.    0.071]]
The mean of the samples was -0.059
Hyperparameter convergence reached at iteration 1979.
The minimum predicted mean of the observed indices is -1.386 at the point [0.    1.    0.033
Iteration 340
Acquisition function convergence reached at iteration 109.
The final UCB loss was -4.78 with predicted mean of [0.81]
The next parameters to simulate from are [[0.557 0.999 0.033 0.082 0.071 0.071]]
The mean of the samples was 0.576
Iteration 341
Acquisition function convergence reached at iteration 100.
The final UCB loss was -4.815 with predicted mean of [0.497]
The next parameters to simulate from are [[1.    0.    0.018 0.084 0.071 0.034]]
The mean of the samples was 0.414
Iteration 342
Acquisition function convergence reached at iteration 84.
The final UCB loss was -4.622 with predicted mean of [0.85]
The next parameters to simulate from are [[0.693 0.53  0.017 0.1   0.071 0.071]]
```

```
The mean of the samples was 0.686
Iteration 343
Acquisition function convergence reached at iteration 124.
The final UCB loss was -4.818 with predicted mean of [0.329]
The next parameters to simulate from are [[0.273 0.    0.    0.055 0.031 0.036]]
The mean of the samples was 0.337
Iteration 344
Acquisition function convergence reached at iteration 99.
The final UCB loss was -4.908 with predicted mean of [0.117]
The next parameters to simulate from are [[0.463 0.303 0.    0.026 0.071 0.071]]
The mean of the samples was -0.082
Iteration 345
Acquisition function convergence reached at iteration 78.
The final UCB loss was -4.798 with predicted mean of [0.46]
The next parameters to simulate from are [[0.579 1.    0.02  0.1   0.    0.071]]
The mean of the samples was 0.098
Iteration 346
Acquisition function convergence reached at iteration 99.
The final UCB loss was -4.786 with predicted mean of [0.435]
The next parameters to simulate from are [[0.511 0.    0.    0.05  0.071 0.071]]
The mean of the samples was 0.626
Iteration 347
Acquisition function convergence reached at iteration 84.
The final UCB loss was -4.549 with predicted mean of [1.085]
The next parameters to simulate from are [[0.703 0.395 0.017 0.1   0.071 0.   ]]
The mean of the samples was 1.453
Iteration 348
Acquisition function convergence reached at iteration 118.
The final UCB loss was -4.622 with predicted mean of [0.358]
The next parameters to simulate from are [[1.    0.    0.    0.069 0.033 0.071]]
The mean of the samples was 0.356
Iteration 349
Acquisition function convergence reached at iteration 264.
The final UCB loss was -4.909 with predicted mean of [0.401]
The next parameters to simulate from are [[0.538 0.523 0.    0.065 0.    0.   ]]
The mean of the samples was 0.621
Iteration 350
Acquisition function convergence reached at iteration 77.
The final UCB loss was -4.5 with predicted mean of [0.794]
The next parameters to simulate from are [[0.549 1.    0.    0.083 0.071 0.035]]
The mean of the samples was 0.929
Trained parameters:
amplitude_champ:0 is 0.721
```

```
length_scales_champ:0 is [0.801 1.    1.    0.334 1.    1.   ]

observation_noise_variance_champ:0 is 0.405

bias_mean:0 is 0.864

Iteration 351
Acquisition function convergence reached at iteration 213.
The final UCB loss was -4.814 with predicted mean of [0.268]
The next parameters to simulate from are [[0.514 1.    0.    0.045 0.    0.   ]]
The mean of the samples was 0.34
Iteration 352
Acquisition function convergence reached at iteration 103.
The final UCB loss was -4.34 with predicted mean of [0.334]
The next parameters to simulate from are [[1.    0.516 0.    0.083 0.071 0.071]]
The mean of the samples was 0.375
Iteration 353
Acquisition function convergence reached at iteration 100.
The final UCB loss was -4.97 with predicted mean of [0.269]
The next parameters to simulate from are [[0.378 0.622 0.    0.069 0.    0.071]]
The mean of the samples was 0.353
Iteration 354
Acquisition function convergence reached at iteration 106.
The final UCB loss was -4.908 with predicted mean of [0.232]
The next parameters to simulate from are [[0.571 0.    0.033 0.064 0.071 0.04 ]]
The mean of the samples was 0.307
Iteration 355
Acquisition function convergence reached at iteration 102.
The final UCB loss was -4.775 with predicted mean of [0.362]
The next parameters to simulate from are [[1.    0.523 0.    0.066 0.    0.035]]
The mean of the samples was 0.408
Iteration 356
Acquisition function convergence reached at iteration 337.
The final UCB loss was -4.667 with predicted mean of [0.724]
The next parameters to simulate from are [[1.    0.477 0.    0.1   0.033 0.   ]]
The mean of the samples was 0.624
Iteration 357
Acquisition function convergence reached at iteration 436.
The final UCB loss was -4.518 with predicted mean of [0.661]
The next parameters to simulate from are [[0.638 0.    0.    0.1   0.038 0.071]]
The mean of the samples was 1.042
Iteration 358
```

```
Acquisition function convergence reached at iteration 99.
The final UCB loss was -4.417 with predicted mean of [0.838]
The next parameters to simulate from are [[0.508 0.627 0.    0.1   0.    0.033]]
The mean of the samples was 0.731
Iteration 359
Acquisition function convergence reached at iteration 93.
The final UCB loss was -4.284 with predicted mean of [0.416]
The next parameters to simulate from are [[1.    0.    0.017 0.09  0.071 0.071]]
The mean of the samples was 0.358
Hyperparameter convergence reached at iteration 1956.
The minimum predicted mean of the observed indices is -1.374 at the point [0.    1.    0.033
Iteration 360
Acquisition function convergence reached at iteration 108.
The final UCB loss was -4.444 with predicted mean of [0.361]
The next parameters to simulate from are [[1.    0.    0.033 0.086 0.035 0.071]]
The mean of the samples was 0.35
Iteration 361
Acquisition function convergence reached at iteration 488.
The final UCB loss was -4.636 with predicted mean of [0.334]
The next parameters to simulate from are [[1.    0.53  0.018 0.1   0.    0.071]]
The mean of the samples was 0.382
Iteration 362
Acquisition function convergence reached at iteration 105.
The final UCB loss was -4.821 with predicted mean of [0.141]
The next parameters to simulate from are [[0.55  0.    0.021 0.08  0.    0.071]]
The mean of the samples was -0.066
Iteration 363
Acquisition function convergence reached at iteration 411.
The final UCB loss was -4.561 with predicted mean of [0.748]
The next parameters to simulate from are [[0.547 1.    0.033 0.1   0.036 0.071]]
The mean of the samples was 0.77
Iteration 364
Acquisition function convergence reached at iteration 118.
The final UCB loss was -4.518 with predicted mean of [0.373]
The next parameters to simulate from are [[0.235 0.438 0.012 0.043 0.071 0.   ]]
The mean of the samples was 0.559
Iteration 365
Acquisition function convergence reached at iteration 437.
The final UCB loss was -4.629 with predicted mean of [0.734]
The next parameters to simulate from are [[0.    0.    0.01  0.1   0.    0.071]]
The mean of the samples was 1.019
Iteration 366
Acquisition function convergence reached at iteration 305.
```

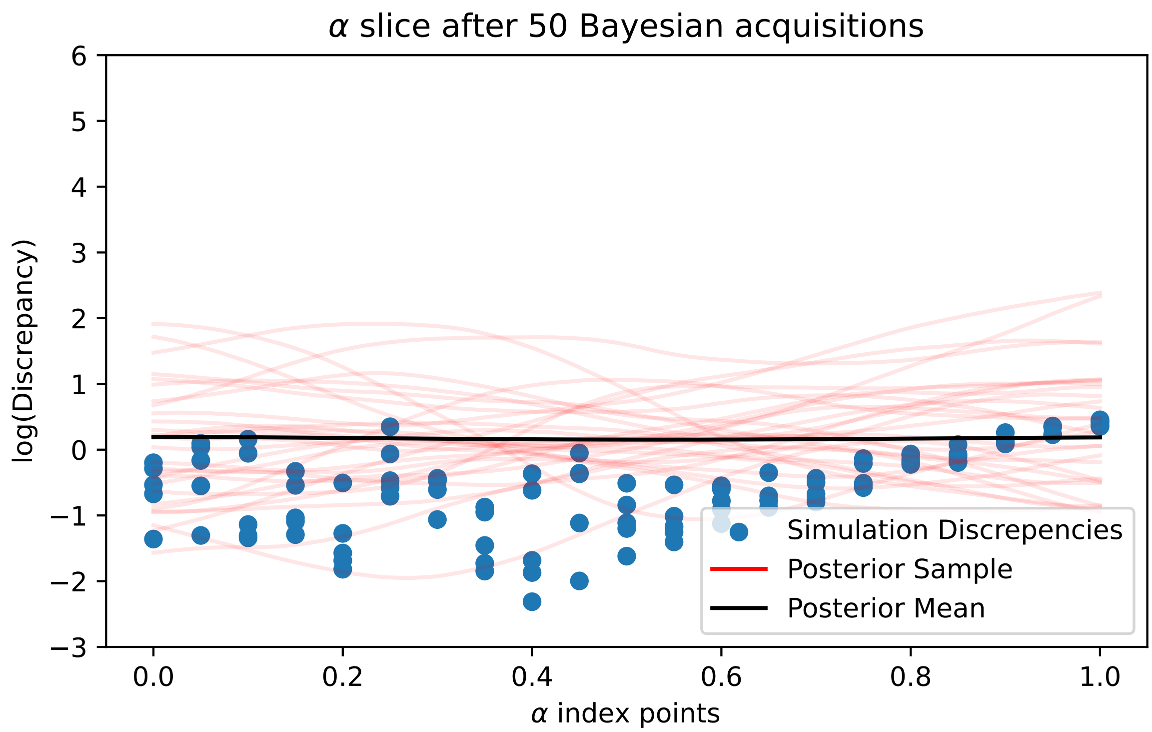
```
The final UCB loss was -4.122 with predicted mean of [1.195]
The next parameters to simulate from are [[0.    0.    0.018 0.085 0.036 0.071]]
The mean of the samples was 1.548
Iteration 367
Acquisition function convergence reached at iteration 88.
The final UCB loss was -4.26 with predicted mean of [0.342]
The next parameters to simulate from are [[1.    0.    0.013 0.083 0.022 0.071]]
The mean of the samples was 0.367
Iteration 368
Acquisition function convergence reached at iteration 416.
The final UCB loss was -4.379 with predicted mean of [1.033]
The next parameters to simulate from are [[0.    1.    0.011 0.1   0.017 0.071]]
The mean of the samples was 1.362
Iteration 369
Acquisition function convergence reached at iteration 69.
The final UCB loss was -4.607 with predicted mean of [1.37]
The next parameters to simulate from are [[0.    0.298 0.033 0.1   0.071 0.   ]]
The mean of the samples was 2.542
Iteration 370
Acquisition function convergence reached at iteration 61.
The final UCB loss was -4.589 with predicted mean of [0.326]
The next parameters to simulate from are [[0.727 1.    0.    0.082 0.    0.071]]
The mean of the samples was 0.35
Iteration 371
Acquisition function convergence reached at iteration 92.
The final UCB loss was -4.628 with predicted mean of [0.43]
The next parameters to simulate from are [[0.631 0.    0.017 0.06  0.071 0.   ]]
The mean of the samples was 0.929
Iteration 372
Acquisition function convergence reached at iteration 105.
The final UCB loss was -5.105 with predicted mean of [-0.493]
The next parameters to simulate from are [[0.42  0.    0.033 0.017 0.071 0.071]]
The mean of the samples was -0.612
Iteration 373
Acquisition function convergence reached at iteration 108.
The final UCB loss was -4.722 with predicted mean of [0.436]
The next parameters to simulate from are [[0.    1.    0.033 0.048 0.071 0.026]]
The mean of the samples was 0.255
Iteration 374
Acquisition function convergence reached at iteration 99.
The final UCB loss was -4.456 with predicted mean of [0.975]
The next parameters to simulate from are [[0.    0.468 0.033 0.064 0.071 0.028]]
The mean of the samples was 1.218
```

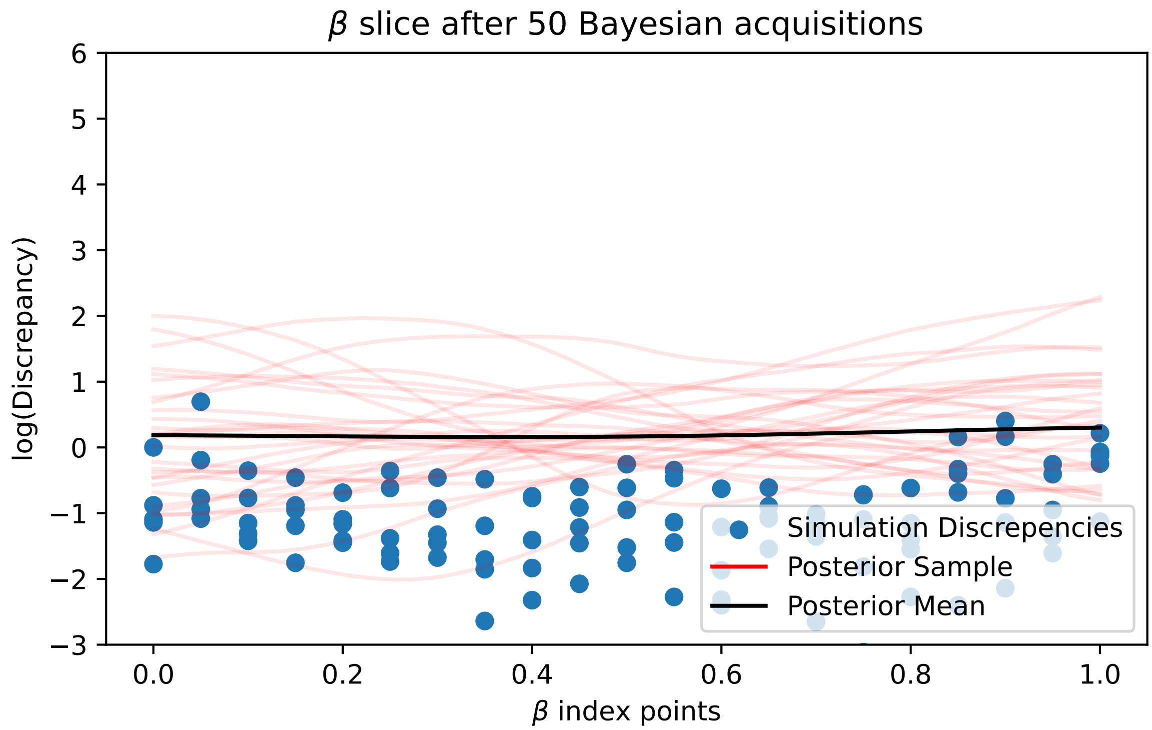
```
Iteration 375
Acquisition function convergence reached at iteration 435.
The final UCB loss was -4.297 with predicted mean of [0.353]
The next parameters to simulate from are [[0.659 0.48  0.    0.1   0.    0.071]]
The mean of the samples was 0.445
Iteration 376
Acquisition function convergence reached at iteration 93.
The final UCB loss was -4.486 with predicted mean of [0.754]
The next parameters to simulate from are [[0.    0.516 0.    0.1   0.    0.071]]
The mean of the samples was 0.562
Iteration 377
Acquisition function convergence reached at iteration 396.
The final UCB loss was -4.98 with predicted mean of [-0.315]
The next parameters to simulate from are [[0.471 0.    0.033 0.062 0.    0.071]]
The mean of the samples was -0.23
Iteration 378
Acquisition function convergence reached at iteration 110.
The final UCB loss was -4.953 with predicted mean of [0.257]
The next parameters to simulate from are [[0.445 0.    0.033 0.064 0.    0.   ]]
The mean of the samples was 0.657
Iteration 379
Acquisition function convergence reached at iteration 84.
The final UCB loss was -3.946 with predicted mean of [0.394]
The next parameters to simulate from are [[0.337 0.459 0.    0.086 0.    0.071]]
The mean of the samples was 0.38
Hyperparameter convergence reached at iteration 1986.
The minimum predicted mean of the observed indices is -1.359 at the point [0.    1.    0.033
Iteration 380
Acquisition function convergence reached at iteration 123.
The final UCB loss was -4.423 with predicted mean of [0.042]
The next parameters to simulate from are [[0.586 0.543 0.015 0.067 0.023 0.035]]
The mean of the samples was -0.24
Iteration 381
Acquisition function convergence reached at iteration 98.
The final UCB loss was -4.147 with predicted mean of [0.759]
The next parameters to simulate from are [[0.584 0.    0.    0.075 0.071 0.038]]
The mean of the samples was 0.44
Iteration 382
Acquisition function convergence reached at iteration 251.
The final UCB loss was -4.292 with predicted mean of [0.352]
The next parameters to simulate from are [[0.    0.    0.    0.064 0.    0.041]]
The mean of the samples was 0.197
Iteration 383
```

```
Acquisition function convergence reached at iteration 93.
The final UCB loss was -4.754 with predicted mean of [-0.067]
The next parameters to simulate from are [[0.715 0.    0.019 0.019 0.041 0.071]]
The mean of the samples was 0.171
Iteration 384
Acquisition function convergence reached at iteration 108.
The final UCB loss was -4.888 with predicted mean of [-0.785]
The next parameters to simulate from are [[0.365 0.    0.014 0.029 0.041 0.036]]
The mean of the samples was -1.273
Iteration 385
Acquisition function convergence reached at iteration 389.
The final UCB loss was -4.989 with predicted mean of [0.223]
The next parameters to simulate from are [[1.    0.    0.023 0.049 0.071 0.071]]
The mean of the samples was 0.424
Iteration 386
Acquisition function convergence reached at iteration 110.
The final UCB loss was -4.311 with predicted mean of [0.568]
The next parameters to simulate from are [[0.    1.    0.033 0.054 0.038 0.   ]]
The mean of the samples was 1.036
Iteration 387
Acquisition function convergence reached at iteration 125.
The final UCB loss was -4.716 with predicted mean of [-0.117]
The next parameters to simulate from are [[0.528 0.535 0.019 0.048 0.045 0.071]]
The mean of the samples was -0.262
Iteration 388
Acquisition function convergence reached at iteration 376.
The final UCB loss was -4.819 with predicted mean of [-0.111]
The next parameters to simulate from are [[0.628 0.    0.    0.018 0.071 0.033]]
The mean of the samples was -0.307
Iteration 389
Acquisition function convergence reached at iteration 95.
The final UCB loss was -4.196 with predicted mean of [0.612]
The next parameters to simulate from are [[0.491 0.37  0.019 0.1   0.025 0.071]]
The mean of the samples was 0.786
Iteration 390
Acquisition function convergence reached at iteration 91.
The final UCB loss was -3.831 with predicted mean of [0.342]
The next parameters to simulate from are [[1.    0.    0.    0.091 0.038 0.071]]
The mean of the samples was 0.364
Iteration 391
Acquisition function convergence reached at iteration 124.
The final UCB loss was -4.748 with predicted mean of [0.237]
The next parameters to simulate from are [[0.    0.609 0.    0.038 0.035 0.02 ]]
```

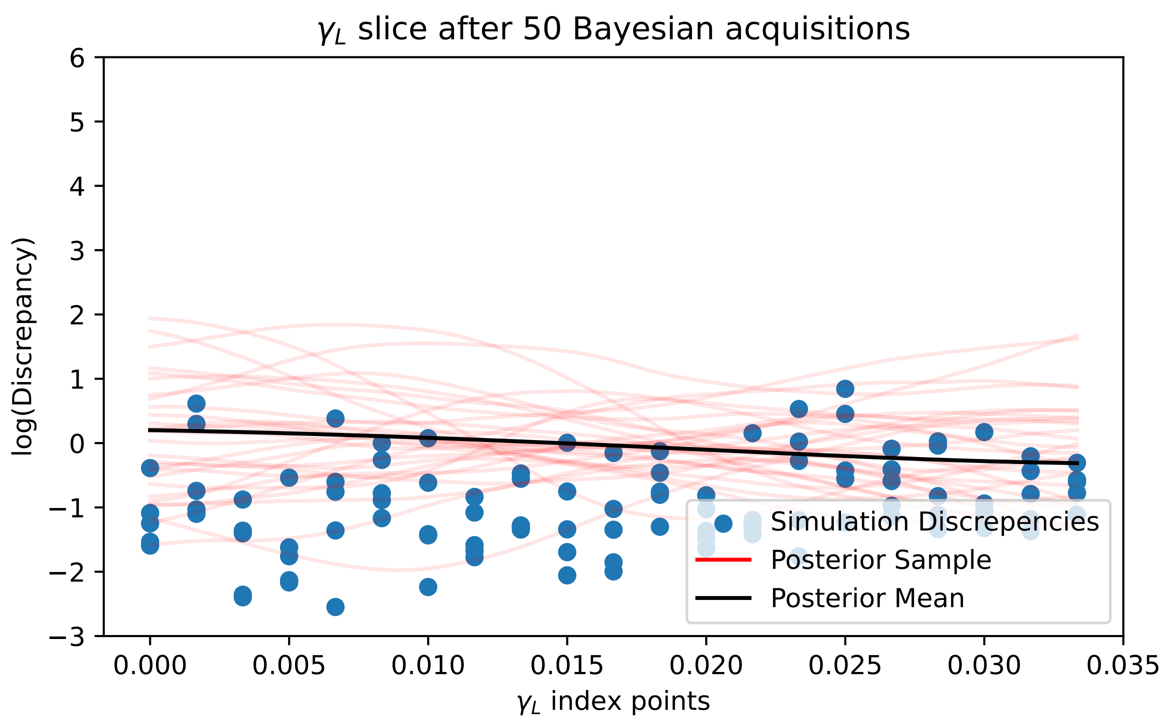
```
The mean of the samples was -0.087
Iteration 392
Acquisition function convergence reached at iteration 339.
The final UCB loss was -4.739 with predicted mean of [0.174]
The next parameters to simulate from are [[0.338 0.    0.016 0.    0.071 0.023]]
The mean of the samples was 0.5
Iteration 393
Acquisition function convergence reached at iteration 92.
The final UCB loss was -4.219 with predicted mean of [0.827]
The next parameters to simulate from are [[0.599 1.    0.    0.081 0.03  0.   ]]
The mean of the samples was 0.917
Iteration 394
Acquisition function convergence reached at iteration 102.
The final UCB loss was -4.427 with predicted mean of [0.348]
The next parameters to simulate from are [[0.683 0.65  0.    0.047 0.071 0.071]]
The mean of the samples was 0.343
Iteration 395
Acquisition function convergence reached at iteration 76.
The final UCB loss was -4.461 with predicted mean of [0.763]
The next parameters to simulate from are [[0.619 0.    0.033 0.086 0.    0.   ]]
The mean of the samples was 0.701
Iteration 396
Acquisition function convergence reached at iteration 101.
The final UCB loss was -4.072 with predicted mean of [0.705]
The next parameters to simulate from are [[0.396 0.    0.    0.076 0.038 0.071]]
The mean of the samples was 0.568
Iteration 397
Acquisition function convergence reached at iteration 83.
The final UCB loss was -4.329 with predicted mean of [0.608]
The next parameters to simulate from are [[0.    1.    0.023 0.069 0.023 0.071]]
The mean of the samples was 0.606
Iteration 398
Acquisition function convergence reached at iteration 104.
The final UCB loss was -4.331 with predicted mean of [0.358]
The next parameters to simulate from are [[0.728 0.    0.016 0.071 0.    0.   ]]
The mean of the samples was 0.682
Iteration 399
Acquisition function convergence reached at iteration 114.
The final UCB loss was -4.583 with predicted mean of [0.401]
The next parameters to simulate from are [[1.    1.    0.    0.076 0.    0.   ]]
The mean of the samples was 0.549
Hyperparameter convergence reached at iteration 1997.
The minimum predicted mean of the observed indices is -1.354 at the point [0.    1.    0.033
```

```
Iteration 400
Acquisition function convergence reached at iteration 309.
The final UCB loss was -4.172 with predicted mean of [0.824]
The next parameters to simulate from are [[0.    0.371 0.    0.057 0.071 0.   ]]
The mean of the samples was 0.731
Trained parameters:
amplitude_champ:0 is 0.694

length_scales_champ:0 is [0.841 1.    1.    0.362 1.    1.   ]

observation_noise_variance_champ:0 is 0.403

bias_mean:0 is 0.848
```

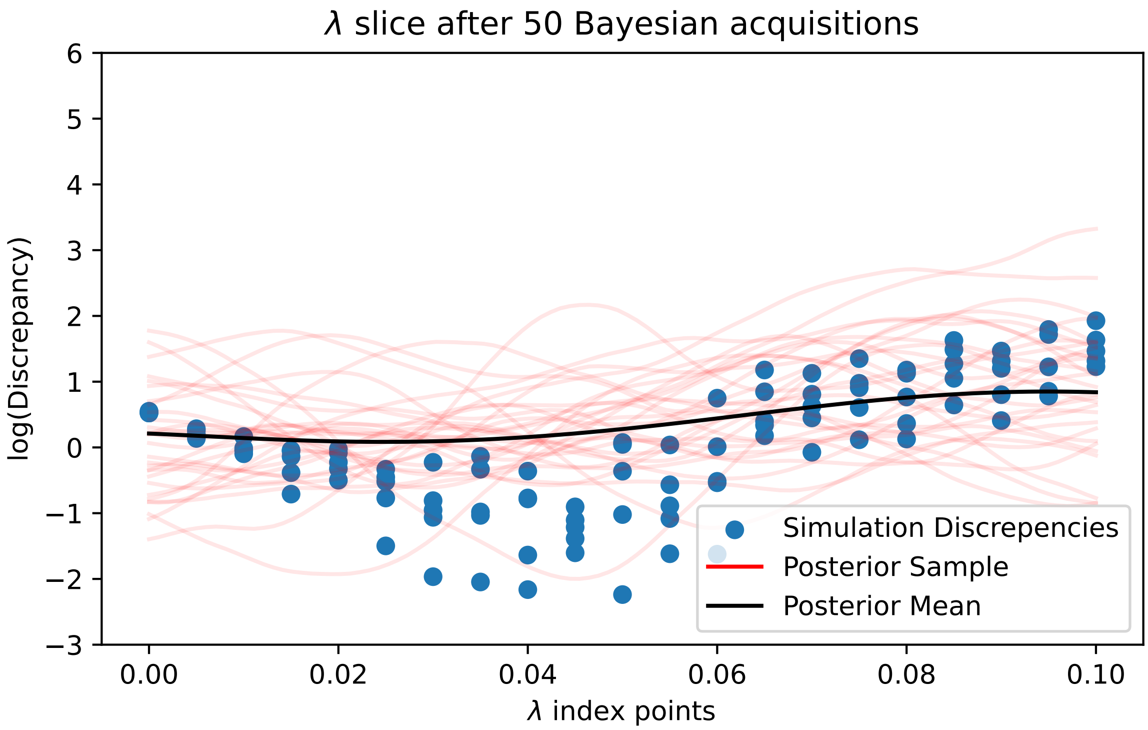




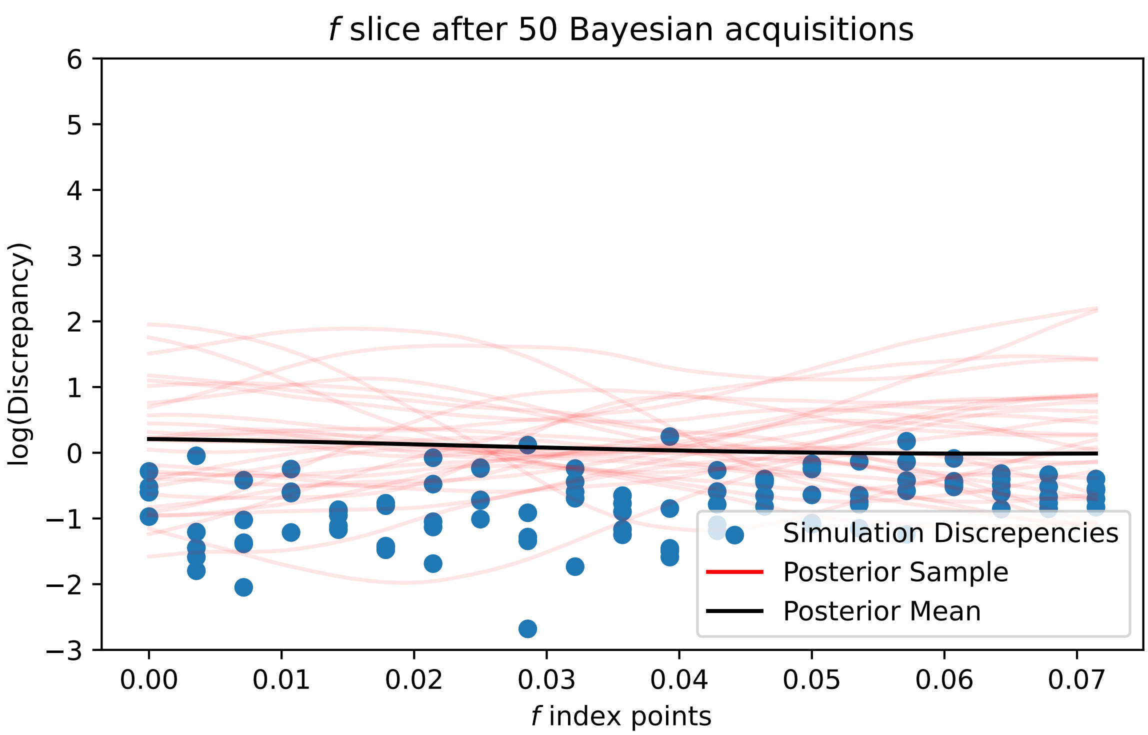
β slice after 50 Bayesian acquisitions
log(Discrepancy)
β index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



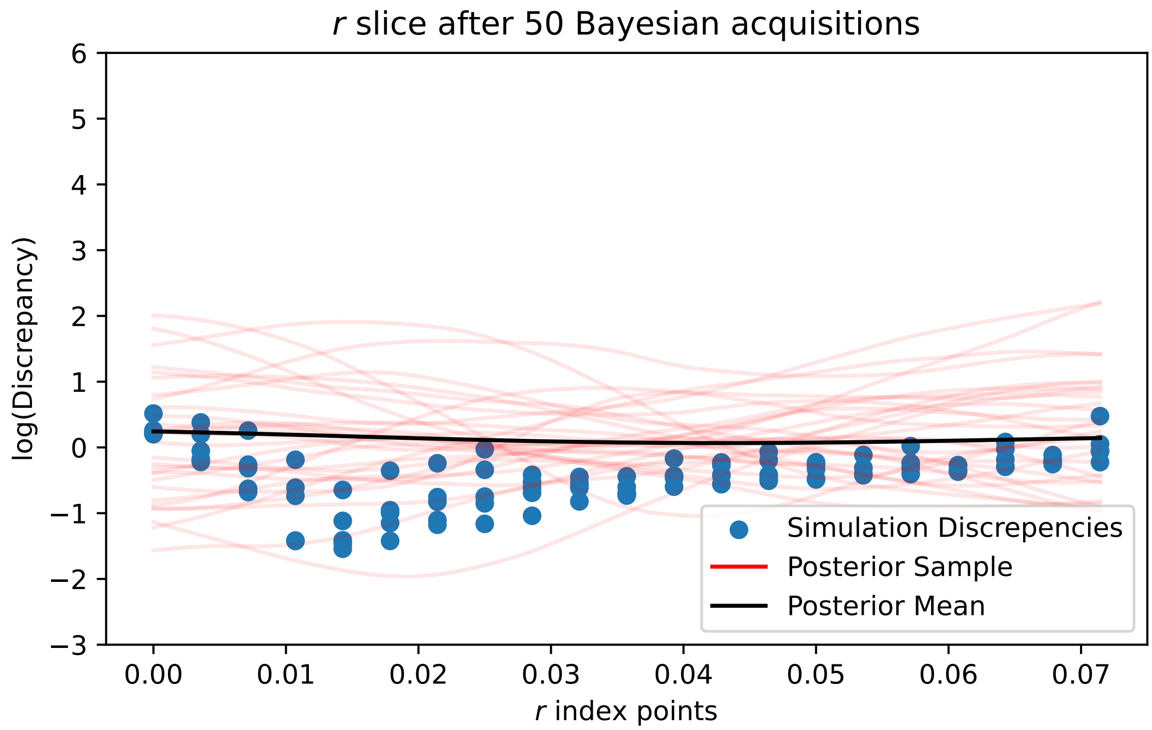
γ_L slice after 50 Bayesian acquisitions
log(Discrepancy)
γ_L index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



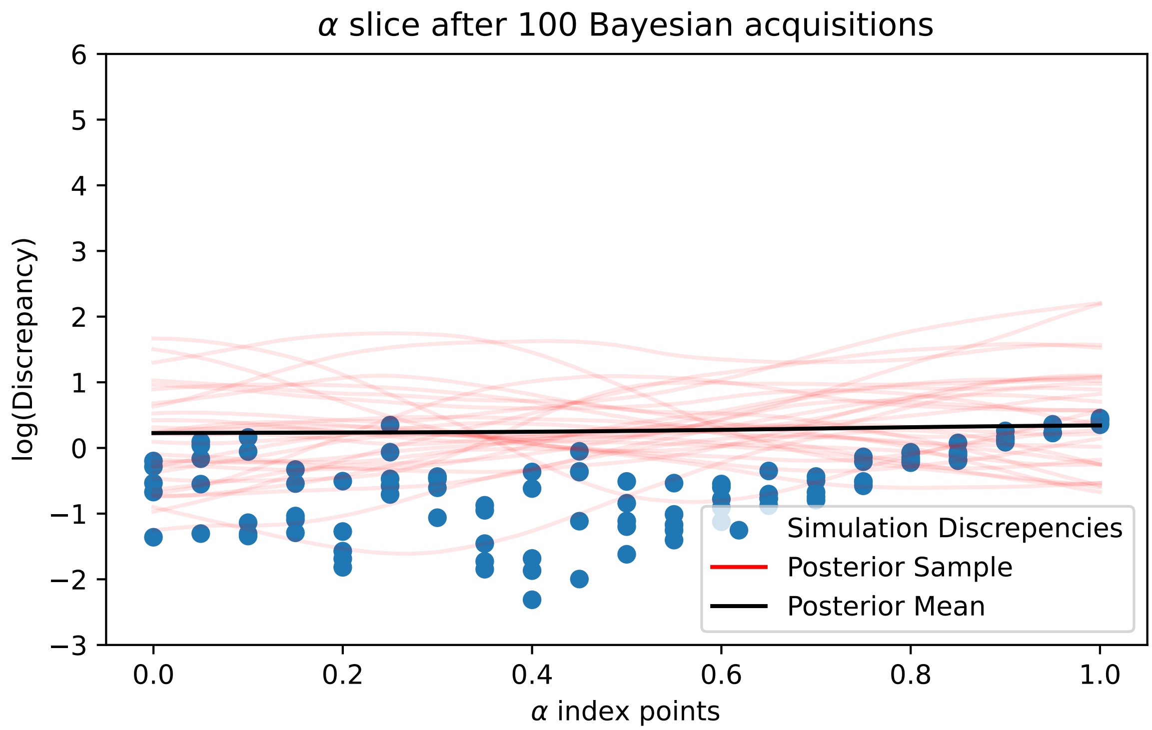
λ slice after 50 Bayesian acquisitions
log(Discrepancy)
λ index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



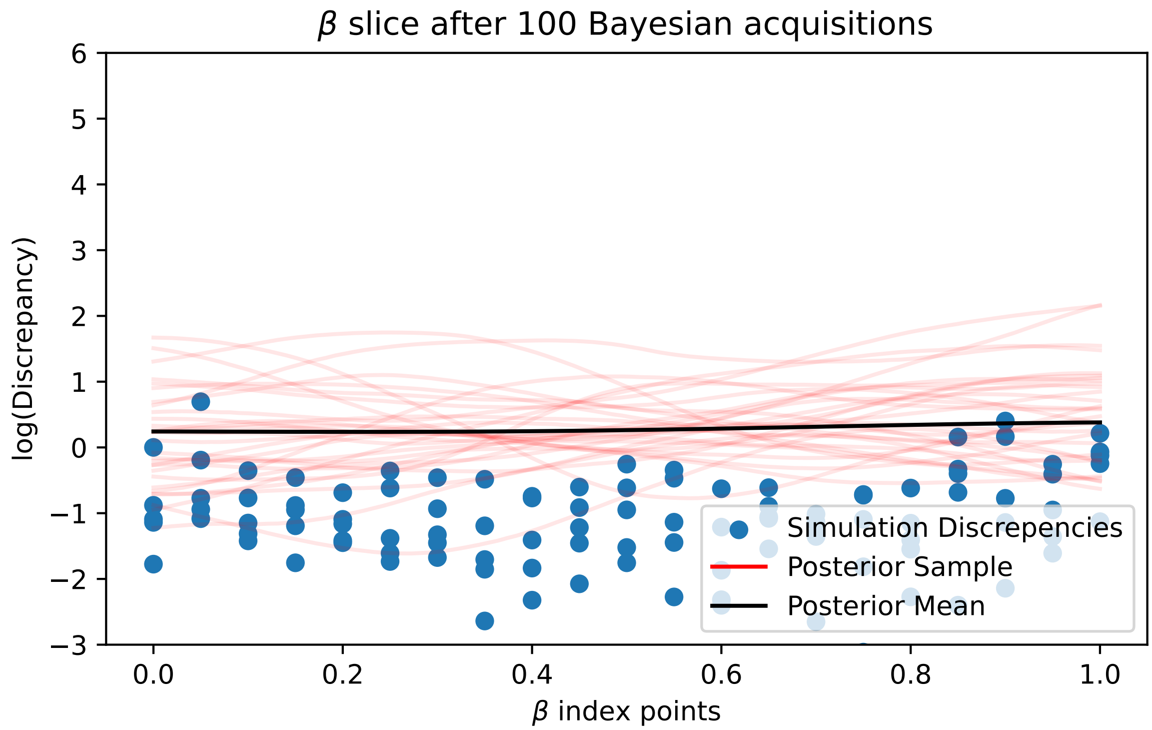
f slice after 50 Bayesian acquisitions
log(Discrepancy)
f index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

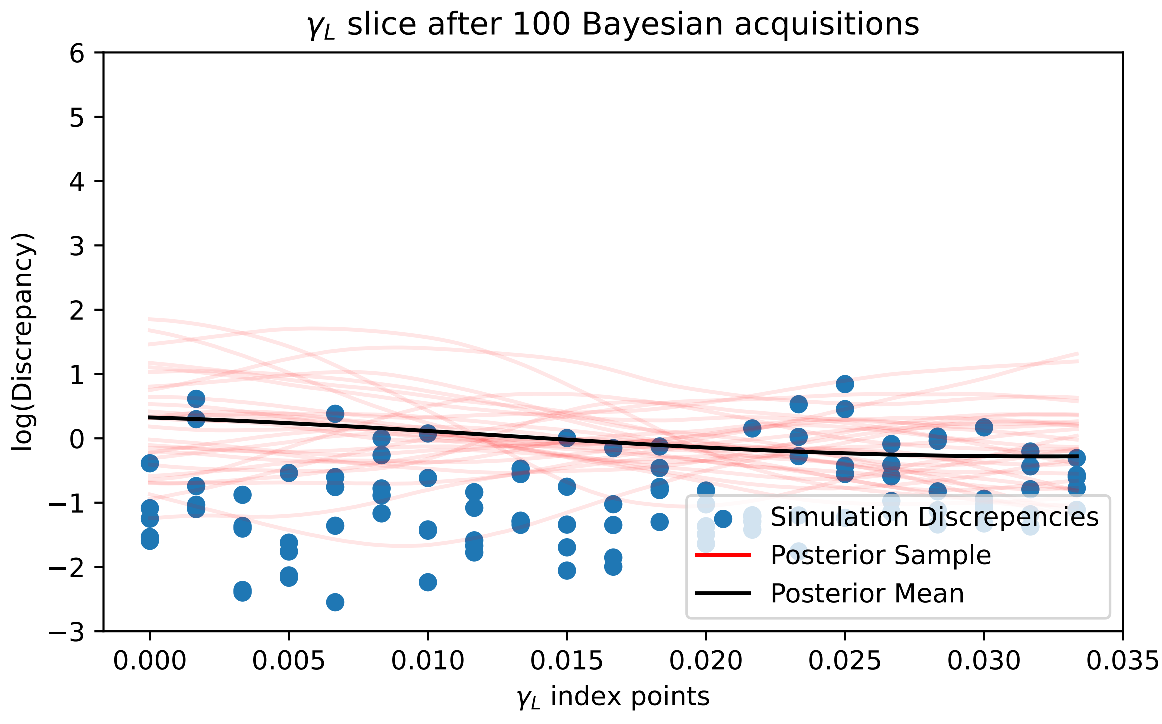


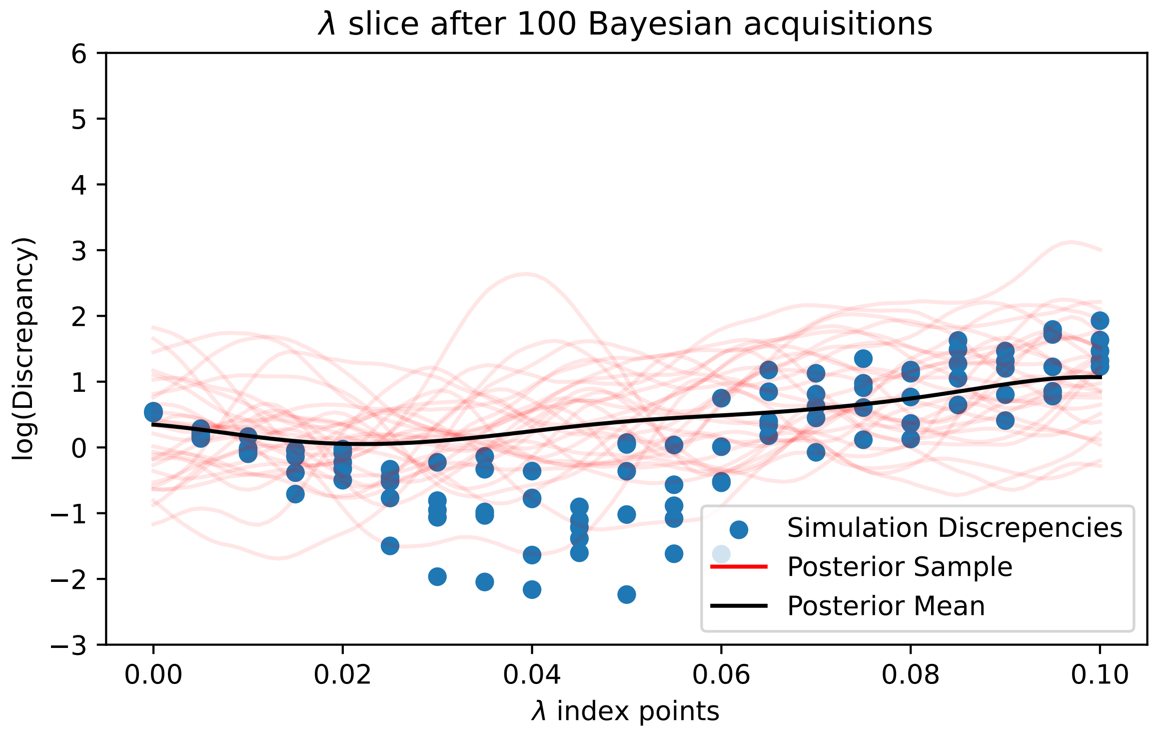
r slice after 50 Bayesian acquisitions
log(Discrepancy)
r index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



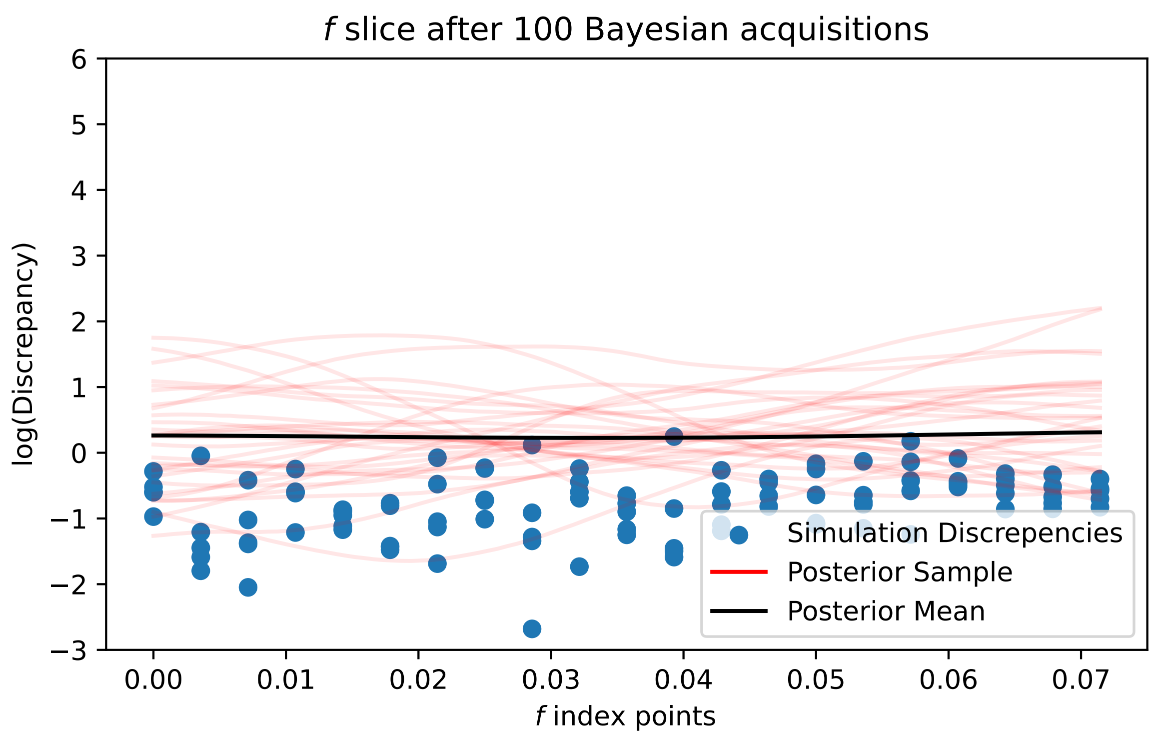
α slice after 100 Bayesian acquisitions
log(Discrepancy)
α index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


β slice after 100 Bayesian acquisitions
log(Discrepancy)
β index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

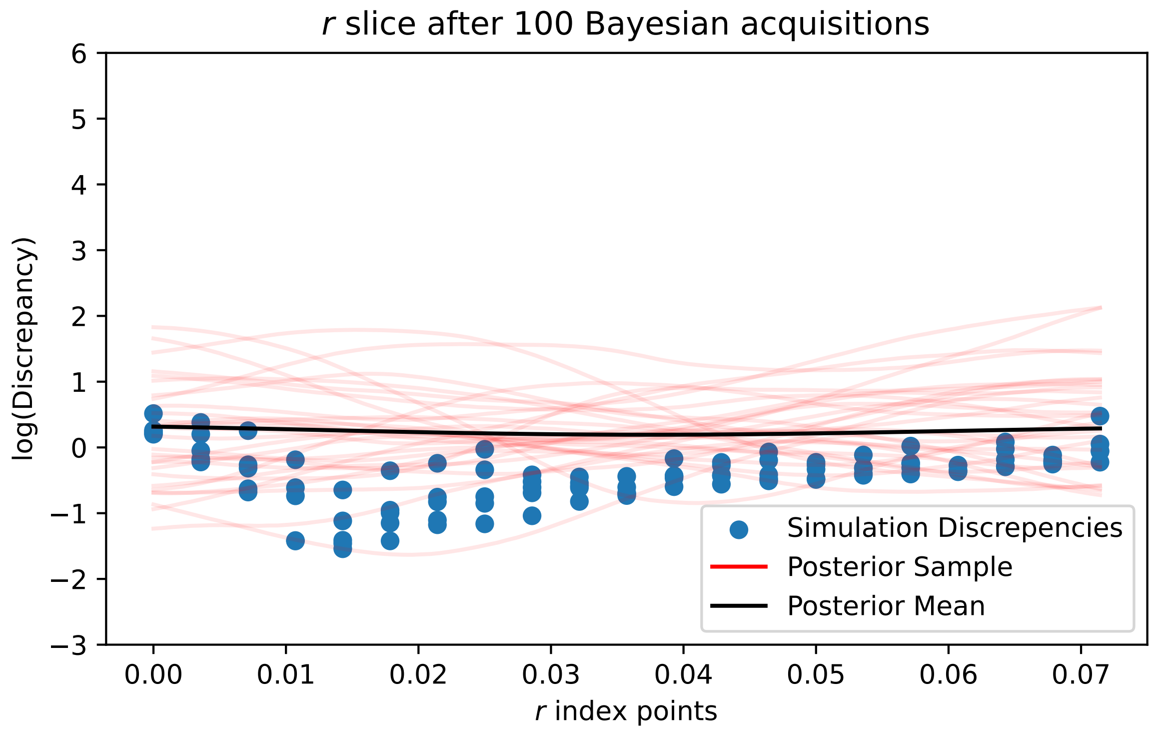

γ_L slice after 100 Bayesian acquisitions
log(Discrepancy)
γ_L index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



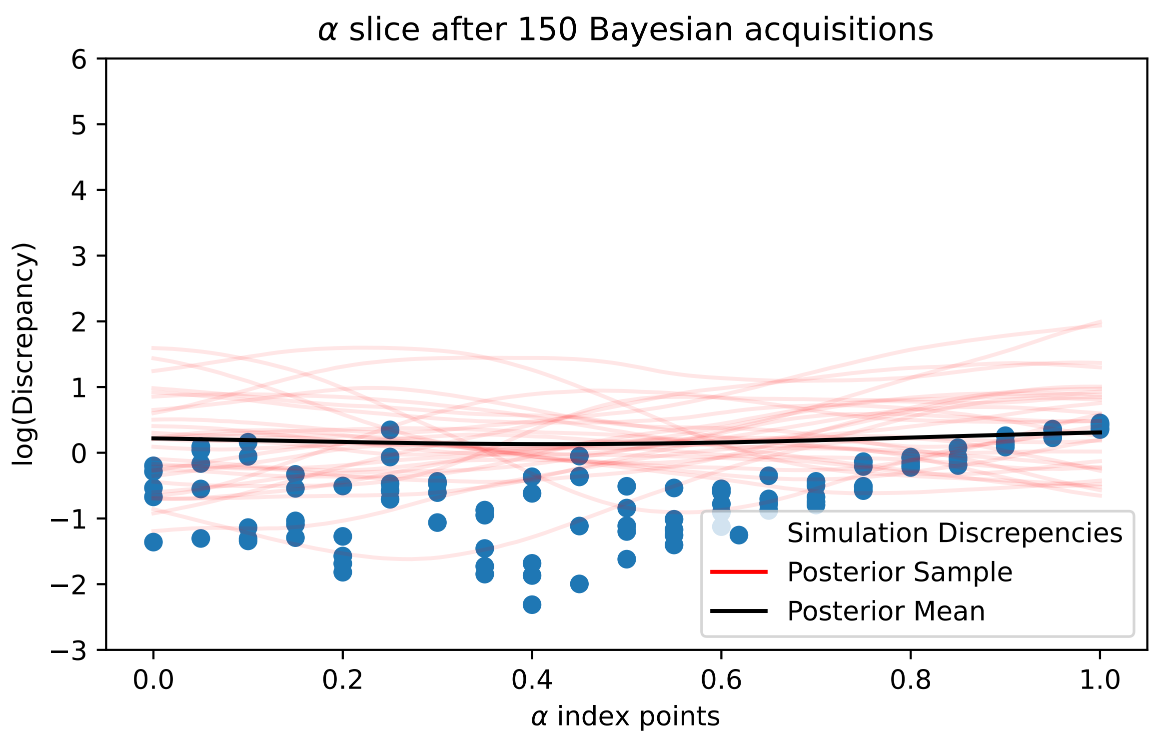
λ slice after 100 Bayesian acquisitions
log(Discrepancy)
λ index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



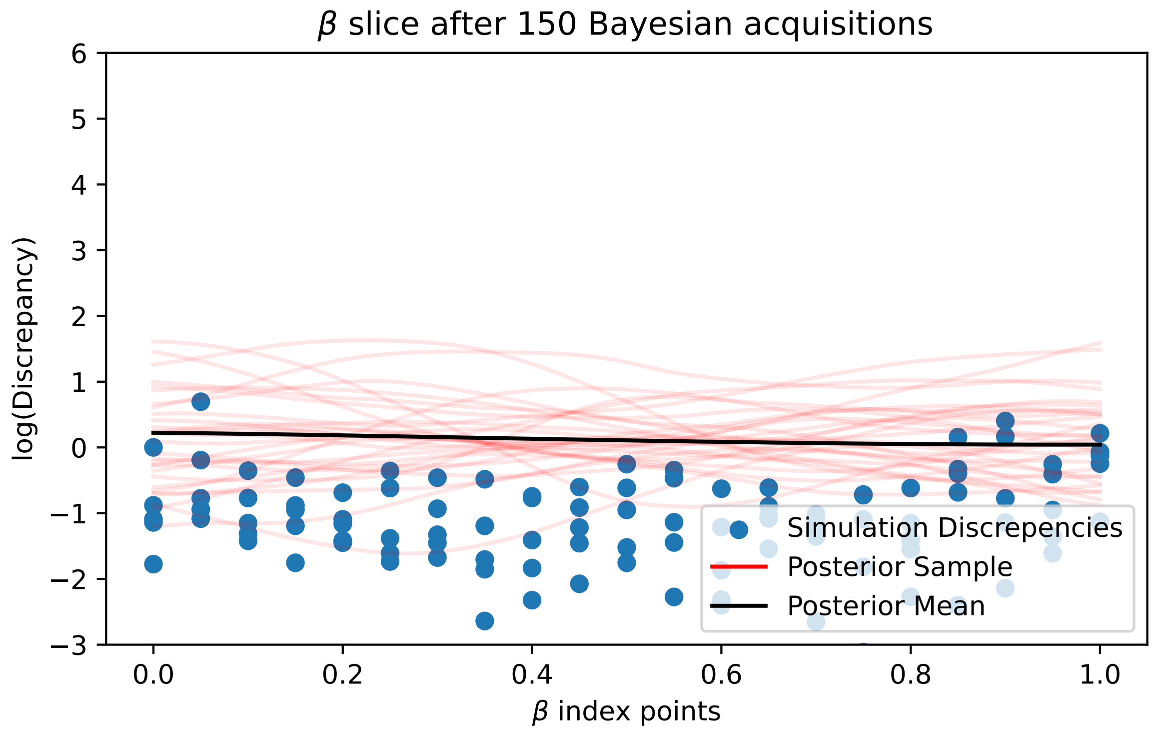
f slice after 100 Bayesian acquisitions
log(Discrepancy)
f index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

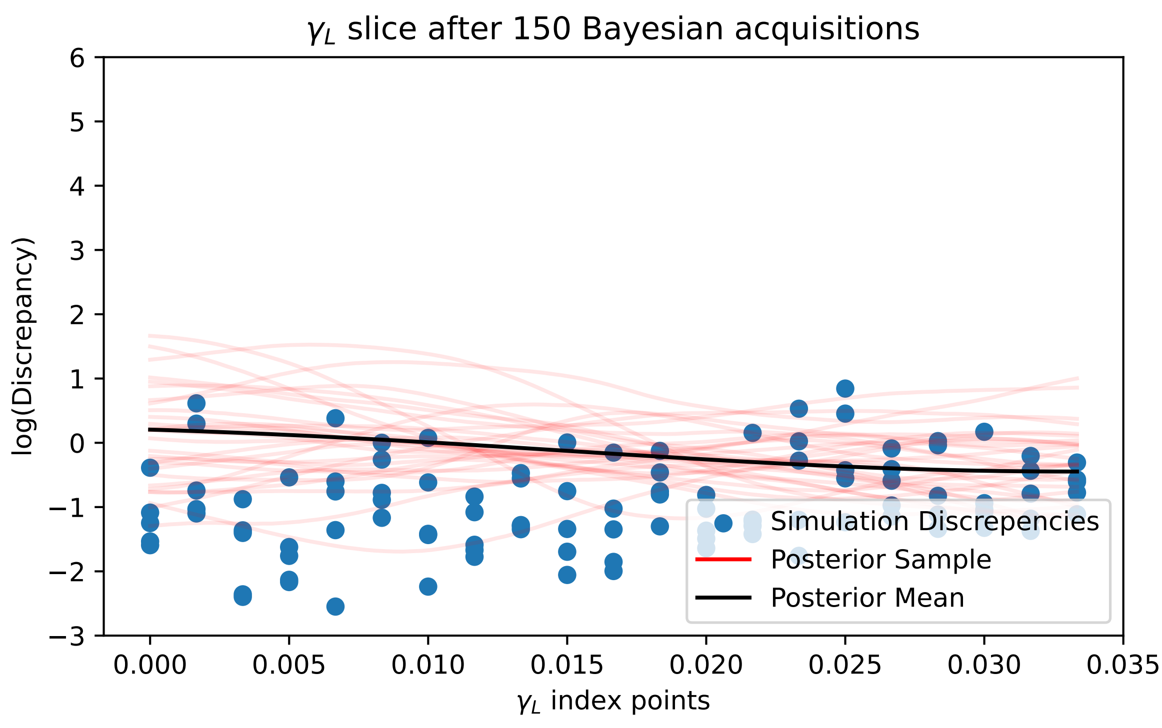


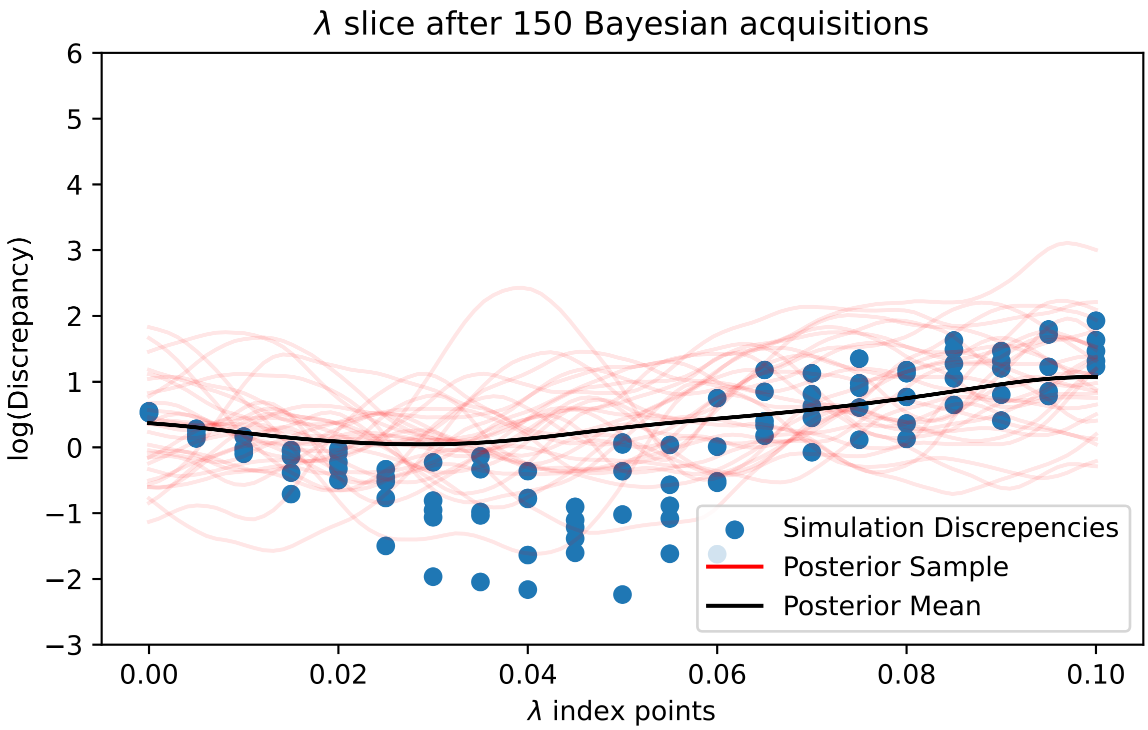
r slice after 100 Bayesian acquisitions
log(Discrepancy)
r index points
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
6
5
4
3
2
1
0
−1
−2
−3
Simulation Discrepencies
Posterior Sample
Posterior Mean

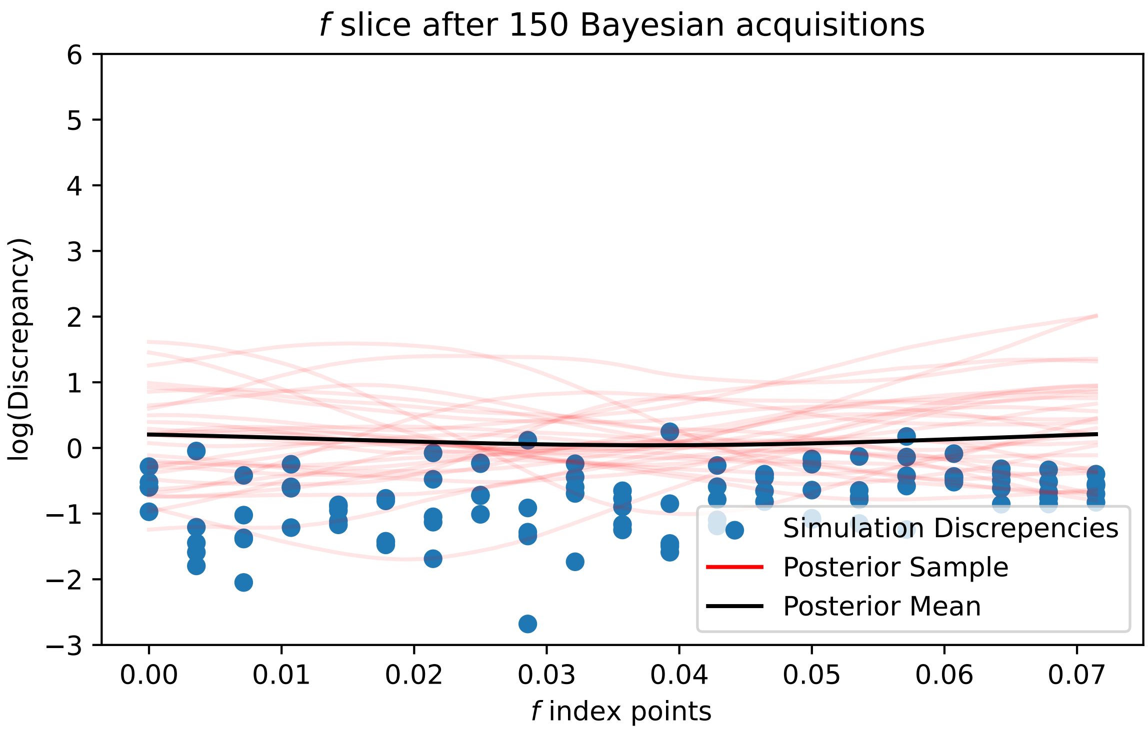


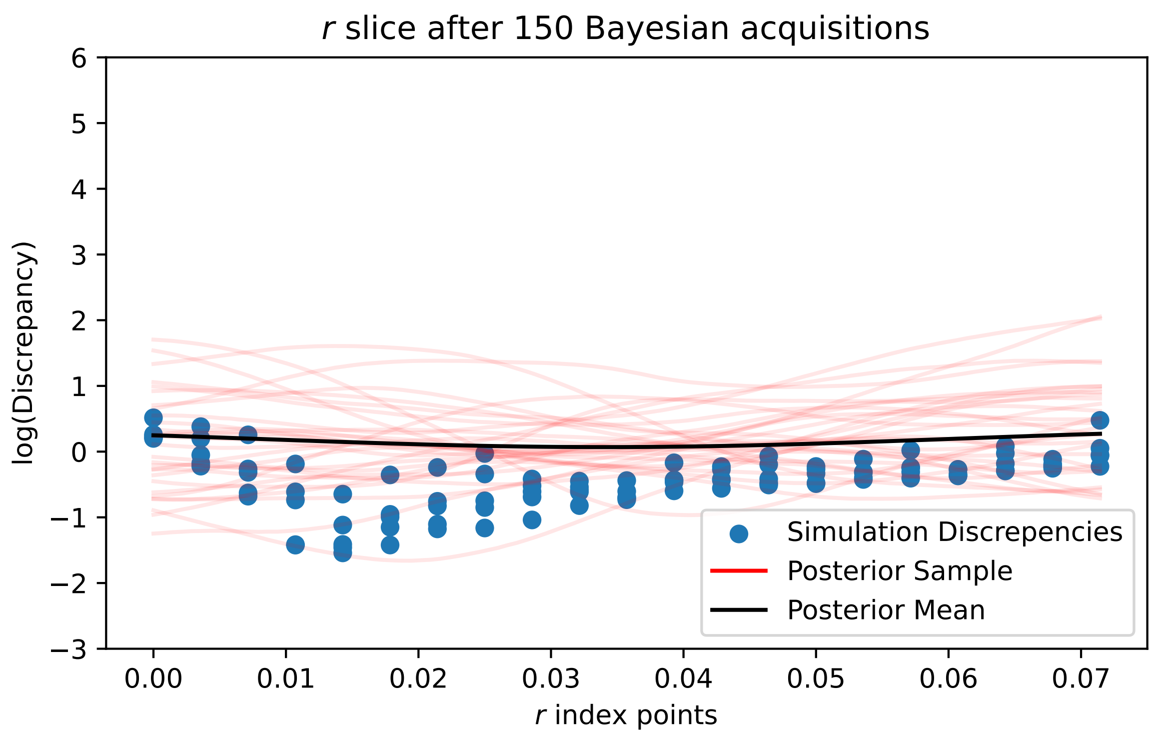
α slice after 150 Bayesian acquisitions
log(Discrepancy)
α index points
0.0
0.2
0.4
0.6
0.8
1.0
6
5
4
3
2
1
0
−1
−2
−3
Simulation Discrepencies
Posterior Sample
Posterior Mean


β slice after 150 Bayesian acquisitions
log(Discrepancy)
β index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

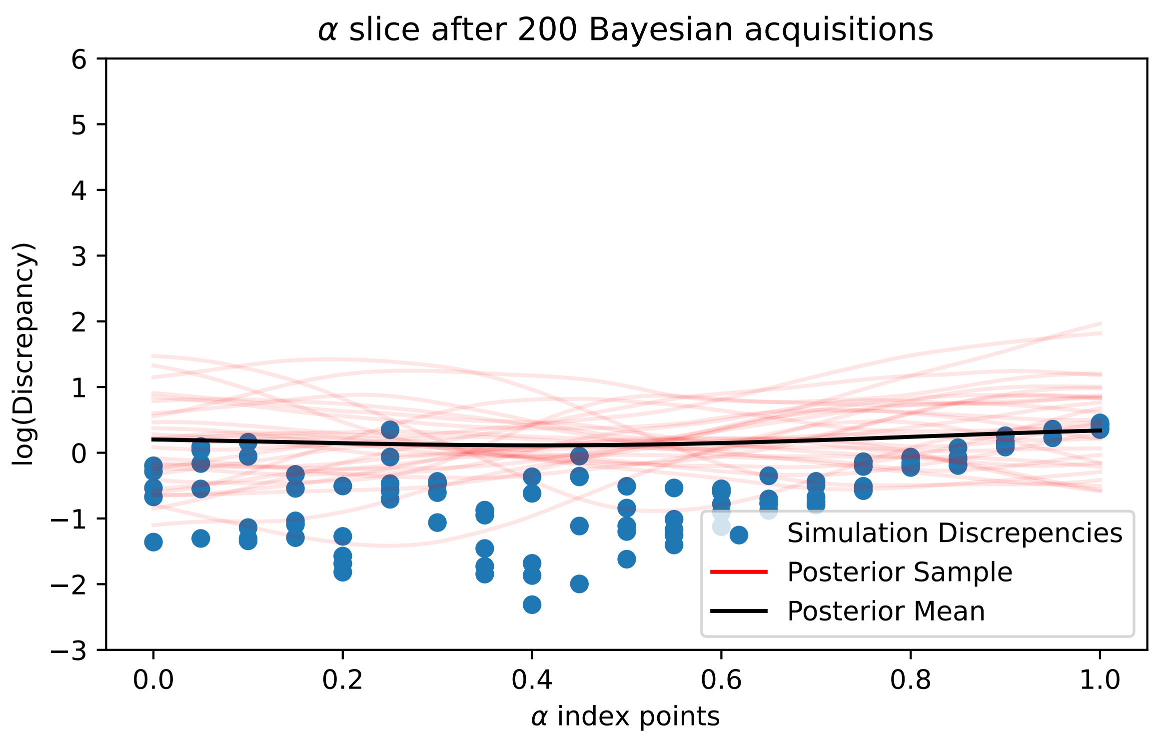

γ_L slice after 150 Bayesian acquisitions
log(Discrepancy)
γ_L index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


λ slice after 150 Bayesian acquisitions
log(Discrepancy)
λ index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

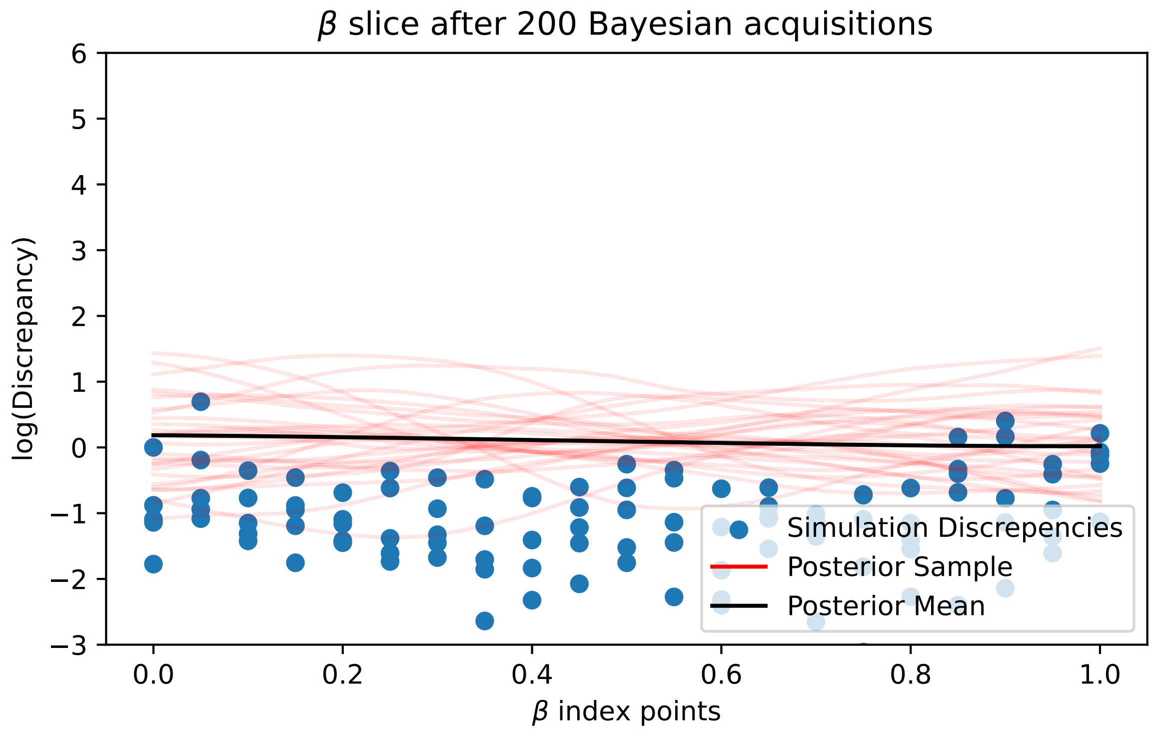

f slice after 150 Bayesian acquisitions
log(Discrepancy)
f index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



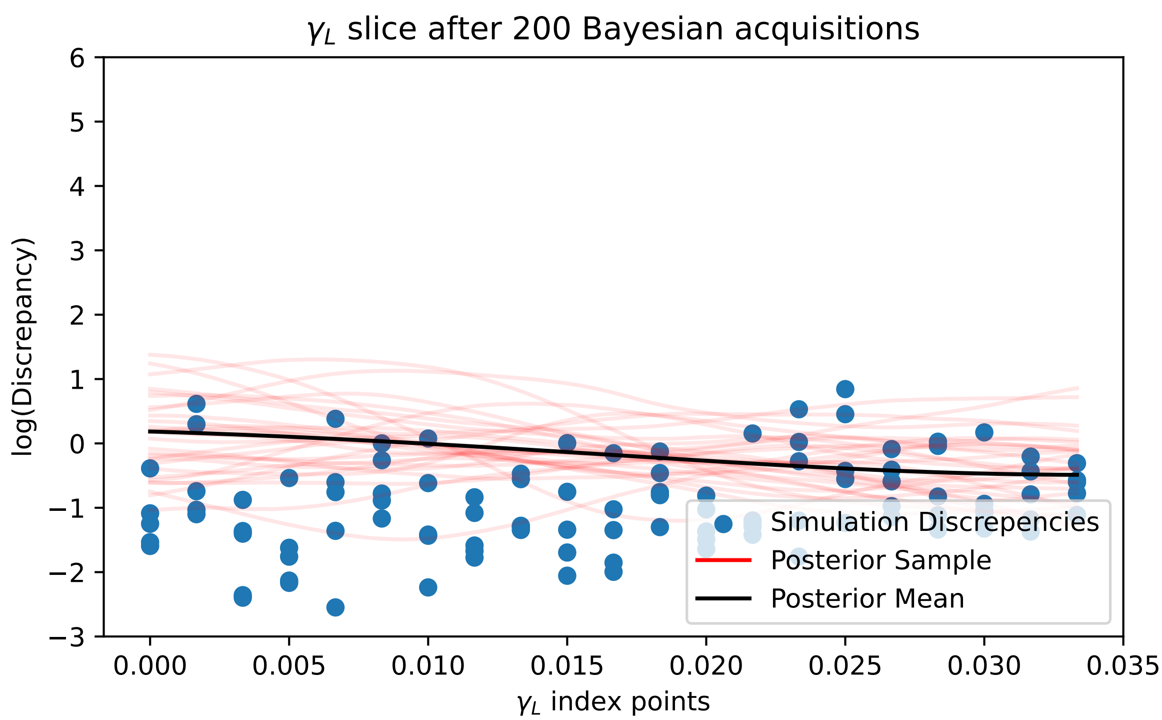
r slice after 150 Bayesian acquisitions
log(Discrepancy)
r index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



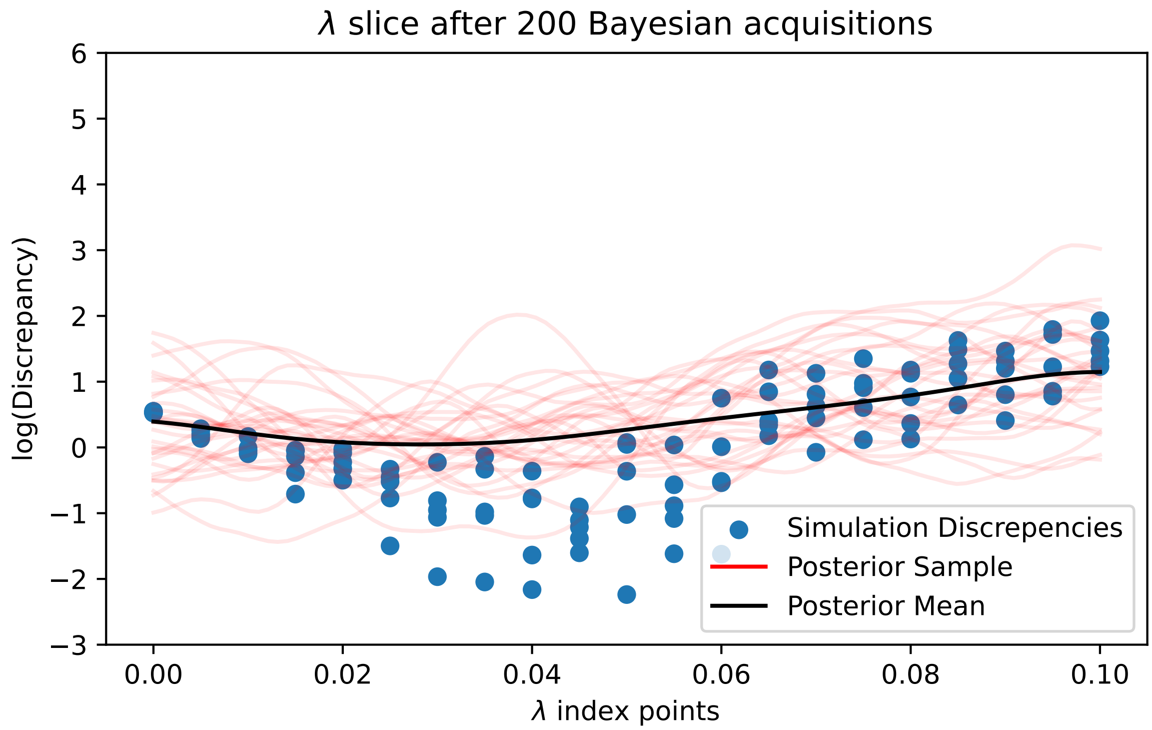
α slice after 200 Bayesian acquisitions
log(Discrepancy)
α index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

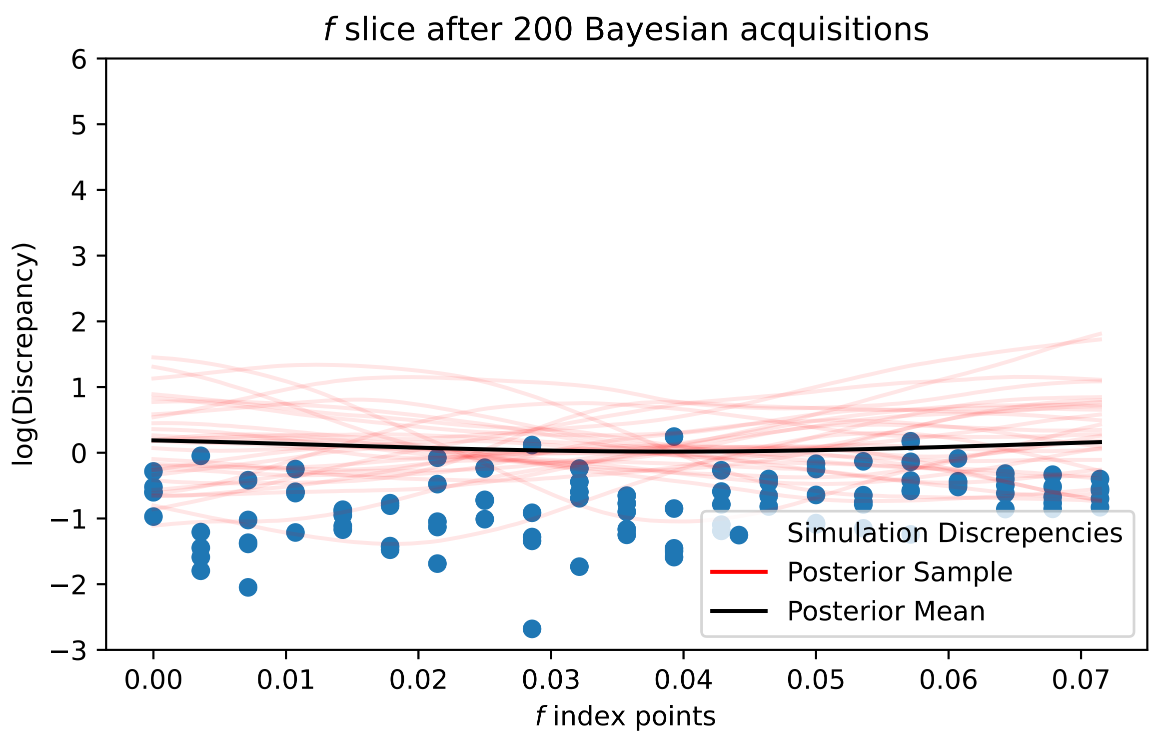


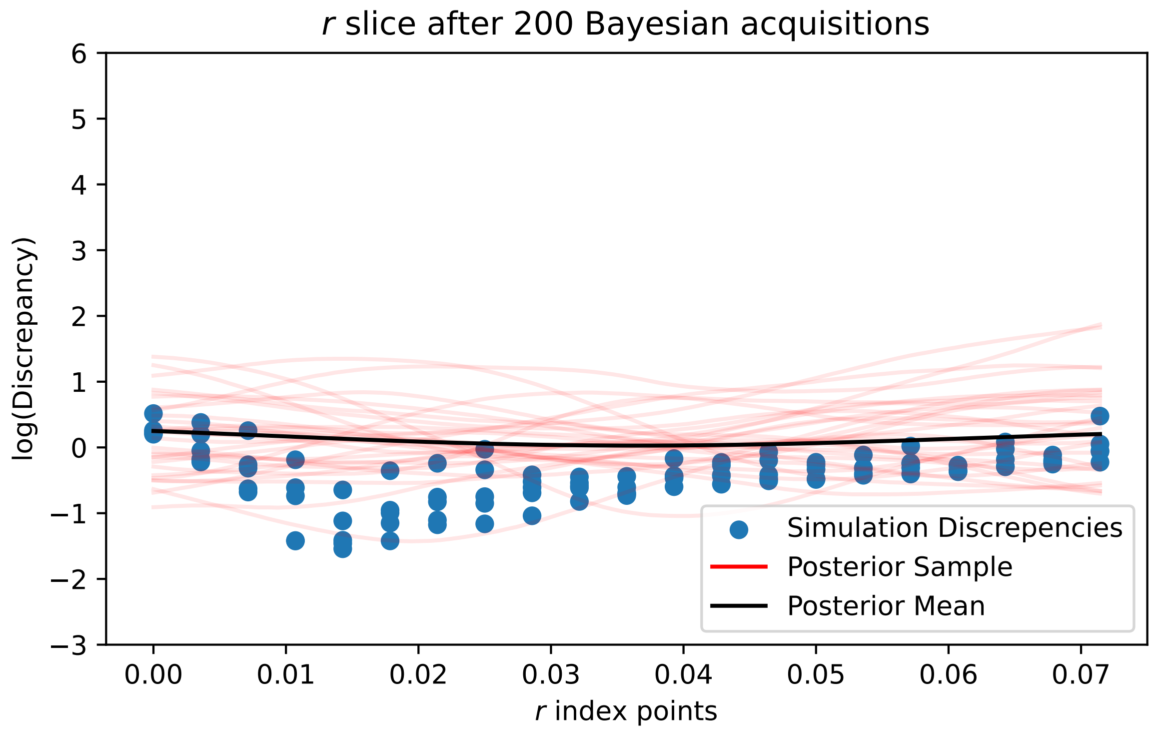
β slice after 200 Bayesian acquisitions
log(Discrepancy)
β index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



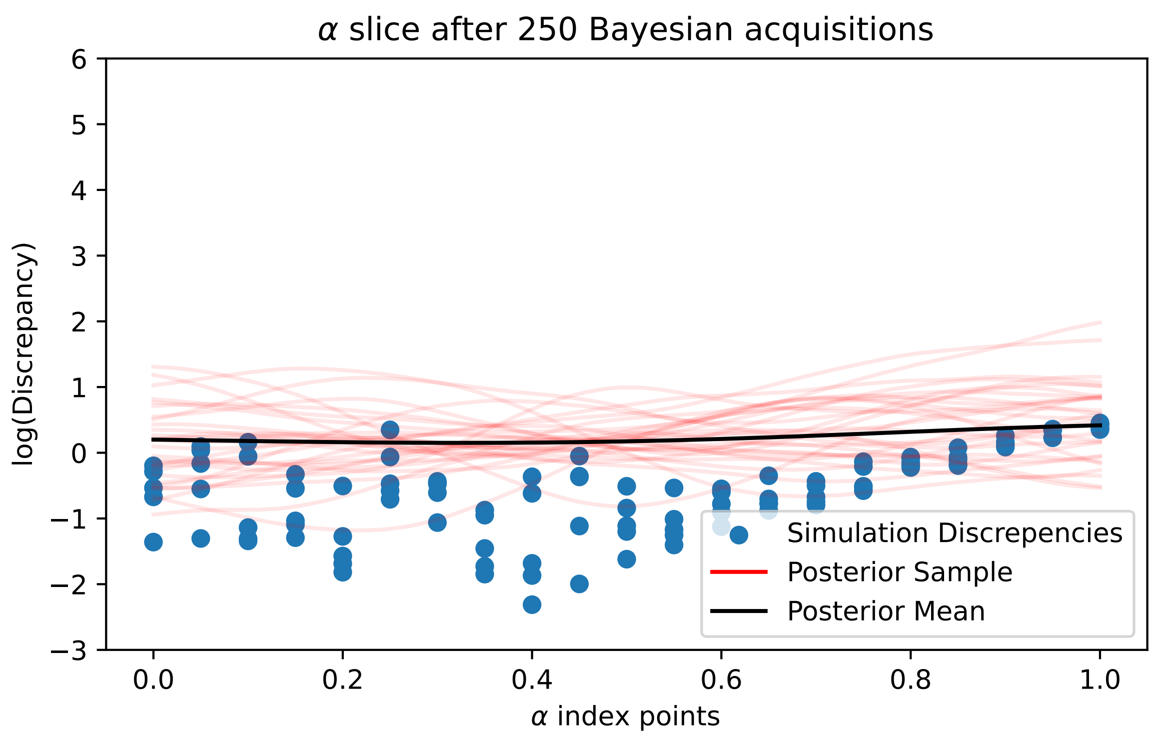
γ_L slice after 200 Bayesian acquisitions
log(Discrepancy)
γ_L index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


λ slice after 200 Bayesian acquisitions
log(Discrepancy)
λ index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

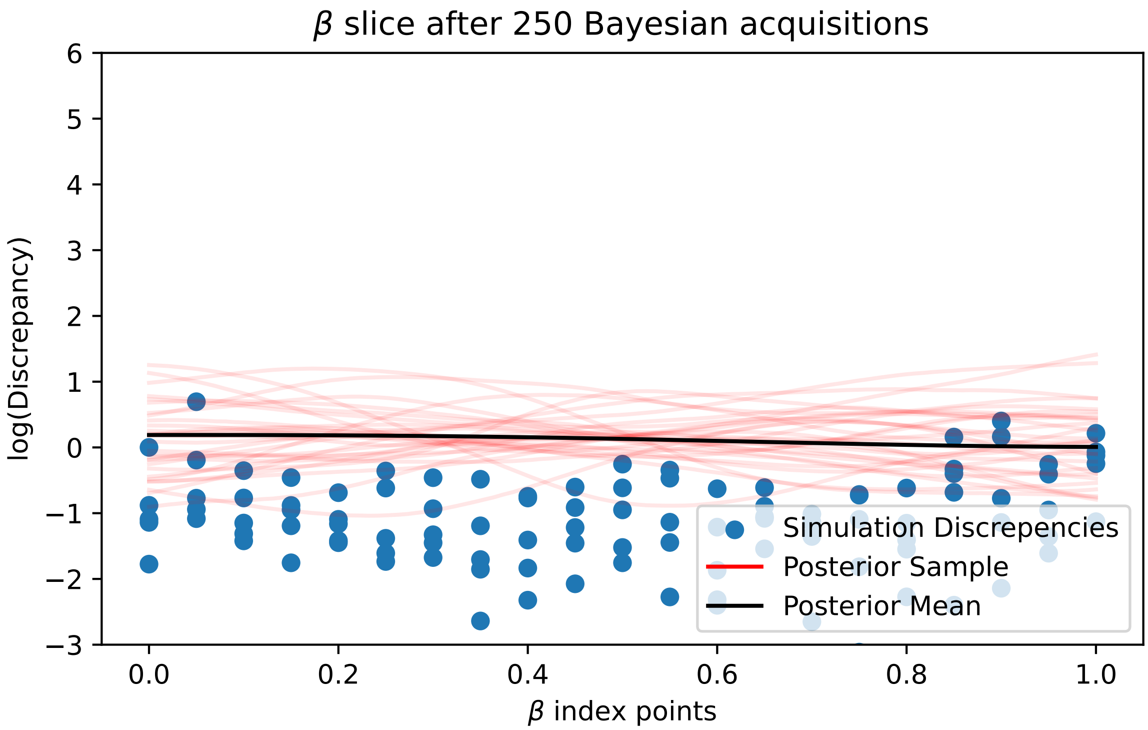

f slice after 200 Bayesian acquisitions
log(Discrepancy)
f index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

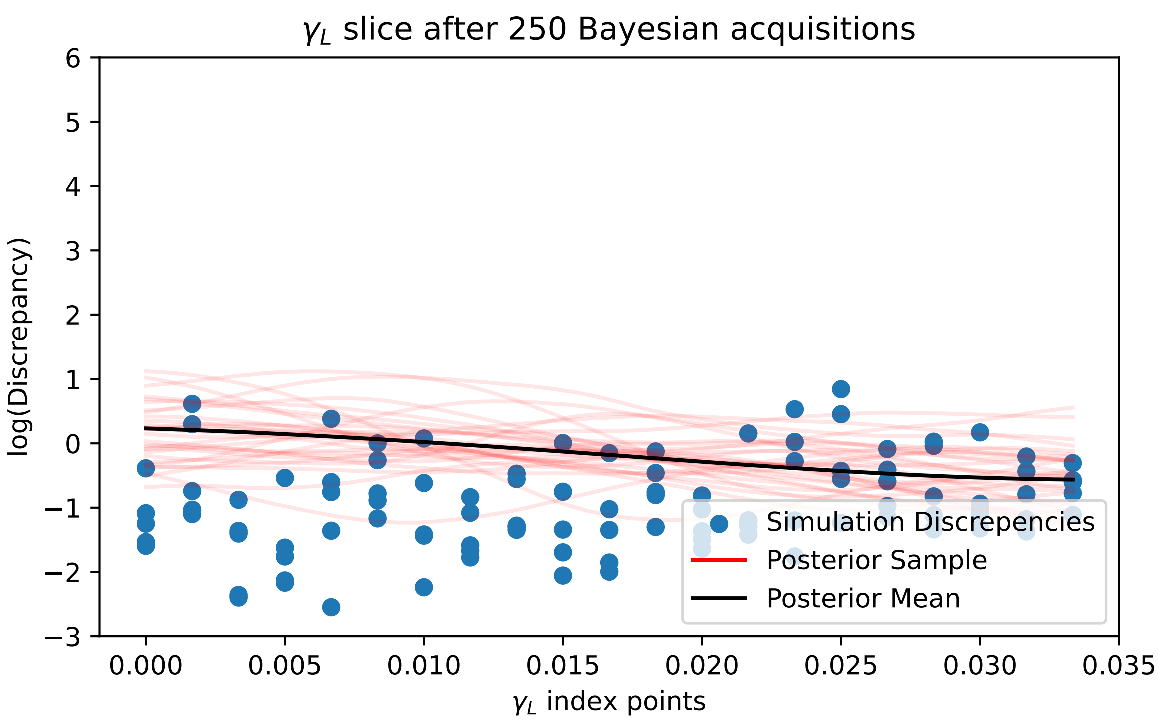


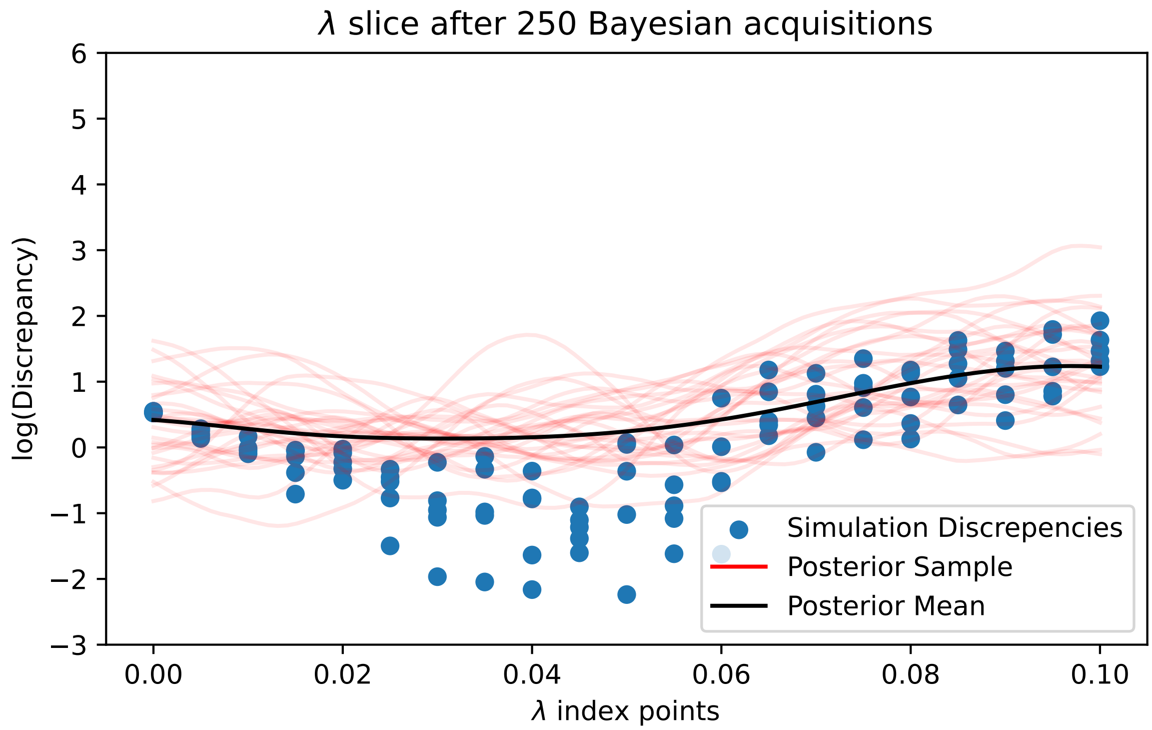
r slice after 200 Bayesian acquisitions
log(Discrepancy)
r index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



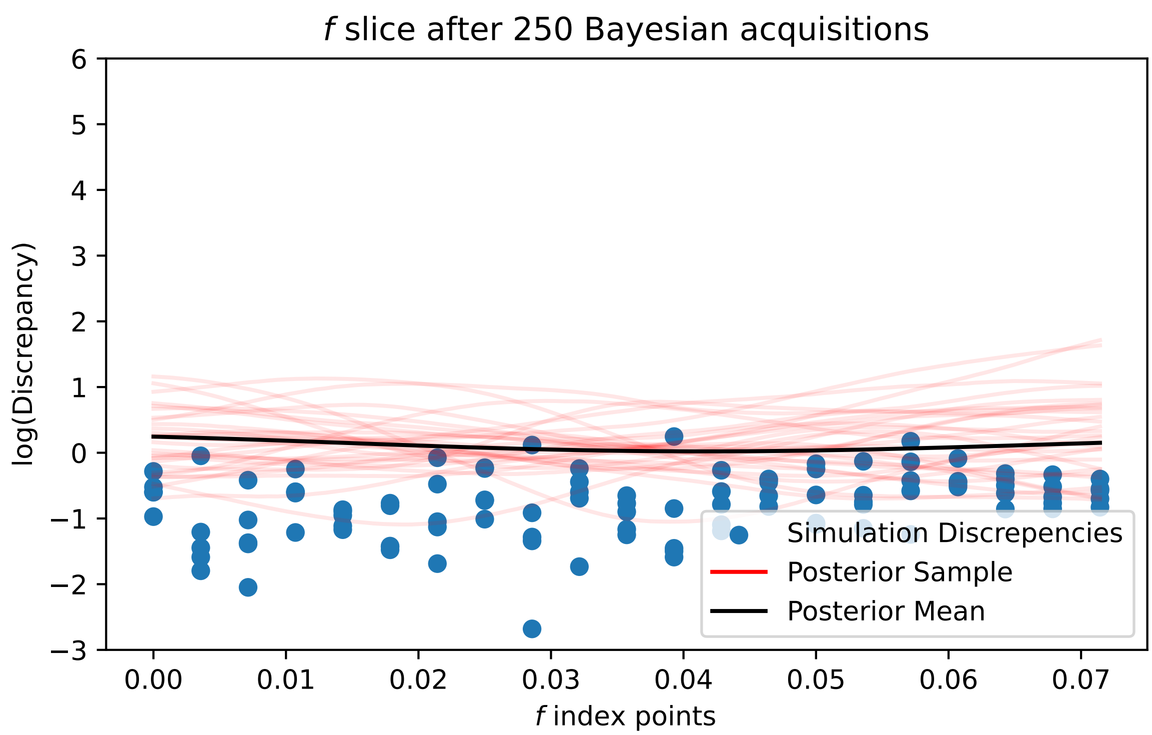
α slice after 250 Bayesian acquisitions
log(Discrepancy)
α index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


β slice after 250 Bayesian acquisitions
log(Discrepancy)
β index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

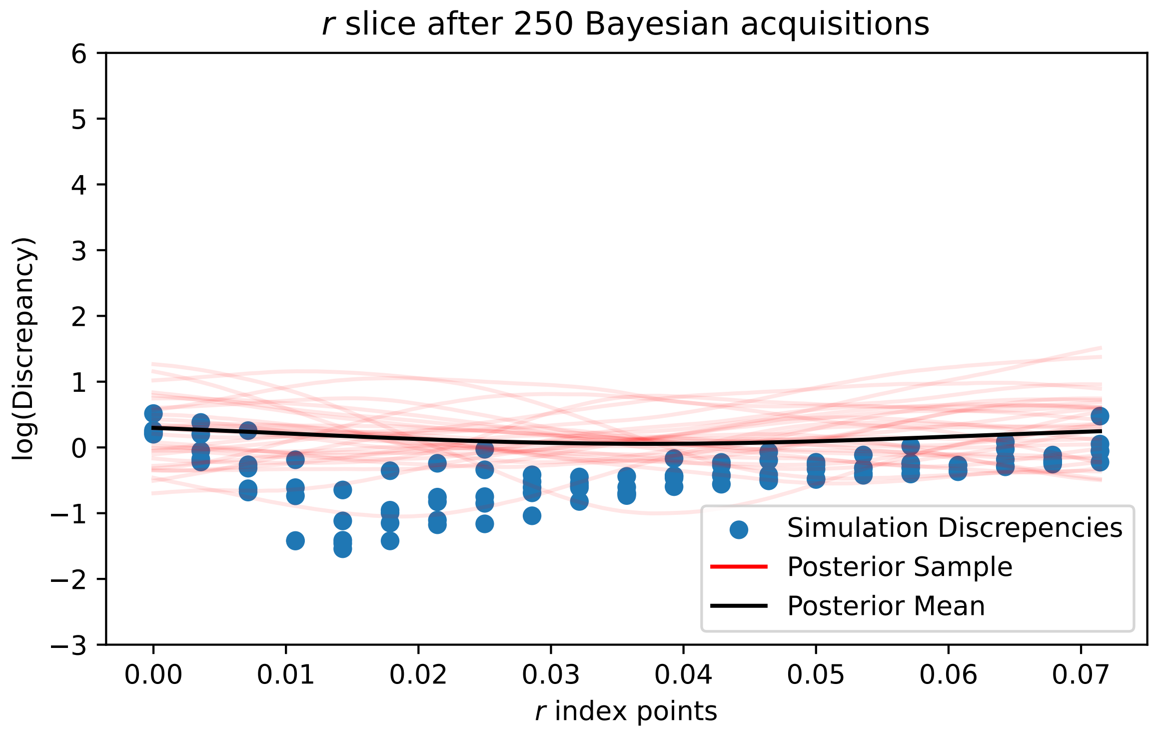

γ_L slice after 250 Bayesian acquisitions
log(Discrepancy)
γ_L index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



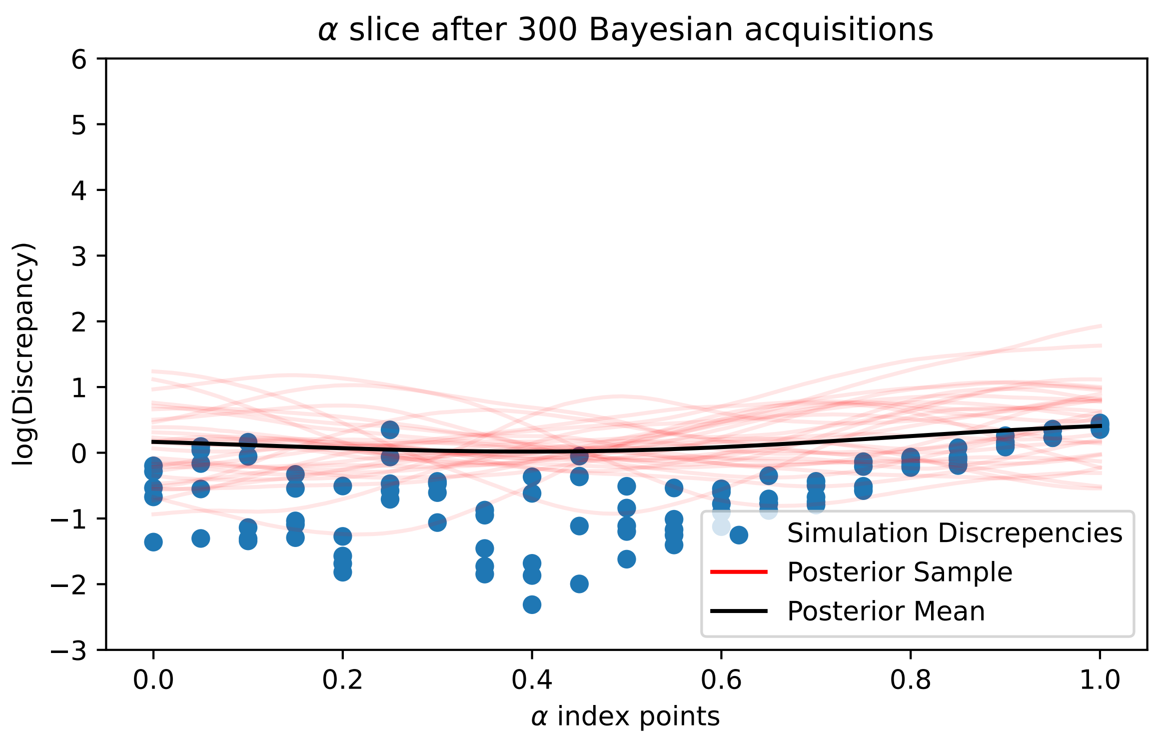
λ slice after 250 Bayesian acquisitions
log(Discrepancy)
λ index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



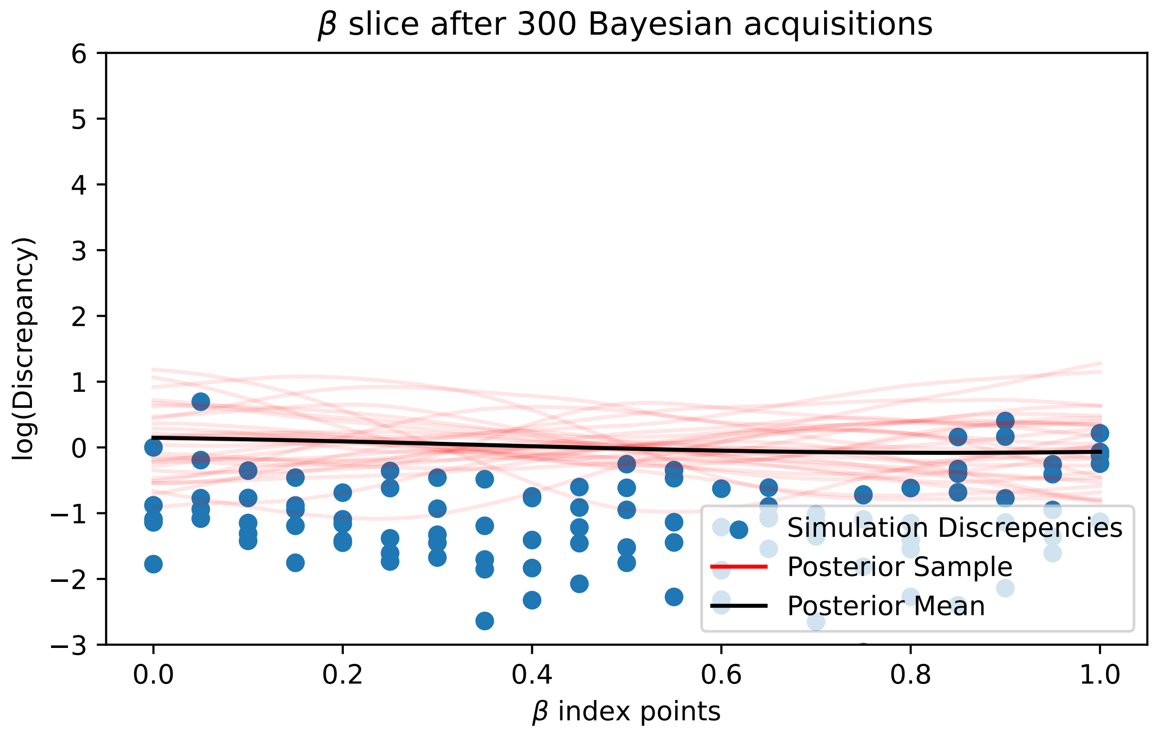
f slice after 250 Bayesian acquisitions
log(Discrepancy)
f index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

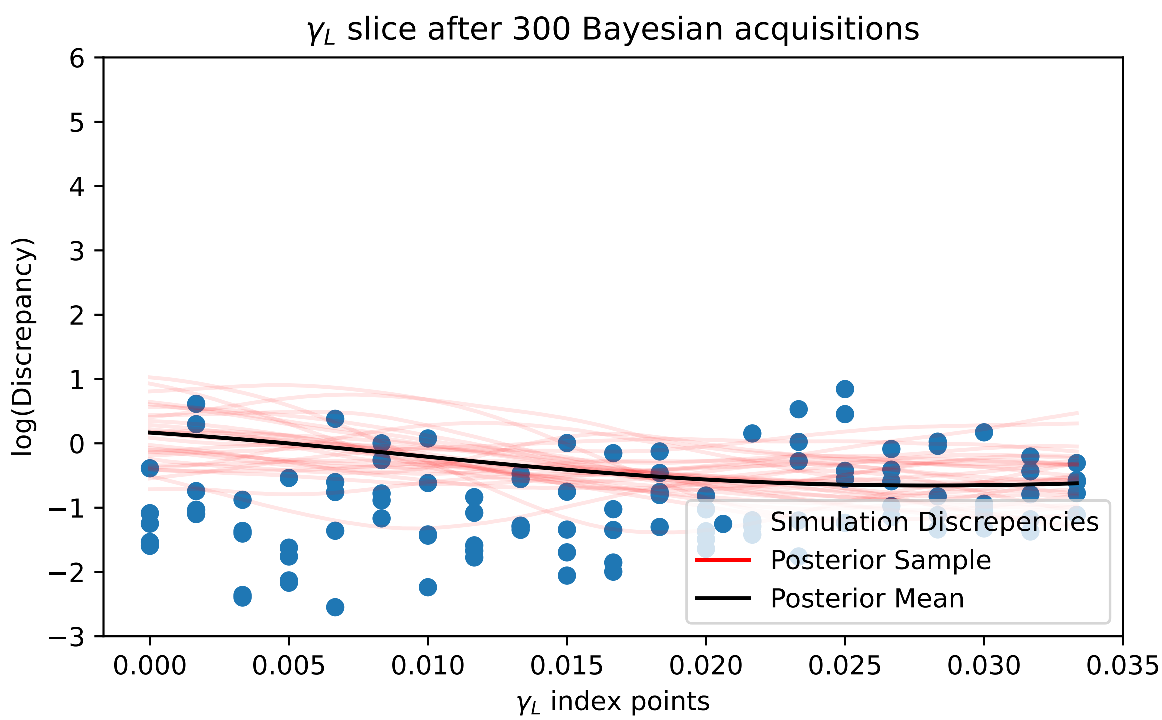


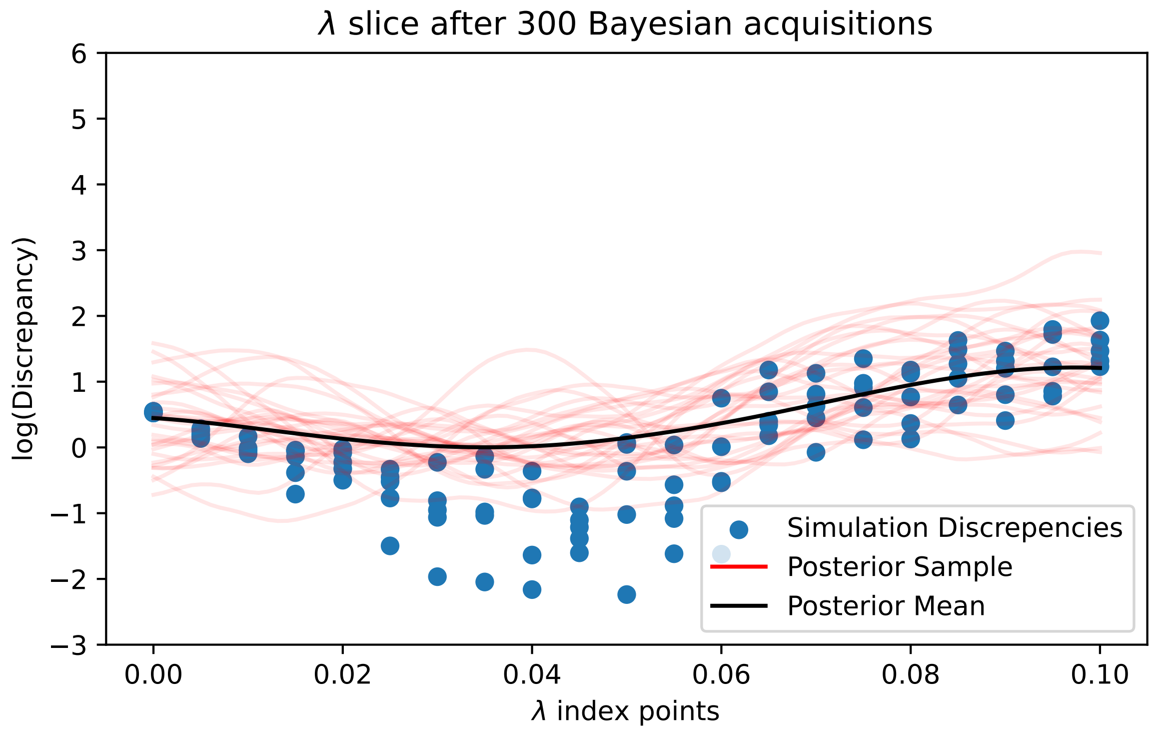
r slice after 250 Bayesian acquisitions
log(Discrepancy)
6
5
4
3
2
1
0
−1
−2
−3
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
r index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

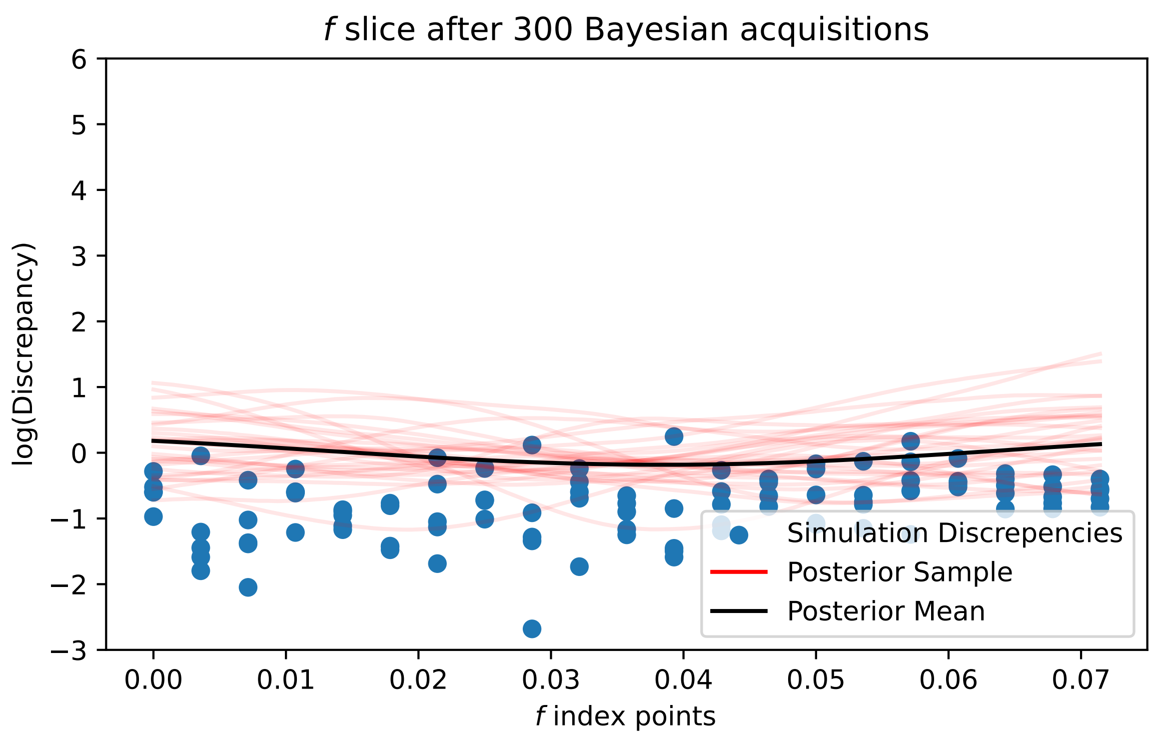


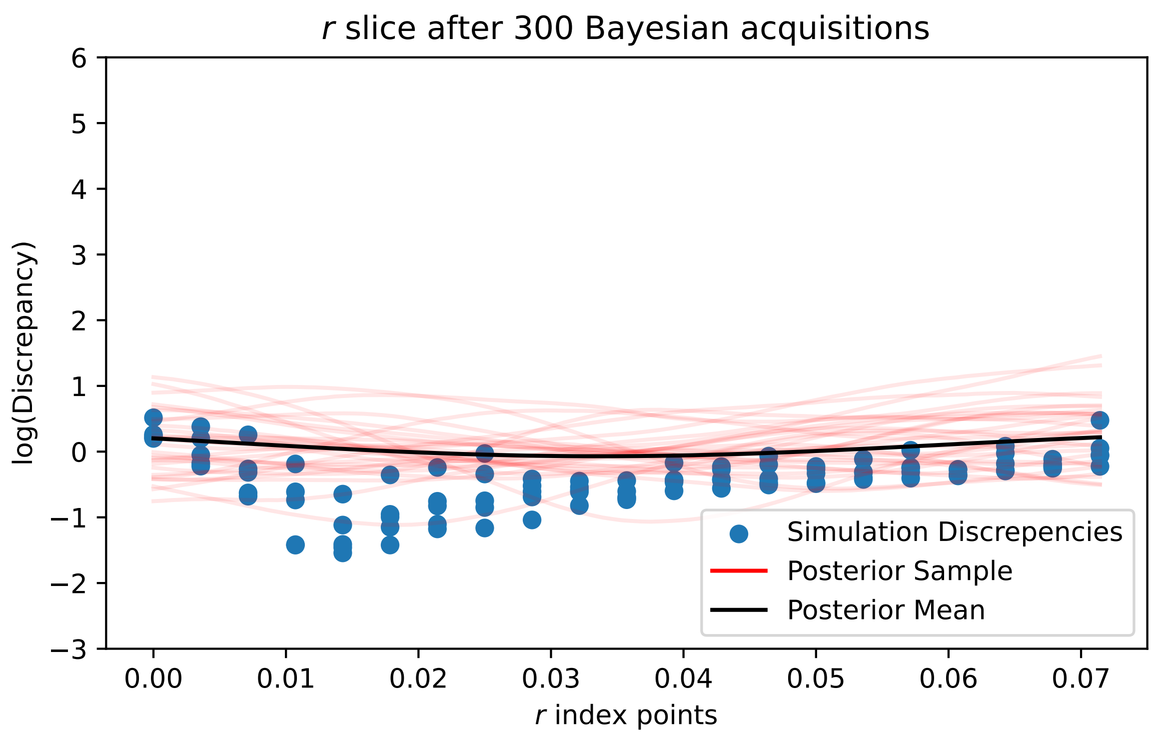
α slice after 300 Bayesian acquisitions
log(Discrepancy)
6
5
4
3
2
1
0
−1
−2
−3
0.0
0.2
0.4
0.6
0.8
1.0
α index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


β slice after 300 Bayesian acquisitions
log(Discrepancy)
β index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

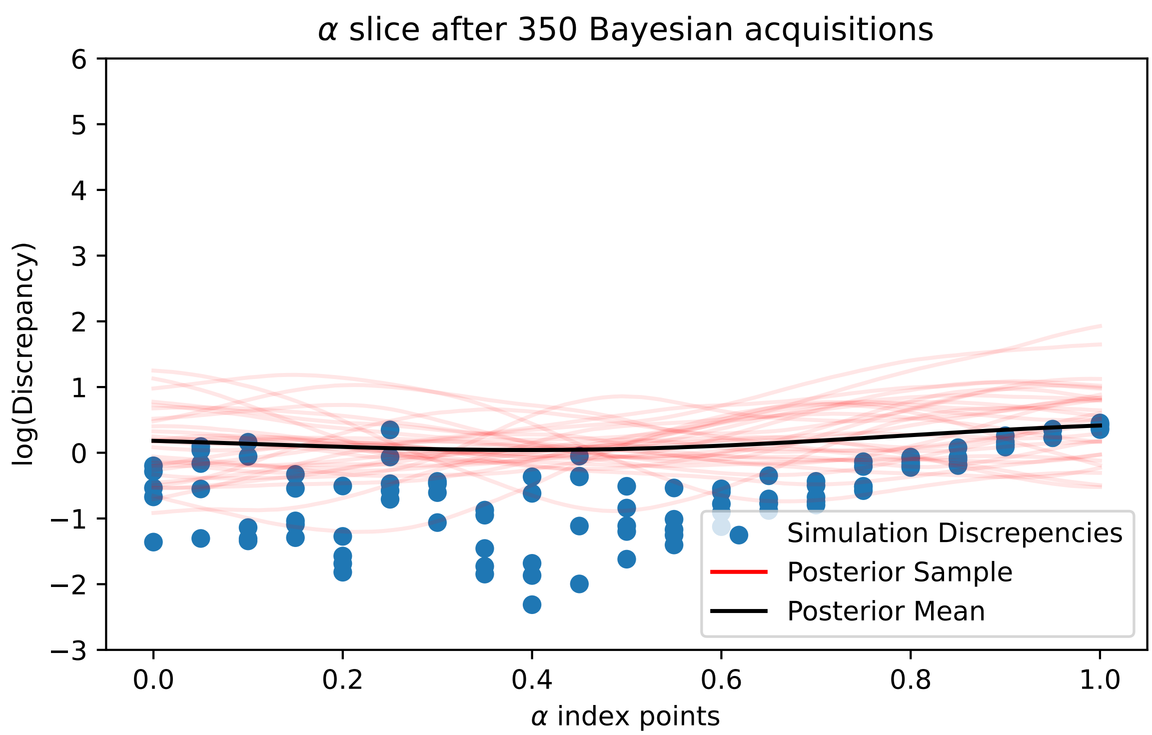

γ_L slice after 300 Bayesian acquisitions
log(Discrepancy)
γ_L index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


λ slice after 300 Bayesian acquisitions
log(Discrepancy)
λ index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

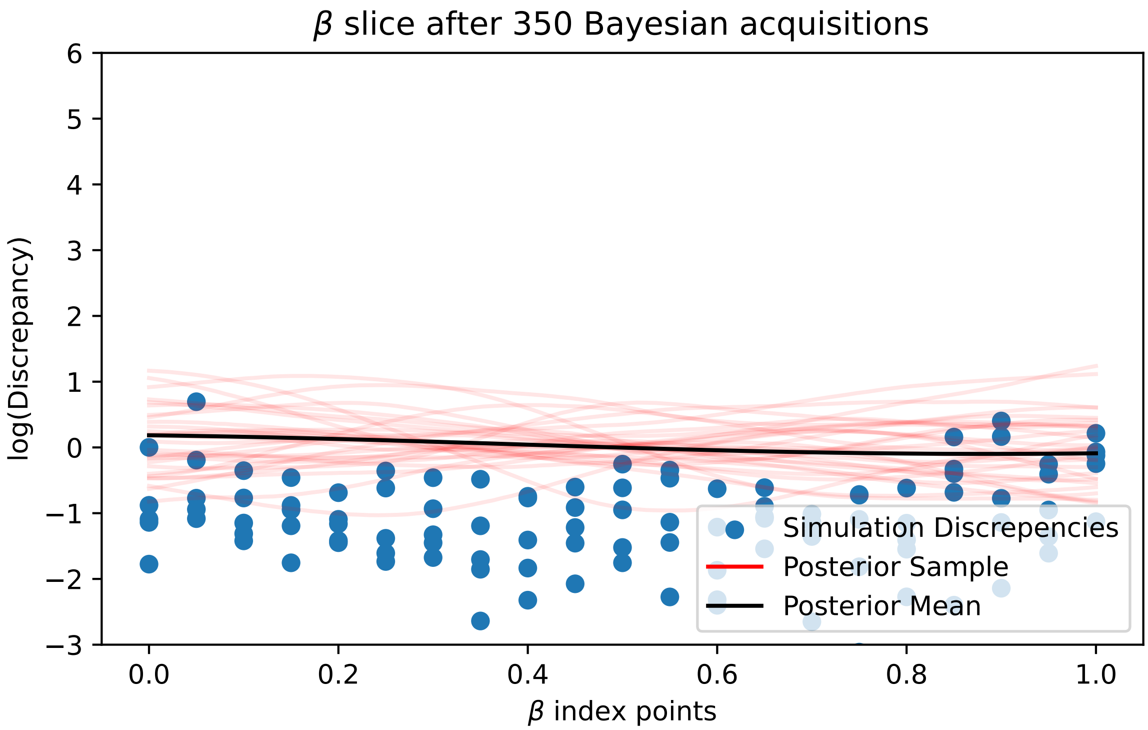

f slice after 300 Bayesian acquisitions
log(Discrepancy)
f index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



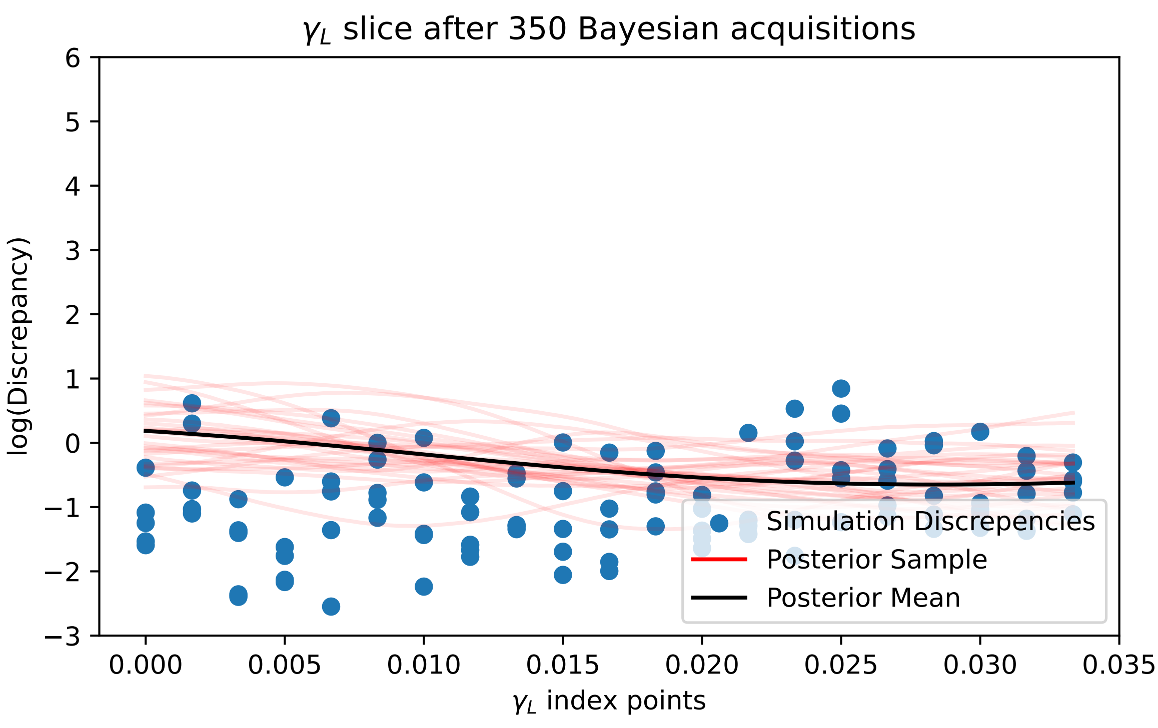
r slice after 300 Bayesian acquisitions
log(Discrepancy)
r index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



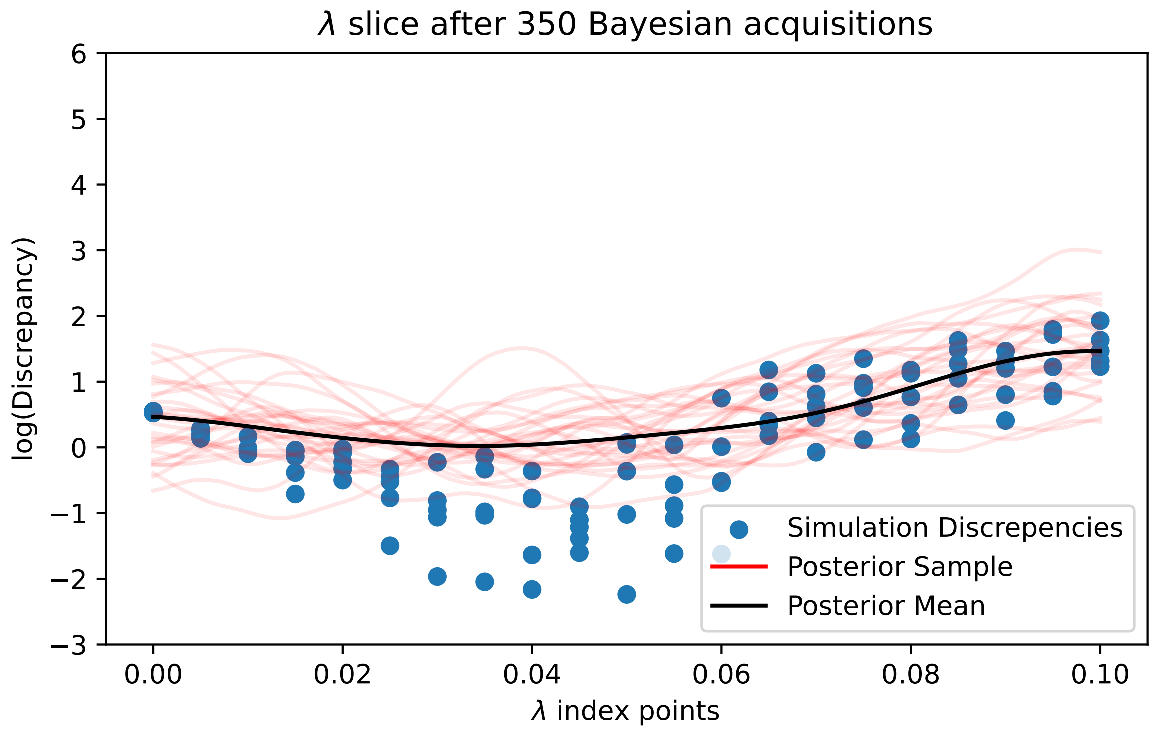
α slice after 350 Bayesian acquisitions
log(Discrepancy)
α index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

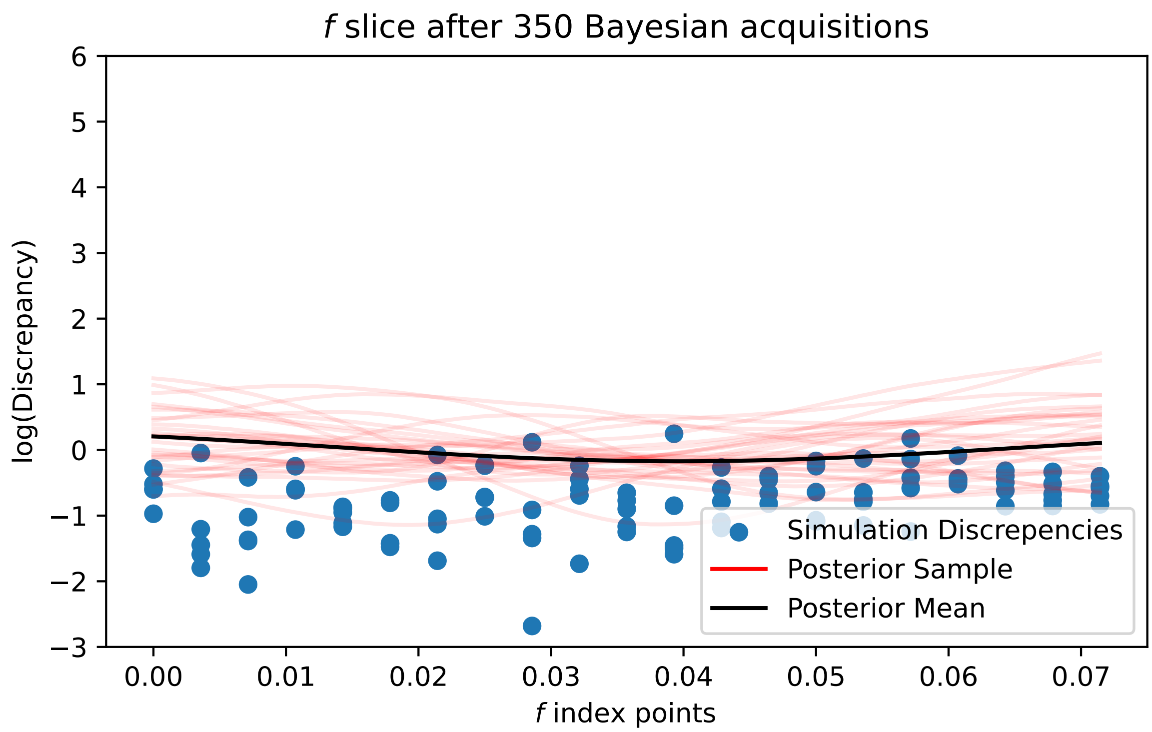


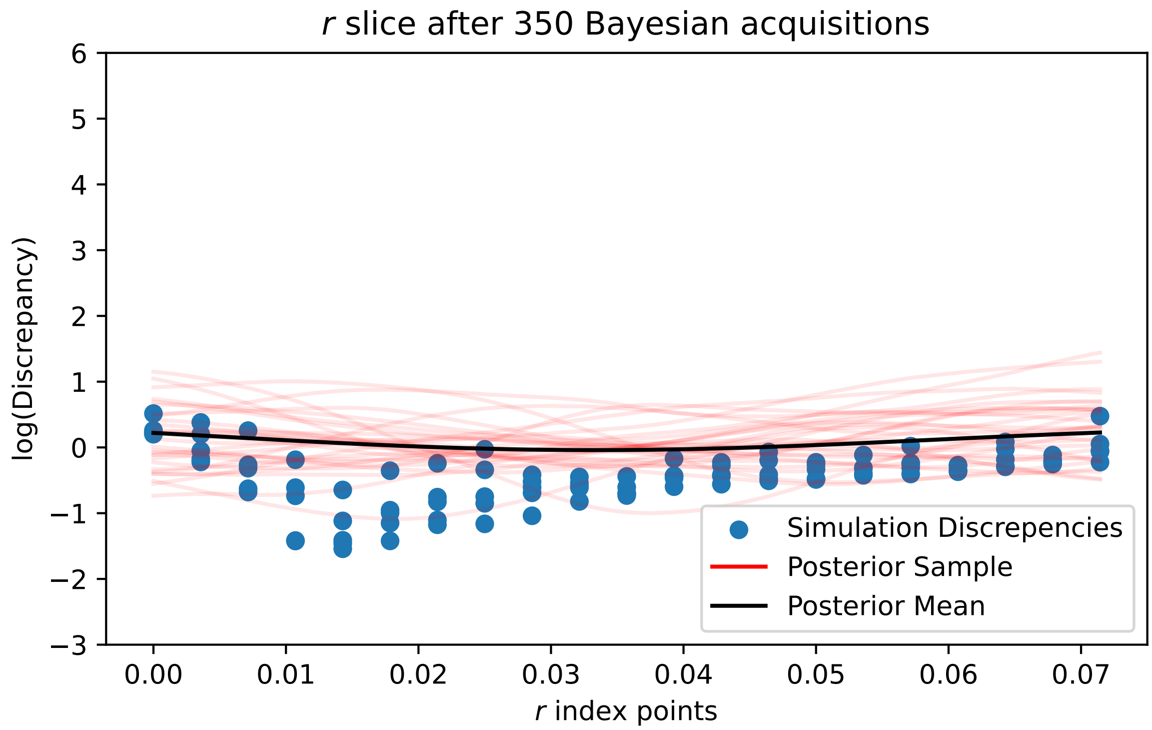
β slice after 350 Bayesian acquisitions
log(Discrepancy)
β index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

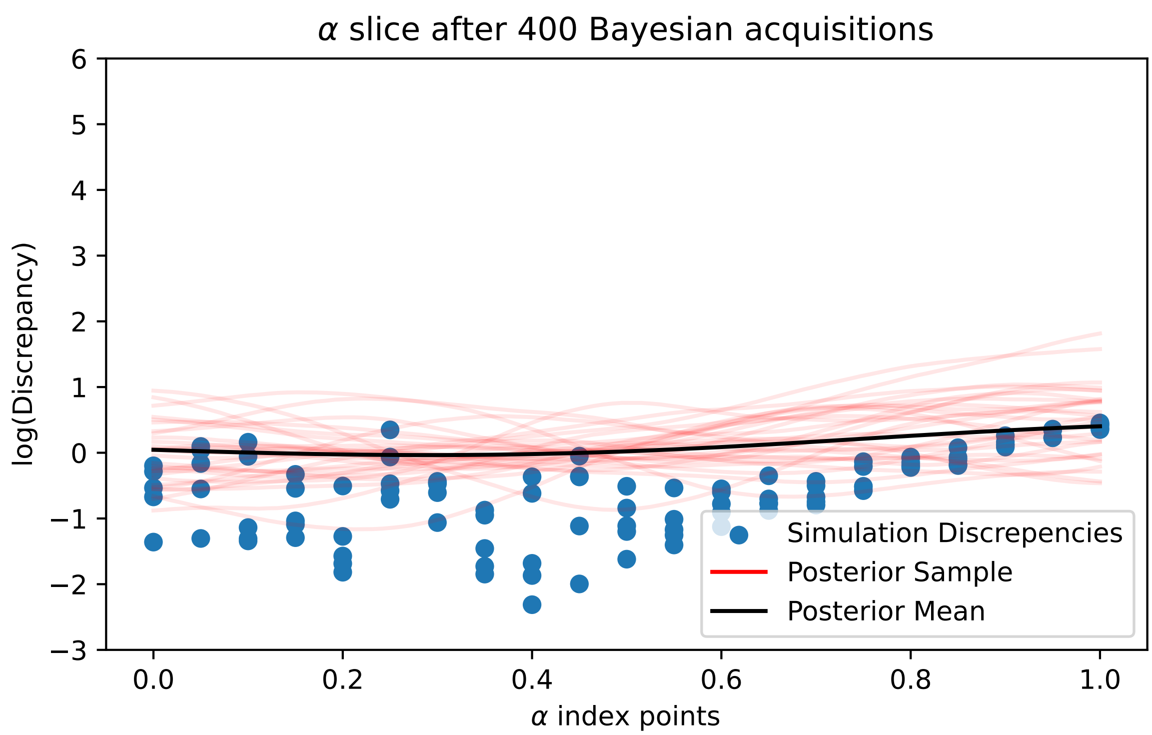


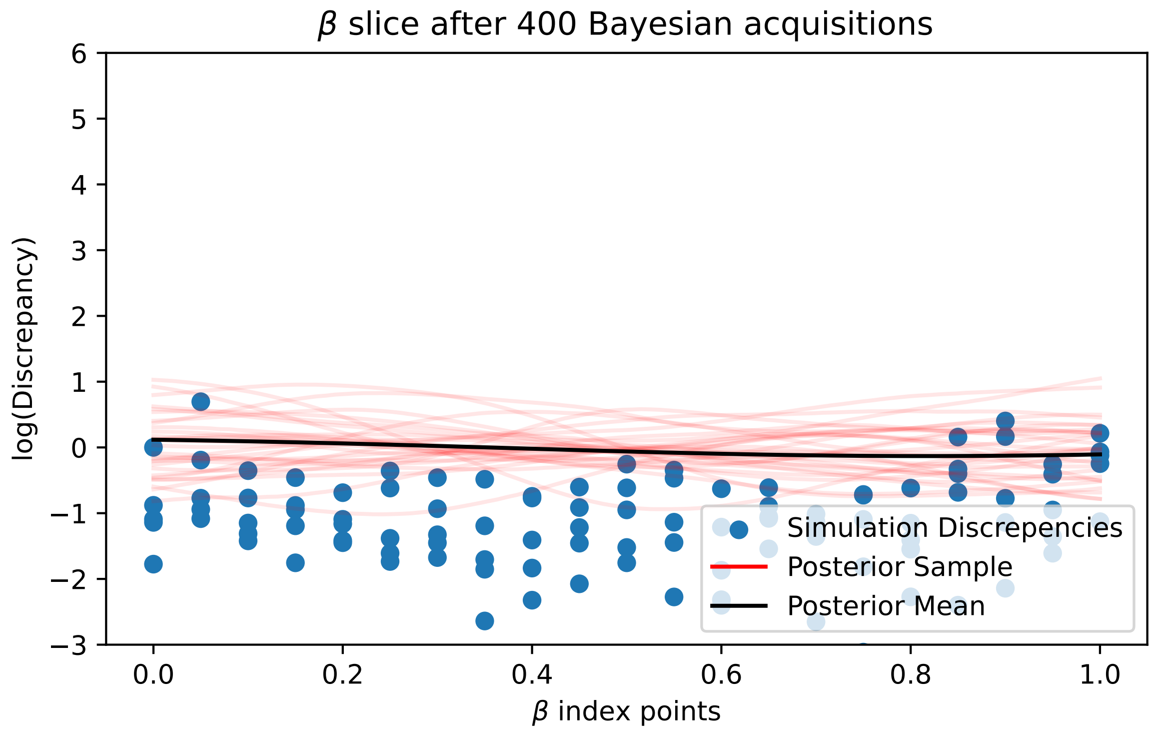
γL slice after 350 Bayesian acquisitions
log(Discrepancy)
γL index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

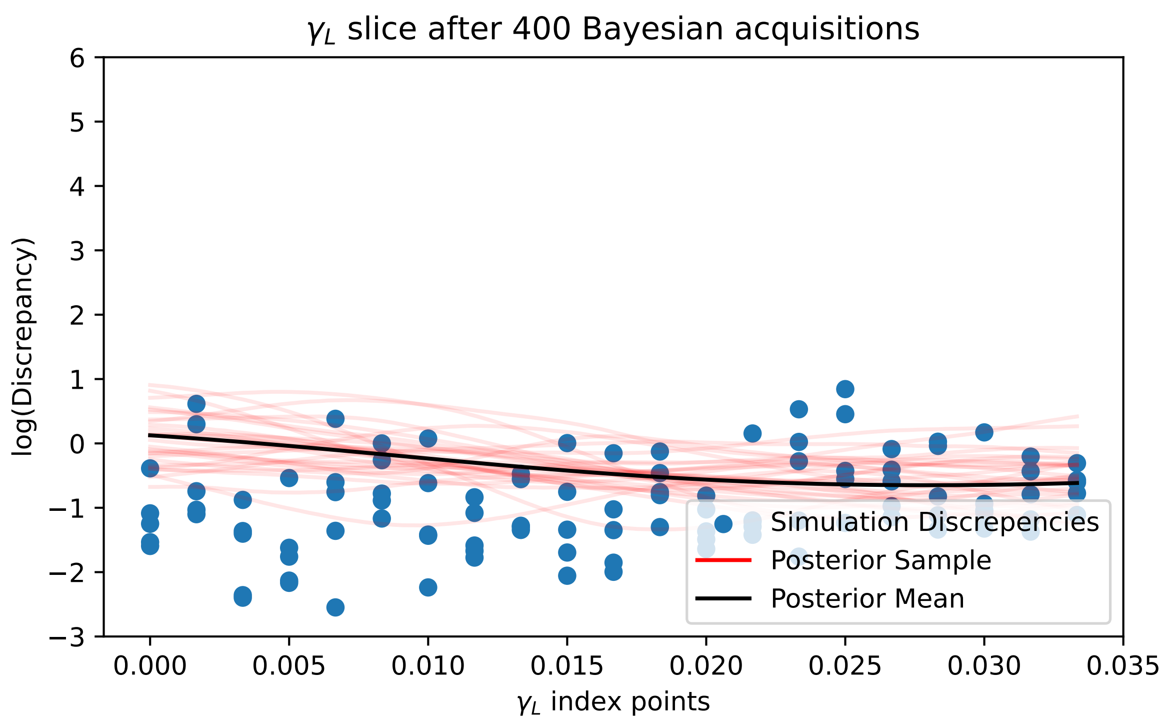

λ slice after 350 Bayesian acquisitions
log(Discrepancy)
λ index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

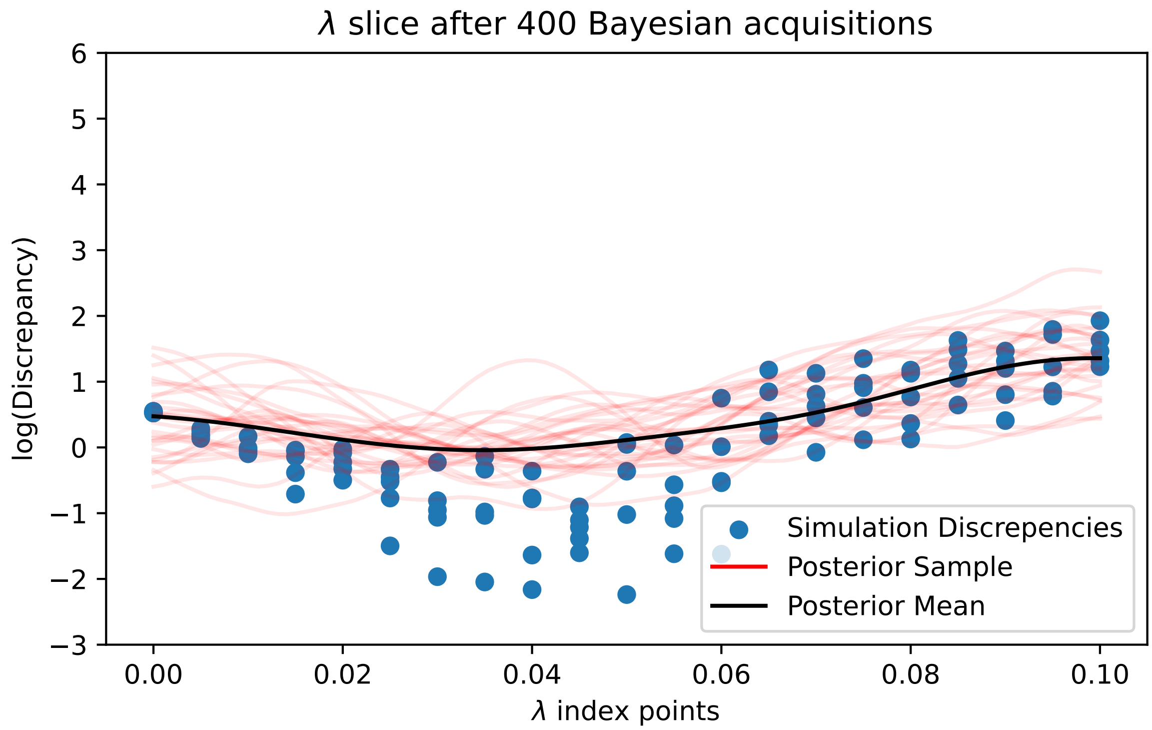

f slice after 350 Bayesian acquisitions
log(Discrepancy)
f index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

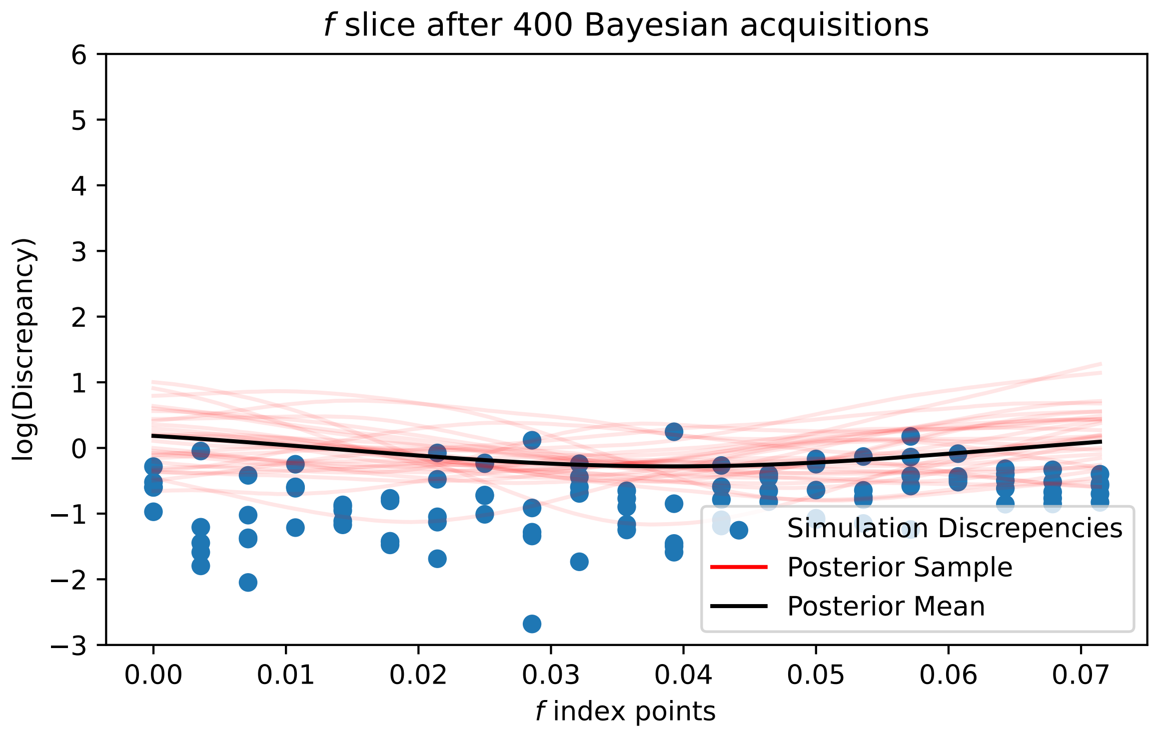

r slice after 350 Bayesian acquisitions
log(Discrepancy)
r index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

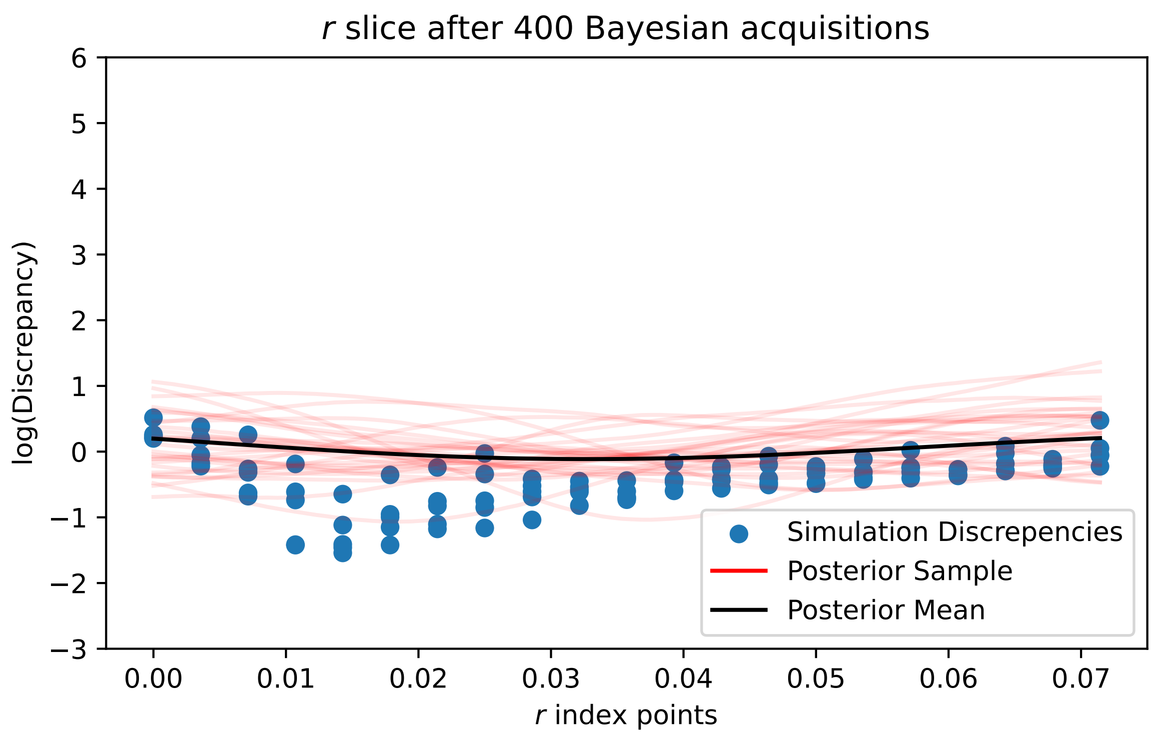

α slice after 400 Bayesian acquisitions
log(Discrepancy)
α index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


β slice after 400 Bayesian acquisitions
log(Discrepancy)
β index points
Simulation Discrepencies
Posterior Sample
Posterior Mean


γ_L slice after 400 Bayesian acquisitions
log(Discrepancy)
γ_L index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

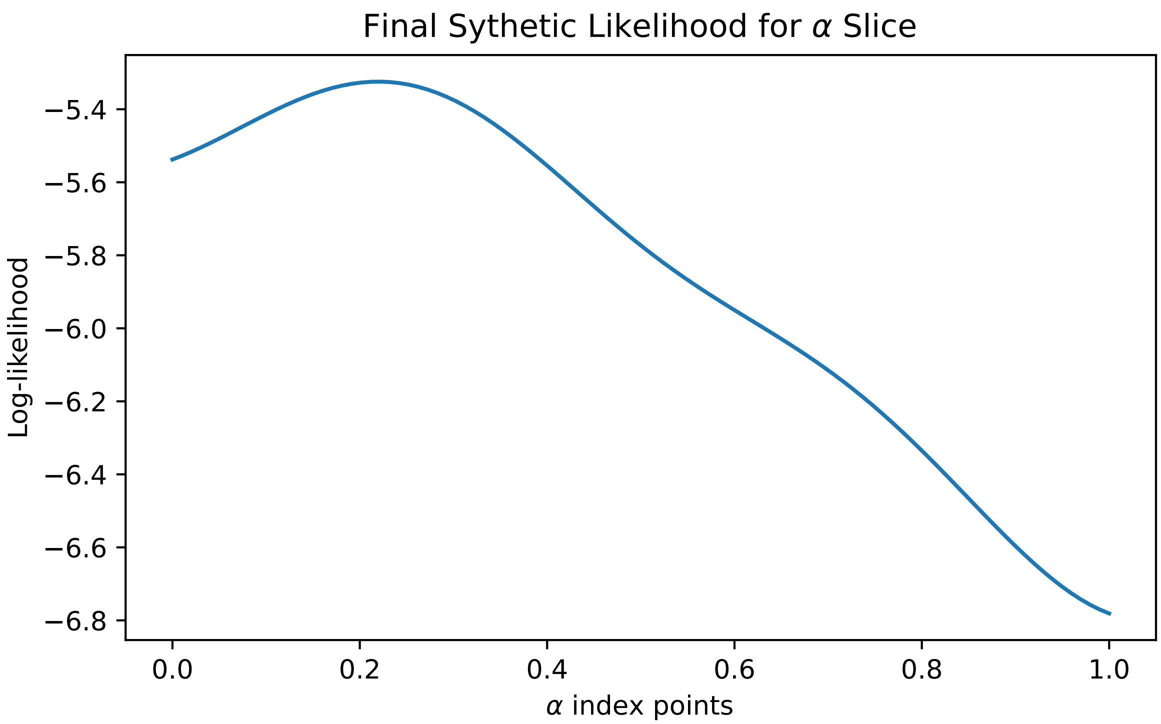

λ slice after 400 Bayesian acquisitions
log(Discrepancy)
6
5
4
3
2
1
0
−1
−2
−3
0.00
0.02
0.04
0.06
0.08
0.10
λ index points
Simulation Discrepencies
Posterior Sample
Posterior Mean

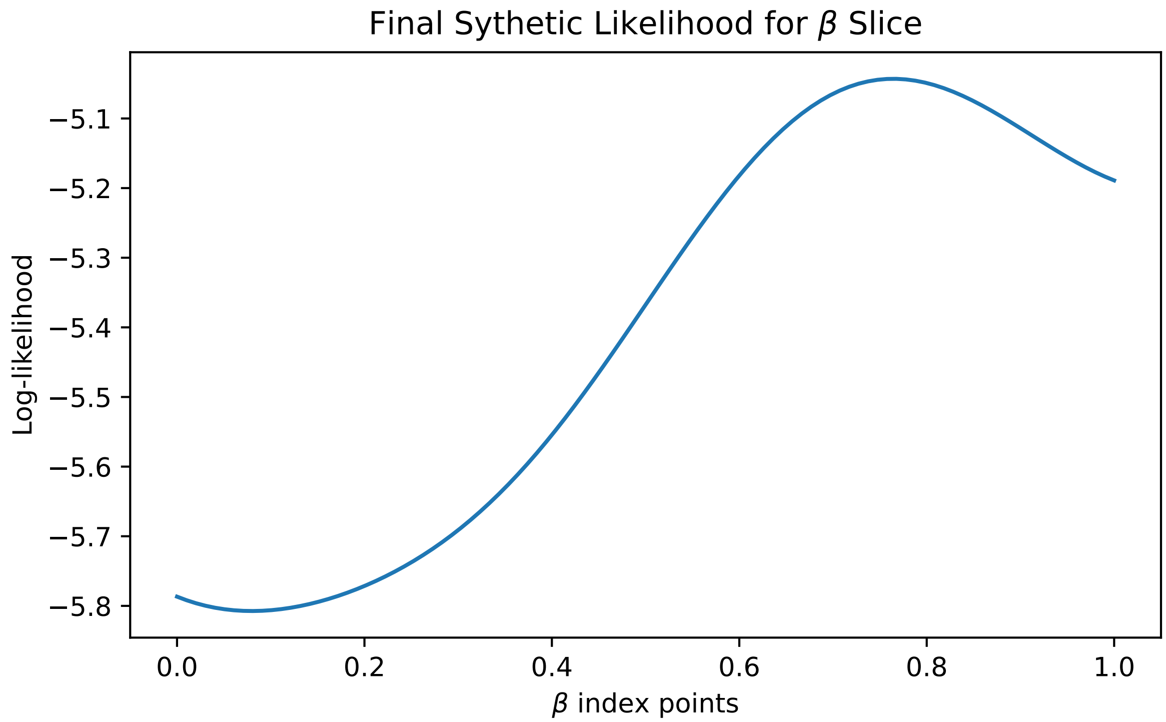

f slice after 400 Bayesian acquisitions
log(Discrepancy)
6
5
4
3
2
1
0
−1
−2
−3
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
f index points
Simulation Discrepencies
Posterior Sample
Posterior Mean



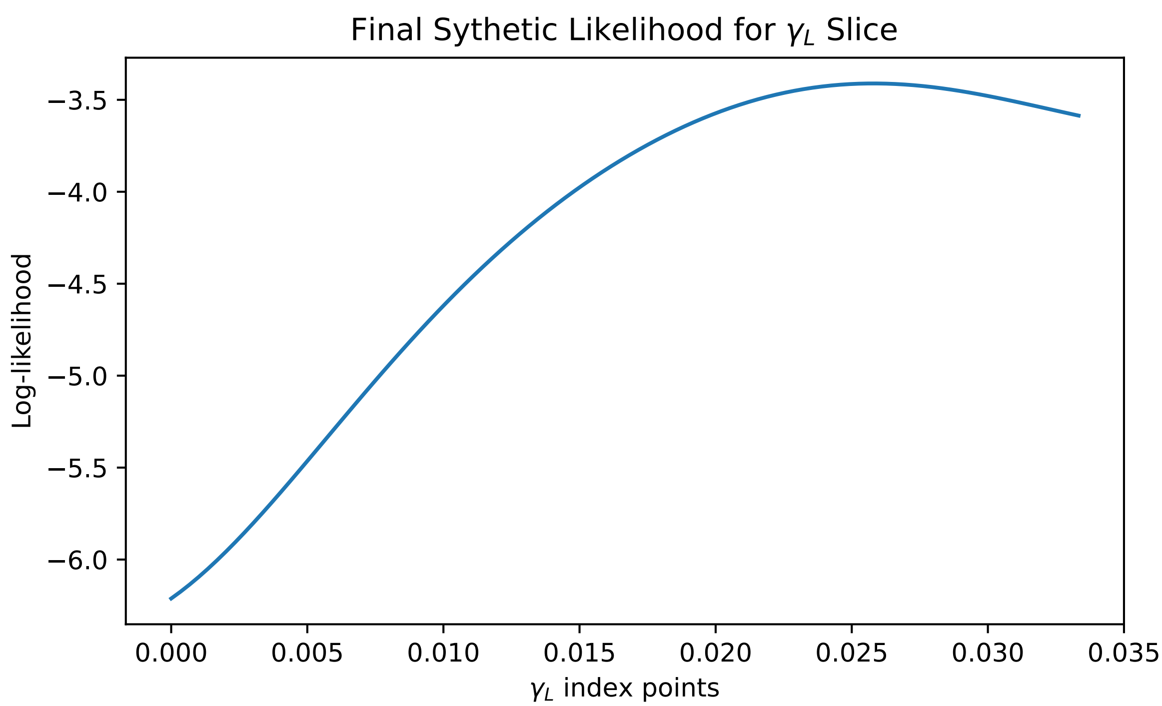

```
epsilon = -2.
for var in vars:
    champ_GP_reg = tfd.GaussianProcessRegressionModel(
        kernel=kernel_champ,
        index_points=slice_indices_dfs_dict[var + "_gp_indices_df"].values,
        observation_index_points=index_vals,
        observations=obs_vals,
        observation_noise_variance=observation_noise_variance_champ,
        predictive_noise_variance=0.0,
        mean_fn=const_mean_fn(),
    )

    indices_for_lik = slice_indices_dfs_dict[var + "_gp_indices_df"].values

    mean = champ_GP_reg.mean_fn(indices_for_lik)
    variance = champ_GP_reg.variance(index_points=indices_for_lik)
    post_std = np.sqrt(variance + observation_noise_variance_champ._value().numpy())
    cdf_vals = tfd.Normal(mean, post_std).log_cdf(epsilon)

    plt.figure(figsize=(7, 4))
```

```
    plt.plot(
        slice_indices_dfs_dict[var + "_gp_indices_df"][var].values,
        cdf_vals,
    )
    if var in ["f", "r"]:
        plt.xlabel("$" + var + "$ index points")
        plt.title("Final Sythetic Likelihood for $" + var + "$ Slice")
    else:
        plt.xlabel("$\\" + var + "$ index points")
        plt.title("Final Sythetic Likelihood for $\\" + var + "$ Slice")
    plt.ylabel("Log-likelihood")
    plt.savefig(
        "champagne_GP_images/"
        + var
        + "_slice_"
        + str(t)
        + "_synth_likelihood.pdf"
    )
    plt.show()
```

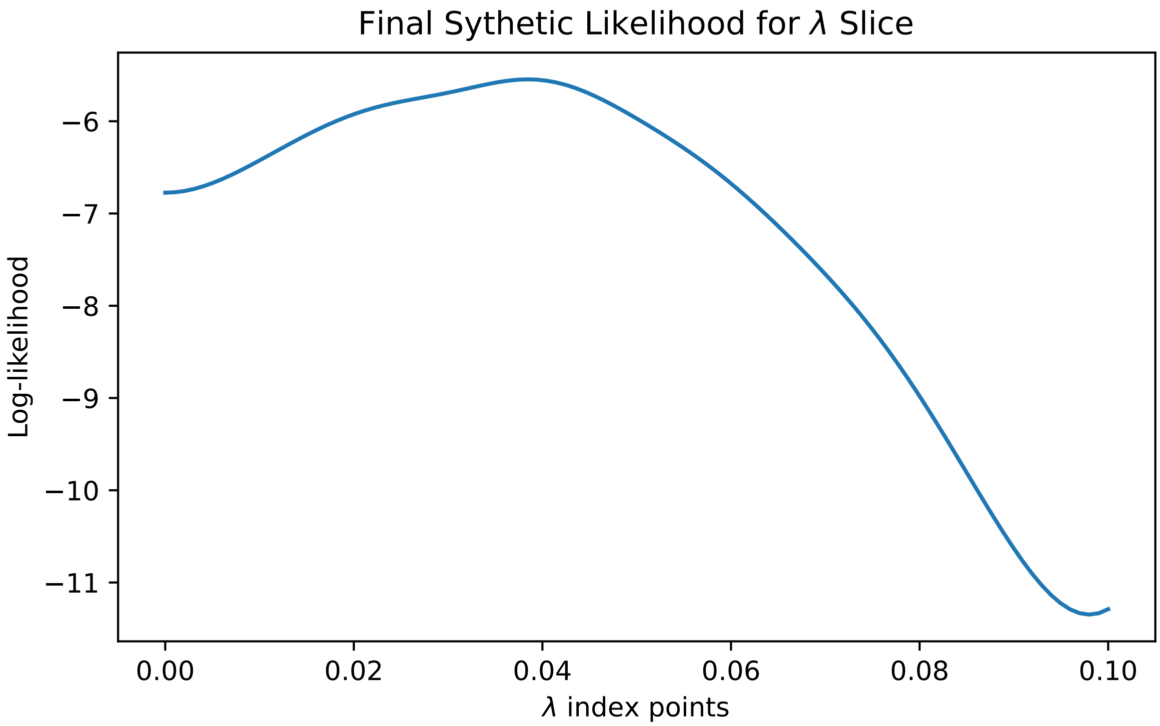


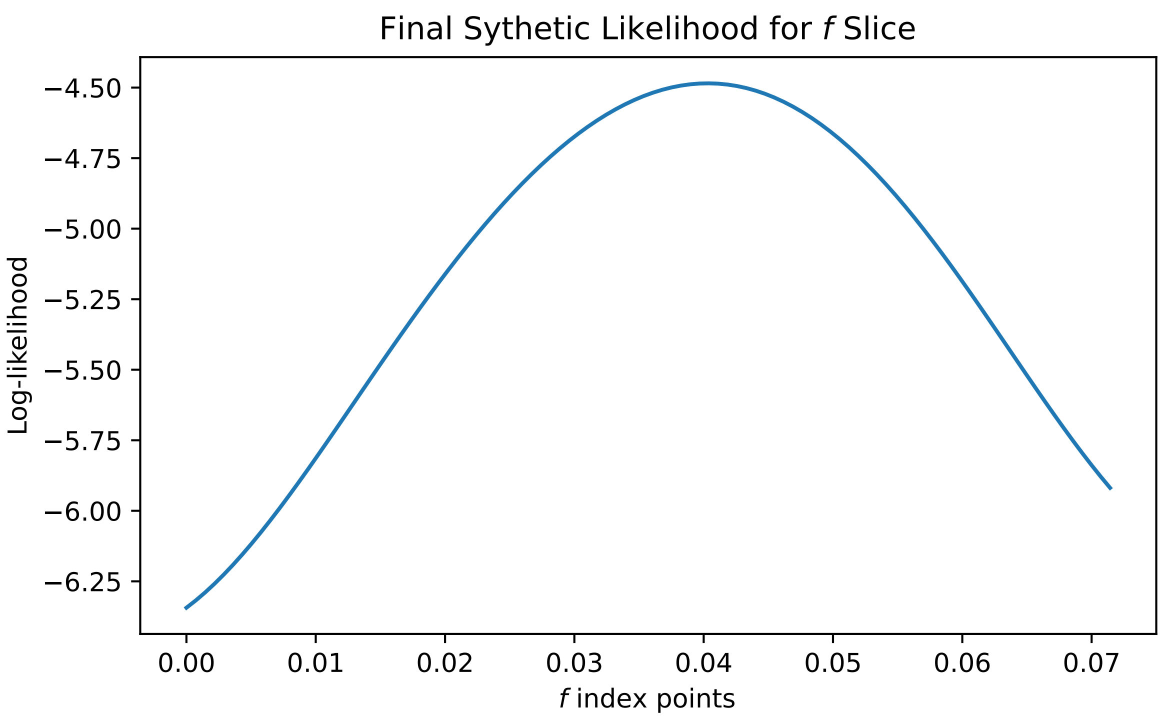


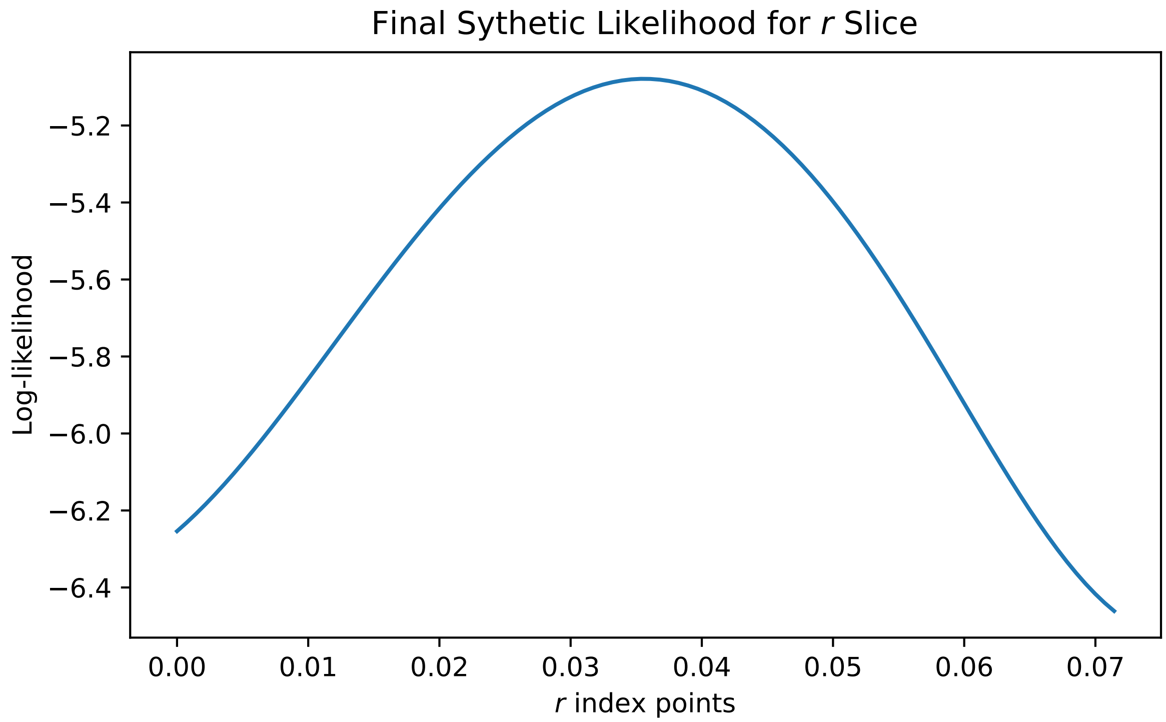
Final Sythetic Likelihood for β Slice
Log-likelihood
−5.1
−5.2
−5.3
−5.4
−5.5
−5.6
−5.7
−5.8
0.0
0.2
0.4
0.6
0.8
1.0
β index points



Final Sythetic Likelihood for γ_L Slice
Log-likelihood
−3.5
−4.0
−4.5
−5.0
−5.5
−6.0
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
γ_L index points


Final Sythetic Likelihood for λ Slice
Log-likelihood
−6
−7
−8
−9
−10
−11
0.00
0.02
0.04
0.06
0.08
0.10
λ index points


Final Sythetic Likelihood for f Slice
Log-likelihood
−4.50
−4.75
−5.00
−5.25
−5.50
−5.75
−6.00
−6.25
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
f index points




```
# print(index_vals[-600,].round(3))
print(index_vals[-400,].round(3))
print(index_vals[-200,].round(3))
print(index_vals[-80,].round(3))
print(index_vals[-40,].round(3))
print(index_vals[-20,].round(3))
print(index_vals[-8,].round(3))
print(index_vals[-4,].round(3))
print(index_vals[-2,].round(3))
print(index_vals[-1,].round(3))
```

```
[0.582 0.    0.    0.1   0.071 0.038]
[1.    0.53  0.018 0.1   0.    0.071]
[1.    0.    0.023 0.049 0.071 0.071]
[0.599 1.    0.    0.081 0.03  0.   ]
[0.    1.    0.023 0.069 0.023 0.071]
[1.    1.    0.    0.076 0.    0.   ]
[0.    0.371 0.    0.057 0.071 0.   ]
[0.    0.371 0.    0.057 0.071 0.   ]
[0.    0.371 0.    0.057 0.071 0.   ]
```