# Inference on the Champagne Model using a Gaussian Process

## TODO

- Set seed for LHC and stuff
- Change to log discrepency with custom observation variance
- Change from MLE to cross validation

## 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

import tensorflow as tf
import tensorflow_probability as tfp

tfb = tfp.bijectors
tfd = tfp.distributions
tfk = tfp.math.psd_kernels
```

```
2024-03-11 10:20:52.220966: I external/local_tsl/tsl/cuda/cudart_stub.cc:31] Could not find c
2024-03-11 10:20:52.267739: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] U
```

```
2024-03-11 10:20:52.267770: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Un
2024-03-11 10:20:52.268914: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515]
2024-03-11 10:20:52.276174: I external/local_tsl/tsl/cuda/cudart_stub.cc:31] Could not find c
2024-03-11 10:20:52.277296: I tensorflow/core/platform/cpu_feature_guard.cc:182] This Tensorf
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with
2024-03-11 10:20:53.511484: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Wa
```

### 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


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_)):
```

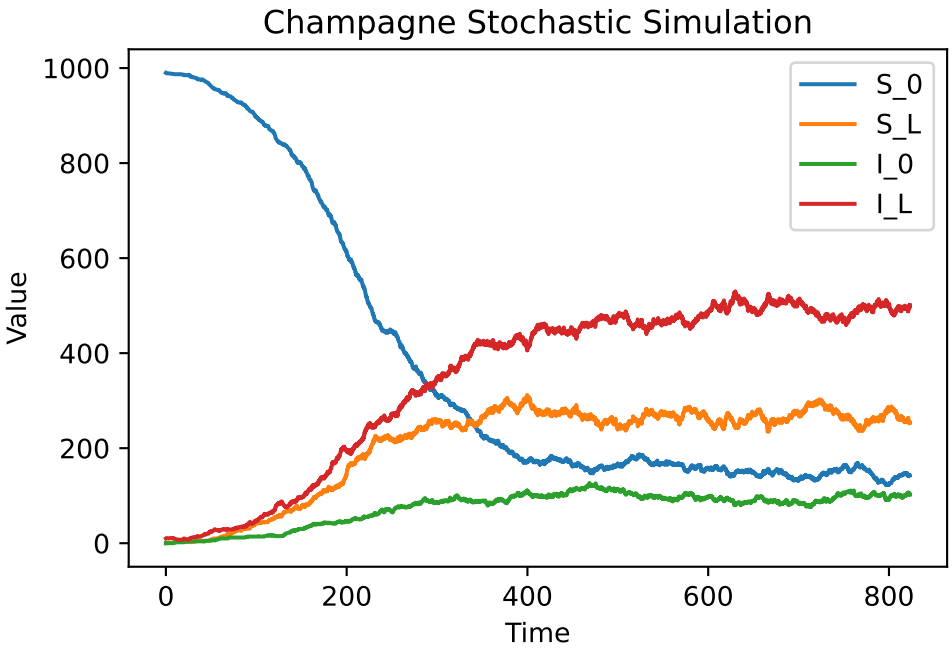
```
        return "Alpha or Beta out of bounds"
    if 0 > (gamma_L or lambda_ or f or r):
        return "Gamma, lambda, f or r out of bounds"

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

    for i in range(num_events):
        if S_0 == N:
            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
        )

        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,
```

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

    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.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 champ_prevalence(alpha_, beta_, gamma_L, lambda_, f, r):
    champ_df_ = champagne_stochastic(alpha_, beta_, gamma_L, lambda_, f, r)

    return champ_df_.iloc[-1]["I_0"] + champ_df_.iloc[-1]["I_L"]


observed_final_prevalence = champ_prevalence(
    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):
    x = champ_prevalence(alpha_, beta_, gamma_L, lambda_, f, r)
    return np.abs(x - observed_final_prevalence)
```

## Gaussian Process Regression on Final Prevalence Discrepency

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

num_samples = 50

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)
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])
```

```
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  0.246202  0.013085  0.051287  0.011657  0.004164
1  0.332324  0.812946  0.000390  0.006251  0.047737  0.018725
2  0.836050  0.343292  0.004725  0.020082  0.004604  0.007983
3  0.566773  0.075311  0.002784  0.007547  0.020959  0.022937
4  0.880603  0.964663  0.004194  0.008378  0.012502  0.009120
      alpha      beta   gamma_L    lambda         f         r
0  0.100008  0.122349  0.005550  0.047169  0.015049  0.023833
1  0.659225  0.590955  0.015422  0.009993  0.026474  0.050003
2  0.503558  0.005003  0.000207  0.024569  0.044514  0.020288
3  0.011840  0.630562  0.001543  0.016033  0.004709  0.010679
4  0.271011  0.942434  0.003873  0.020250  0.006580  0.004226
```

### 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    104.0
1    449.0
2     12.0
3      8.0
4    208.0
dtype: float64
```

### Differing Methods to Iterate Function

```
# import timeit

# def function1():
#     np.vectorize(champ_prevalence)(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_prevalence(
#             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=400.0,
            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=50.0,
            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=500.0,
            dtype=np.float64,
            name="amp_gamma_L_mean",
        )
        self.gamma_L_tp = tf.Variable(
            pv_champ_gamma_L, dtype=np.float64, name="gamma_L_tp"
        )
        self.amp_lambda_mean = tfp.util.TransformedVariable(
            bijector=constrain_positive,
            initial_value=16000.0,
            dtype=np.float64,
            name="amp_lambda_mean",
        )
        self.lambda_tp = tf.Variable(
            pv_champ_lambda, dtype=np.float64, name="lambda_tp"
        )
        self.amp_f_mean = tfp.util.TransformedVariable(
            bijector=constrain_positive,
            initial_value=15000.0,
            dtype=np.float64,
            name="amp_f_mean",
        )
        self.f_tp = tf.Variable(pv_champ_f, dtype=np.float64, name="f_tp")
```

```
        self.amp_r_mean = tfp.util.TransformedVariable(
            bijector=constrain_positive,
            initial_value=13000.0,
            dtype=np.float64,
            name="amp_r_mean",
        )
        self.r_tp = tf.Variable(pv_champ_r, dtype=np.float64, name="r_tp")
        self.bias_mean = tfp.util.TransformedVariable(
            bijector=constrain_positive,
            initial_value=50.0,
            dtype=np.float64,
            name="bias_mean",
        )

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

### 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=150.0,
    dtype=np.float64,
    name="amplitude_champ",
)

observation_noise_variance_champ = tfp.util.TransformedVariable(
    bijector=constrain_positive,
    initial_value=1000.0,
```

```
    dtype=np.float64,
    name="observation_noise_variance_champ",
)
```

```
length_scales_champ = tfp.util.TransformedVariable(
    bijector=constrain_positive,
    initial_value=[0.01, 0.01, 0.35, 0.02, 0.27, 0.2],
    dtype=np.float64,
    name="length_scales_champ",
)
```

```
kernel_champ = tfk.FeatureScaled(
    tfk.ExponentiatedQuadratic(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=quad_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=5.0106352940962555>, <tf.Vari
array([-4.60517019, -4.60517019, -1.04982212, -3.91202301, -1.30933332,
       -1.60943791])>, <tf.Variable 'observation_noise_variance_champ:0' shape=() dtype=float
```

### Train the Hyperparameters

```
@tf.function(autograph=False, jit_compile=False)
def optimize():
    with tf.GradientTape() as tape:
        loss = -champ_GP.log_prob(obs_vals)
    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
```

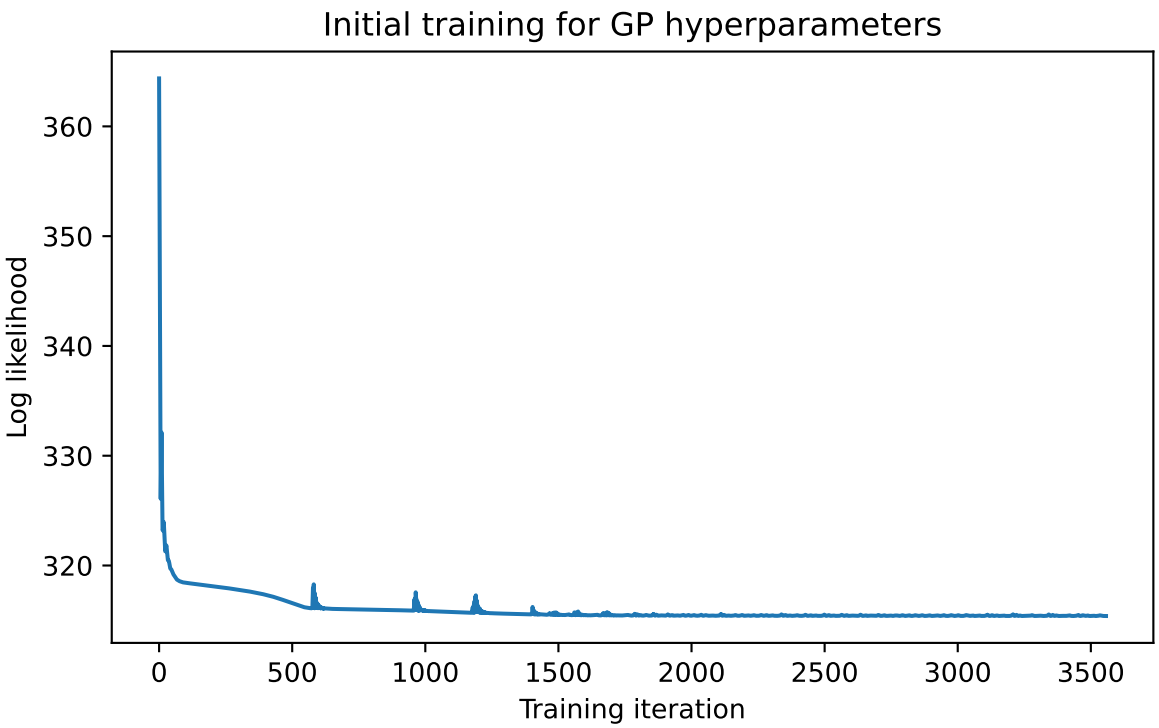
```
Hyperparameter convergence reached at iteration 3558.
```

```
print("Trained parameters:")
for var in champ_GP.trainable_variables:
    if "tp" in var.name:
        print("{} is {}".format(var.name, var.numpy().round(3)))
    else:
        print(
            "{} is {}".format(
                var.name, constrain_positive.forward(var).numpy().round(3)
            )
        )
```

```
Trained parameters:
```

```
amplitude_champ:0 is 132.705
length_scales_champ:0 is [7.900e-02 2.000e-02 2.122e+00 1.000e-03 7.661e+00 5.608e+00]
observation_noise_variance_champ:0 is 151.369
alpha_tp:0 is 0.215
amp_alpha_mean:0 is 591.499
amp_beta_mean:0 is 37.292
amp_f_mean:0 is 165497.327
amp_gamma_L_mean:0 is 510282.921
amp_lambda_mean:0 is 9542.029
amp_r_mean:0 is 47976.955
beta_tp:0 is -0.573
bias_mean:0 is 4.294
f_tp:0 is 0.029
gamma_L_tp:0 is 0.012
lambda_tp:0 is 0.07
r_tp:0 is 0.003
```

```
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.pdf")
plt.show()
```




**Fitting the GP Regression across alpha**

```
plot_samp_no = 21
gp_samp_no = 50
```

```
alpha_slice_samples = 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,
)

alpha_slice_indices_df = pd.DataFrame(alpha_slice_samples, columns=variables_names)
```

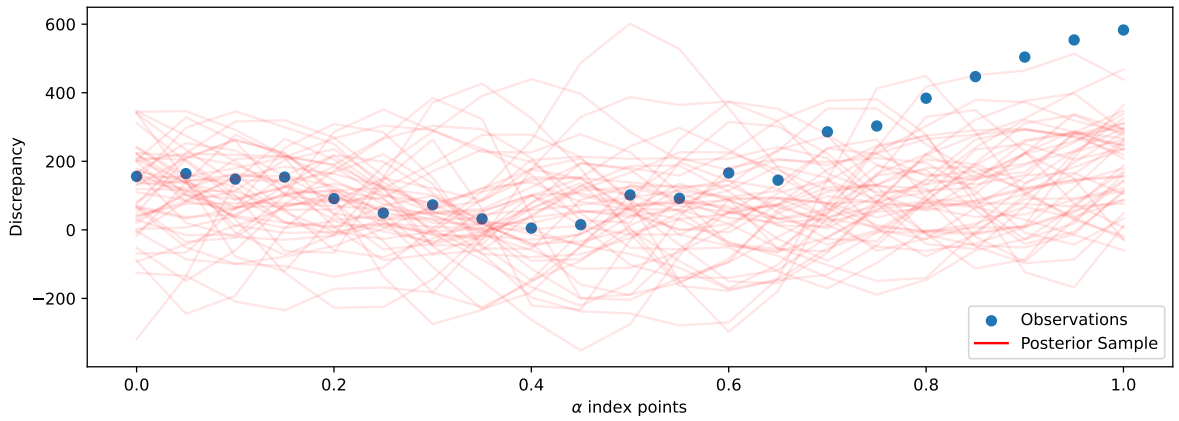
```
print(alpha_slice_indices_df.head())

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

alpha_slice_index_vals = alpha_slice_indices_df.values
```

| | alpha | beta | gamma_L | lambda | f | r |
|---|---|---|---|---|---|---|
| 0 | 0.00 | 0.4 | 0.004484 | 0.04 | 0.013889 | 0.016667 |
| 1 | 0.05 | 0.4 | 0.004484 | 0.04 | 0.013889 | 0.016667 |
| 2 | 0.10 | 0.4 | 0.004484 | 0.04 | 0.013889 | 0.016667 |
| 3 | 0.15 | 0.4 | 0.004484 | 0.04 | 0.013889 | 0.016667 |
| 4 | 0.20 | 0.4 | 0.004484 | 0.04 | 0.013889 | 0.016667 |

```
GP_seed = tfp.random.sanitize_seed(4362)

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

GP_samples = champ_GP_reg.sample(gp_samp_no, seed=GP_seed)
```

```
plt.figure(figsize=(12, 4))
plt.scatter(
    alpha_slice_index_vals[:, 0], alpha_slice_discrepencies, label="Observations"
)
for i in range(gp_samp_no):
    plt.plot(
        alpha_slice_index_vals[:, 0],
        GP_samples[i, :],
        c="r",
```

```
        alpha=0.1,
        label="Posterior Sample" if i == 0 else None,
    )
leg = plt.legend(loc="lower right")
for lh in leg.legend_handles:
    lh.set_alpha(1)
plt.xlabel(r"$\alpha$ index points")
plt.ylabel("Discrepancy")
plt.savefig("champagne_GP_images/initial_alpha_slice.pdf")
plt.show()
```




### Fitting the GP Regression across beta

```
beta_slice_samples = 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,
)

beta_slice_indices_df = pd.DataFrame(beta_slice_samples, columns=variables_names)
```

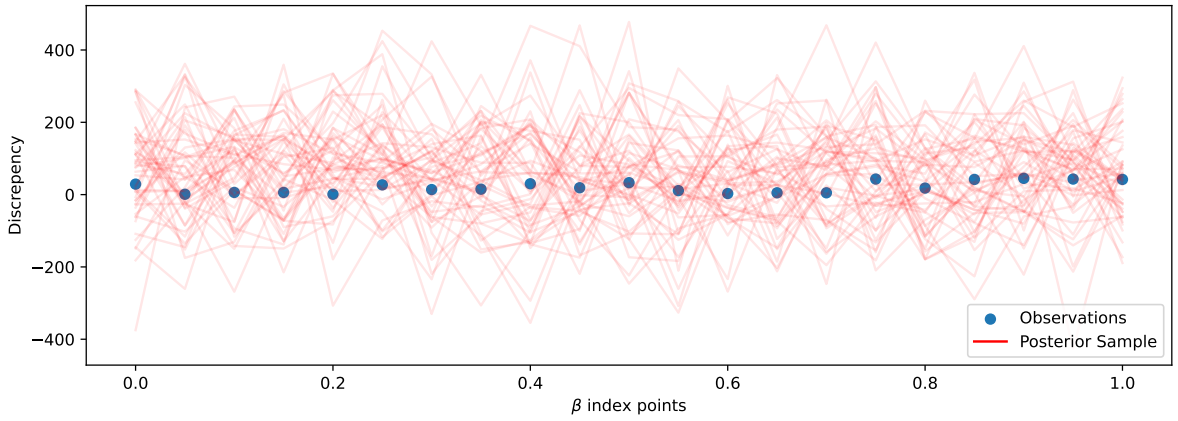
```
print(beta_slice_indices_df.head())

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

beta_slice_index_vals = beta_slice_indices_df.values
```

```
   alpha  beta   gamma_L  lambda         f         r
0    0.4  0.00  0.004484    0.04  0.013889  0.016667
1    0.4  0.05  0.004484    0.04  0.013889  0.016667
2    0.4  0.10  0.004484    0.04  0.013889  0.016667
3    0.4  0.15  0.004484    0.04  0.013889  0.016667
4    0.4  0.20  0.004484    0.04  0.013889  0.016667
```

```
champ_GP_reg = tfd.GaussianProcessRegressionModel(
    kernel=kernel_champ,
    index_points=beta_slice_index_vals,
    observation_index_points=index_vals,
    observations=obs_vals,
    observation_noise_variance=observation_noise_variance_champ,
    predictive_noise_variance=0.0,
    mean_fn=quad_mean_fn(),
)

GP_samples = champ_GP_reg.sample(gp_samp_no, seed=GP_seed)
```

```
plt.figure(figsize=(12, 4))
plt.scatter(beta_slice_index_vals[:, 1], beta_slice_discrepencies, label="Observations")
for i in range(gp_samp_no):
    plt.plot(
        beta_slice_index_vals[:, 1],
        GP_samples[i, :],
        c="r",
        alpha=0.1,
        label="Posterior Sample" if i == 0 else None,
    )
leg = plt.legend(loc="lower right")
```

```
for lh in leg.legend_handles:
    lh.set_alpha(1)
plt.xlabel(r"$\beta$ index points")
plt.ylabel("Discrepency")
plt.savefig("champagne_GP_images/initial_beta_slice.pdf")
plt.show()
```




## 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()
```

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

```
Self Kernel is 17610.725
Others Kernel is [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0.]
[[ 5.62996682e-005 -3.38945232e-213 -6.00129645e-125 ... -2.07803132e-098
  -2.56893851e-223  1.25300719e-067]
 [-3.38945232e-213  5.62996850e-005 -3.18464650e-140 ... -2.58423919e-131
   2.25082804e-086 -1.62498077e-150]
 [-6.00129645e-125 -3.18464650e-140  5.62996682e-005 ... -4.03392613e-031
   2.40998569e-103 -2.69648092e-062]
 ...
 [-2.07803132e-098 -2.58423919e-131 -4.03392613e-031 ...  5.62996682e-005
  -1.72677825e-129 -9.33693557e-036]
 [-2.56893851e-223  2.25082804e-086  2.40998569e-103 ... -1.72677825e-129
   5.62996682e-005 -1.15426621e-160]
 [ 1.25300719e-067 -1.62498077e-150 -2.69648092e-062 ... -9.33693557e-036
  -1.15426621e-160  5.62996682e-005]]
Variance function is [17610.725]
Variance function is 17610.725
```

### Loss function

```
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=0.1,
    bijector=constrain_positive,
    dtype=np.float64,
    name="next_gamma_L",
)

next_lambda = tfp.util.TransformedVariable(
    initial_value=0.1,
    bijector=constrain_positive,
    dtype=np.float64,
    name="next_lambda",
)

next_f = tfp.util.TransformedVariable(
    initial_value=0.1,
    bijector=constrain_positive,
    dtype=np.float64,
    name="next_f",
)

next_r = tfp.util.TransformedVariable(
    initial_value=0.1,
    bijector=constrain_positive,
    dtype=np.float64,
    name="next_r",
)

next_vars = [
    v.trainable_variables[0]
    for v in [next_alpha, next_beta, next_gamma_L, next_lambda, next_f, next_r]
]
```

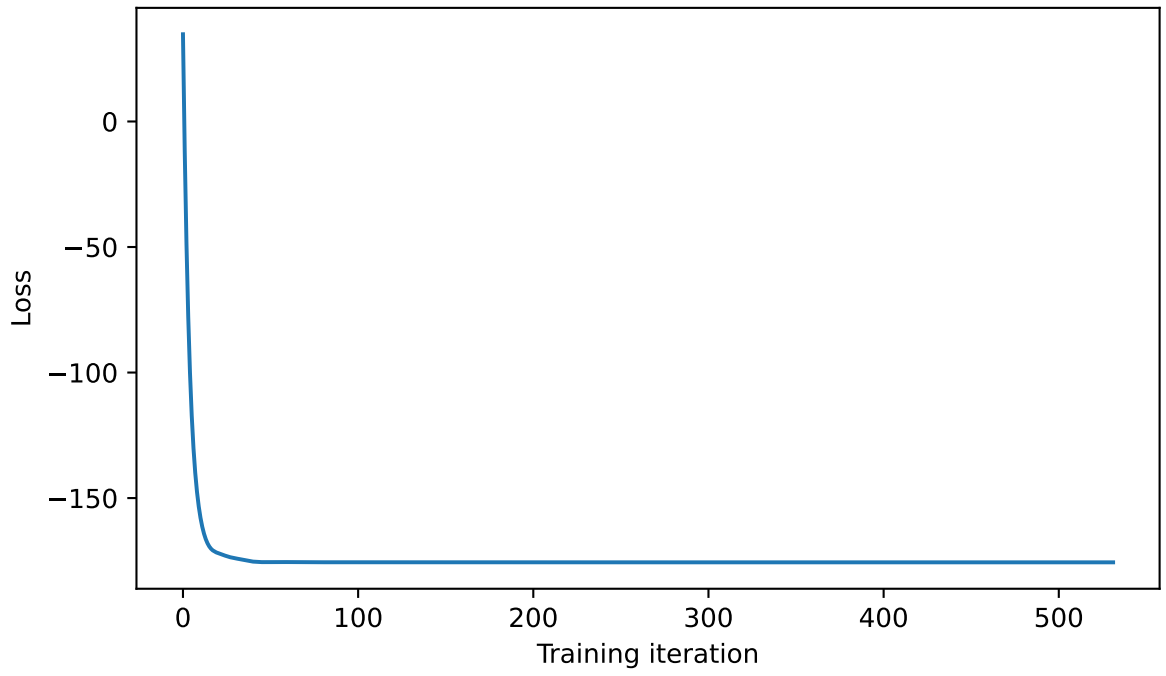
```
Adam_optim = tf.optimizers.Adam(learning_rate=0.1)


@tf.function(autograph=False, jit_compile=False)
def optimize():
    with tf.GradientTape() as tape:
        next_guess = tf.reshape(
            [
                tfb.Sigmoid().forward(next_vars[0]),
                tfb.Sigmoid().forward(next_vars[1]),
```

```
                tfb.Sigmoid().forward(next_vars[2]),
                tfb.Sigmoid().forward(next_vars[3]),
                tfb.Sigmoid().forward(next_vars[4]),
                tfb.Sigmoid().forward(next_vars[5]),
            ],
            [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_vars:
    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 532.
Trained parameters:
next_alpha:0 is 0.4
next_beta:0 is 0.4
next_gamma_L:0 is 0.005
next_lambda:0 is 0.042
next_f:0 is 0.014
next_r:0 is 0.017
```

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



```
exploration_rate = 0.1
d = 6

for t in range(200):
    eta_t = np.sqrt(np.log((t + 1) ** (d / 2 + 2) * np.pi**2 / (3 * exploration_rate)))
    print(t)
    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,
        np.array(
            [
                next_alpha.numpy(),
                next_beta.numpy(),
                next_gamma_L.numpy(),
                next_lambda.numpy(),
                next_f.numpy(),
                next_r.numpy(),
            ]
        ).reshape(1, -1),
        axis=0,
    )
    obs_vals = np.append(obs_vals, new_discrepency)

    champ_GP = tfd.GaussianProcess(
        kernel=kernel_champ,
        observation_noise_variance=observation_noise_variance_champ,
        index_points=index_vals,
        mean_fn=quad_mean_fn(),
    )

    Adam_optim = tf.optimizers.Adam()

    @tf.function(autograph=False, jit_compile=False)
    def optimize():
```

```
        with tf.GradientTape() as tape:
            loss = -champ_GP.log_prob(obs_vals)
        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

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

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

    @tf.function(autograph=False, jit_compile=False)
    def optimize():
        with tf.GradientTape() as tape:
            next_guess = tf.reshape(
                [
                    tfb.Sigmoid().forward(next_vars[0]),
```

```
                    tfb.Sigmoid().forward(next_vars[1]),
                    tfb.Sigmoid().forward(next_vars[2]),
                    tfb.Sigmoid().forward(next_vars[3]),
                    tfb.Sigmoid().forward(next_vars[4]),
                    tfb.Sigmoid().forward(next_vars[5]),
                ],
                [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)
        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
```

```
0
Hyperparameter convergence reached at iteration 5948.
Acquisition function convergence reached at iteration 96.
1
Hyperparameter convergence reached at iteration 6174.
Acquisition function convergence reached at iteration 152.
2
Hyperparameter convergence reached at iteration 6056.
Acquisition function convergence reached at iteration 5641.
```

```
3
Acquisition function convergence reached at iteration 8472.
4
Hyperparameter convergence reached at iteration 9147.
Acquisition function convergence reached at iteration 1079.
5
Hyperparameter convergence reached at iteration 8853.
Acquisition function convergence reached at iteration 2699.
6
Hyperparameter convergence reached at iteration 9027.
7
Hyperparameter convergence reached at iteration 5919.
8
Hyperparameter convergence reached at iteration 5369.
Acquisition function convergence reached at iteration 1195.
9
Hyperparameter convergence reached at iteration 5913.
10
Hyperparameter convergence reached at iteration 5643.
Acquisition function convergence reached at iteration 1202.
11
Hyperparameter convergence reached at iteration 5753.
12
Hyperparameter convergence reached at iteration 5744.
Acquisition function convergence reached at iteration 1674.
13
Hyperparameter convergence reached at iteration 4656.
14
Hyperparameter convergence reached at iteration 4983.
15
Hyperparameter convergence reached at iteration 8297.
Acquisition function convergence reached at iteration 742.
16
Hyperparameter convergence reached at iteration 4358.
Acquisition function convergence reached at iteration 194.
17
Hyperparameter convergence reached at iteration 4645.
Acquisition function convergence reached at iteration 153.
18
Hyperparameter convergence reached at iteration 5211.
Acquisition function convergence reached at iteration 155.
19
Hyperparameter convergence reached at iteration 6478.
```

```
Acquisition function convergence reached at iteration 181.
20
Acquisition function convergence reached at iteration 163.
21
Hyperparameter convergence reached at iteration 6330.
Acquisition function convergence reached at iteration 162.
22
Hyperparameter convergence reached at iteration 5370.
Acquisition function convergence reached at iteration 157.
23
Hyperparameter convergence reached at iteration 5628.
Acquisition function convergence reached at iteration 166.
24
Hyperparameter convergence reached at iteration 3538.
Acquisition function convergence reached at iteration 143.
25
Hyperparameter convergence reached at iteration 5243.
Acquisition function convergence reached at iteration 173.
26
Hyperparameter convergence reached at iteration 4335.
Acquisition function convergence reached at iteration 646.
27
Hyperparameter convergence reached at iteration 4721.
Acquisition function convergence reached at iteration 292.
28
Hyperparameter convergence reached at iteration 4606.
Acquisition function convergence reached at iteration 144.
29
Hyperparameter convergence reached at iteration 4709.
Acquisition function convergence reached at iteration 187.
30
Hyperparameter convergence reached at iteration 4753.
31
Hyperparameter convergence reached at iteration 4387.
Acquisition function convergence reached at iteration 678.
32
Hyperparameter convergence reached at iteration 4264.
Acquisition function convergence reached at iteration 182.
33
Hyperparameter convergence reached at iteration 4268.
Acquisition function convergence reached at iteration 164.
34
Hyperparameter convergence reached at iteration 3626.
```

```
Acquisition function convergence reached at iteration 161.
35
Hyperparameter convergence reached at iteration 3988.
Acquisition function convergence reached at iteration 171.
36
Hyperparameter convergence reached at iteration 3937.
Acquisition function convergence reached at iteration 1940.
37
Hyperparameter convergence reached at iteration 4067.
Acquisition function convergence reached at iteration 161.
38
Hyperparameter convergence reached at iteration 3976.
Acquisition function convergence reached at iteration 493.
39
Hyperparameter convergence reached at iteration 3951.
Acquisition function convergence reached at iteration 189.
40
Hyperparameter convergence reached at iteration 4281.
Acquisition function convergence reached at iteration 1300.
41
Hyperparameter convergence reached at iteration 4371.
Acquisition function convergence reached at iteration 170.
42
Hyperparameter convergence reached at iteration 4344.
Acquisition function convergence reached at iteration 1291.
43
Hyperparameter convergence reached at iteration 4338.
Acquisition function convergence reached at iteration 167.
44
Hyperparameter convergence reached at iteration 6921.
Acquisition function convergence reached at iteration 702.
45
Hyperparameter convergence reached at iteration 4742.
Acquisition function convergence reached at iteration 4979.
46
Hyperparameter convergence reached at iteration 4317.
Acquisition function convergence reached at iteration 855.
47
Hyperparameter convergence reached at iteration 5378.
48
Hyperparameter convergence reached at iteration 6022.
Acquisition function convergence reached at iteration 171.
49
```

```
Hyperparameter convergence reached at iteration 5193.
Acquisition function convergence reached at iteration 160.
50
Hyperparameter convergence reached at iteration 4702.
Acquisition function convergence reached at iteration 166.
51
Hyperparameter convergence reached at iteration 7344.
Acquisition function convergence reached at iteration 180.
52
Hyperparameter convergence reached at iteration 9301.
Acquisition function convergence reached at iteration 173.
53
Hyperparameter convergence reached at iteration 6667.
Acquisition function convergence reached at iteration 447.
54
Hyperparameter convergence reached at iteration 5263.
Acquisition function convergence reached at iteration 131.
55
Hyperparameter convergence reached at iteration 6759.
Acquisition function convergence reached at iteration 167.
56
Hyperparameter convergence reached at iteration 6514.
Acquisition function convergence reached at iteration 185.
57
Hyperparameter convergence reached at iteration 5037.
Acquisition function convergence reached at iteration 963.
58
Hyperparameter convergence reached at iteration 5187.
Acquisition function convergence reached at iteration 181.
59
Hyperparameter convergence reached at iteration 9426.
Acquisition function convergence reached at iteration 162.
60
Hyperparameter convergence reached at iteration 6138.
Acquisition function convergence reached at iteration 799.
61
Hyperparameter convergence reached at iteration 6918.
Acquisition function convergence reached at iteration 166.
62
Hyperparameter convergence reached at iteration 5146.
Acquisition function convergence reached at iteration 180.
63
Hyperparameter convergence reached at iteration 6426.
```

```
Acquisition function convergence reached at iteration 188.
64
Hyperparameter convergence reached at iteration 9292.
Acquisition function convergence reached at iteration 246.
65
Hyperparameter convergence reached at iteration 6312.
Acquisition function convergence reached at iteration 178.
66
Hyperparameter convergence reached at iteration 6462.
Acquisition function convergence reached at iteration 187.
67
Hyperparameter convergence reached at iteration 6390.
Acquisition function convergence reached at iteration 158.
68
Hyperparameter convergence reached at iteration 4967.
Acquisition function convergence reached at iteration 198.
69
Hyperparameter convergence reached at iteration 3963.
Acquisition function convergence reached at iteration 179.
70
Hyperparameter convergence reached at iteration 4722.
Acquisition function convergence reached at iteration 195.
71
Hyperparameter convergence reached at iteration 8242.
Acquisition function convergence reached at iteration 184.
72
Hyperparameter convergence reached at iteration 7901.
Acquisition function convergence reached at iteration 193.
73
Hyperparameter convergence reached at iteration 8941.
Acquisition function convergence reached at iteration 157.
74
Hyperparameter convergence reached at iteration 8181.
Acquisition function convergence reached at iteration 170.
75
Hyperparameter convergence reached at iteration 4096.
Acquisition function convergence reached at iteration 643.
76
Hyperparameter convergence reached at iteration 9080.
Acquisition function convergence reached at iteration 181.
77
Hyperparameter convergence reached at iteration 5801.
Acquisition function convergence reached at iteration 204.
```

```
78
Hyperparameter convergence reached at iteration 8994.
Acquisition function convergence reached at iteration 192.
79
Hyperparameter convergence reached at iteration 5736.
Acquisition function convergence reached at iteration 107.
80
Hyperparameter convergence reached at iteration 7738.
Acquisition function convergence reached at iteration 170.
81
Hyperparameter convergence reached at iteration 7559.
Acquisition function convergence reached at iteration 177.
82
Hyperparameter convergence reached at iteration 7642.
Acquisition function convergence reached at iteration 140.
83
Hyperparameter convergence reached at iteration 8864.
Acquisition function convergence reached at iteration 176.
84
Hyperparameter convergence reached at iteration 8966.
Acquisition function convergence reached at iteration 182.
85
Hyperparameter convergence reached at iteration 8250.
Acquisition function convergence reached at iteration 150.
86
Hyperparameter convergence reached at iteration 4727.
Acquisition function convergence reached at iteration 1049.
87
Hyperparameter convergence reached at iteration 8462.
Acquisition function convergence reached at iteration 159.
88
Hyperparameter convergence reached at iteration 6545.
Acquisition function convergence reached at iteration 170.
89
Hyperparameter convergence reached at iteration 8465.
Acquisition function convergence reached at iteration 184.
90
Hyperparameter convergence reached at iteration 8905.
Acquisition function convergence reached at iteration 173.
91
Hyperparameter convergence reached at iteration 4770.
Acquisition function convergence reached at iteration 134.
92
```

```
Hyperparameter convergence reached at iteration 6329.
Acquisition function convergence reached at iteration 1121.
93
Hyperparameter convergence reached at iteration 5892.
Acquisition function convergence reached at iteration 274.
94
Hyperparameter convergence reached at iteration 4270.
Acquisition function convergence reached at iteration 300.
95
Hyperparameter convergence reached at iteration 6543.
Acquisition function convergence reached at iteration 171.
96
Hyperparameter convergence reached at iteration 9224.
Acquisition function convergence reached at iteration 160.
97
Hyperparameter convergence reached at iteration 3817.
Acquisition function convergence reached at iteration 178.
98
Hyperparameter convergence reached at iteration 8889.
Acquisition function convergence reached at iteration 167.
99
Hyperparameter convergence reached at iteration 4659.
Acquisition function convergence reached at iteration 183.
100
Hyperparameter convergence reached at iteration 8738.
Acquisition function convergence reached at iteration 150.
101
Hyperparameter convergence reached at iteration 9010.
Acquisition function convergence reached at iteration 171.
102
Hyperparameter convergence reached at iteration 6681.
Acquisition function convergence reached at iteration 167.
103
Hyperparameter convergence reached at iteration 9220.
Acquisition function convergence reached at iteration 163.
104
Hyperparameter convergence reached at iteration 9391.
Acquisition function convergence reached at iteration 178.
105
Hyperparameter convergence reached at iteration 6638.
Acquisition function convergence reached at iteration 128.
106
Hyperparameter convergence reached at iteration 7392.
```

```
Acquisition function convergence reached at iteration 1090.
107
Hyperparameter convergence reached at iteration 9309.
Acquisition function convergence reached at iteration 173.
108
Hyperparameter convergence reached at iteration 6878.
Acquisition function convergence reached at iteration 176.
109
Hyperparameter convergence reached at iteration 8916.
Acquisition function convergence reached at iteration 184.
110
Hyperparameter convergence reached at iteration 9425.
Acquisition function convergence reached at iteration 854.
111
Hyperparameter convergence reached at iteration 5821.
Acquisition function convergence reached at iteration 341.
112
Hyperparameter convergence reached at iteration 9309.
Acquisition function convergence reached at iteration 159.
113
Hyperparameter convergence reached at iteration 8930.
Acquisition function convergence reached at iteration 165.
114
Hyperparameter convergence reached at iteration 6525.
Acquisition function convergence reached at iteration 182.
115
Hyperparameter convergence reached at iteration 4125.
Acquisition function convergence reached at iteration 175.
116
Hyperparameter convergence reached at iteration 6921.
Acquisition function convergence reached at iteration 162.
117
Hyperparameter convergence reached at iteration 6544.
Acquisition function convergence reached at iteration 200.
118
Hyperparameter convergence reached at iteration 9144.
Acquisition function convergence reached at iteration 172.
119
Hyperparameter convergence reached at iteration 5744.
Acquisition function convergence reached at iteration 211.
120
Hyperparameter convergence reached at iteration 8898.
Acquisition function convergence reached at iteration 171.
```

```
121
Hyperparameter convergence reached at iteration 8297.
Acquisition function convergence reached at iteration 146.
122
Hyperparameter convergence reached at iteration 9395.
Acquisition function convergence reached at iteration 176.
123
Hyperparameter convergence reached at iteration 8696.
Acquisition function convergence reached at iteration 181.
124
Hyperparameter convergence reached at iteration 9000.
Acquisition function convergence reached at iteration 148.
125
Hyperparameter convergence reached at iteration 9375.
Acquisition function convergence reached at iteration 4983.
126
Hyperparameter convergence reached at iteration 7979.
Acquisition function convergence reached at iteration 321.
127
Hyperparameter convergence reached at iteration 5664.
Acquisition function convergence reached at iteration 1337.
128
Hyperparameter convergence reached at iteration 3767.
Acquisition function convergence reached at iteration 291.
129
Hyperparameter convergence reached at iteration 8737.
Acquisition function convergence reached at iteration 379.
130
Hyperparameter convergence reached at iteration 8864.
Acquisition function convergence reached at iteration 218.
131
Hyperparameter convergence reached at iteration 9315.
Acquisition function convergence reached at iteration 138.
132
Hyperparameter convergence reached at iteration 7466.
Acquisition function convergence reached at iteration 303.
133
Hyperparameter convergence reached at iteration 9311.
Acquisition function convergence reached at iteration 151.
134
Hyperparameter convergence reached at iteration 5814.
Acquisition function convergence reached at iteration 143.
135
```

```
Hyperparameter convergence reached at iteration 9062.
Acquisition function convergence reached at iteration 193.
136
Acquisition function convergence reached at iteration 418.
137
Hyperparameter convergence reached at iteration 6281.
Acquisition function convergence reached at iteration 163.
138
Hyperparameter convergence reached at iteration 9534.
139
140
Hyperparameter convergence reached at iteration 7219.
Acquisition function convergence reached at iteration 161.
141
Hyperparameter convergence reached at iteration 2738.
142
Hyperparameter convergence reached at iteration 4359.
Acquisition function convergence reached at iteration 177.
143
Hyperparameter convergence reached at iteration 5749.
Acquisition function convergence reached at iteration 160.
144
Hyperparameter convergence reached at iteration 6069.
Acquisition function convergence reached at iteration 159.
145
Acquisition function convergence reached at iteration 175.
146
Hyperparameter convergence reached at iteration 7560.
Acquisition function convergence reached at iteration 201.
147
Hyperparameter convergence reached at iteration 8103.
148
Acquisition function convergence reached at iteration 160.
149
Hyperparameter convergence reached at iteration 8220.
Acquisition function convergence reached at iteration 184.
150
Acquisition function convergence reached at iteration 203.
151
Hyperparameter convergence reached at iteration 5800.
152
Hyperparameter convergence reached at iteration 7616.
Acquisition function convergence reached at iteration 181.
```

```
153
Acquisition function convergence reached at iteration 164.
154
Hyperparameter convergence reached at iteration 7950.
Acquisition function convergence reached at iteration 189.
155
Hyperparameter convergence reached at iteration 7986.
Acquisition function convergence reached at iteration 208.
156
Hyperparameter convergence reached at iteration 6898.
Acquisition function convergence reached at iteration 327.
157
Hyperparameter convergence reached at iteration 9640.
Acquisition function convergence reached at iteration 3148.
158
Hyperparameter convergence reached at iteration 9241.
159
Acquisition function convergence reached at iteration 165.
160
Hyperparameter convergence reached at iteration 5360.
Acquisition function convergence reached at iteration 166.
161
Hyperparameter convergence reached at iteration 5933.
Acquisition function convergence reached at iteration 181.
162
Hyperparameter convergence reached at iteration 9894.
Acquisition function convergence reached at iteration 158.
163
Hyperparameter convergence reached at iteration 9879.
Acquisition function convergence reached at iteration 178.
164
Hyperparameter convergence reached at iteration 7169.
Acquisition function convergence reached at iteration 178.
165
Hyperparameter convergence reached at iteration 9519.
Acquisition function convergence reached at iteration 172.
166
Hyperparameter convergence reached at iteration 5806.
Acquisition function convergence reached at iteration 195.
167
Hyperparameter convergence reached at iteration 5278.
Acquisition function convergence reached at iteration 164.
168
```

```
Hyperparameter convergence reached at iteration 9986.
Acquisition function convergence reached at iteration 154.
169
Hyperparameter convergence reached at iteration 5989.
Acquisition function convergence reached at iteration 179.
170
Hyperparameter convergence reached at iteration 9397.
Acquisition function convergence reached at iteration 192.
171
Acquisition function convergence reached at iteration 176.
172
Hyperparameter convergence reached at iteration 6662.
Acquisition function convergence reached at iteration 164.
173
Acquisition function convergence reached at iteration 192.
174
Hyperparameter convergence reached at iteration 7938.
175
Acquisition function convergence reached at iteration 194.
176
Hyperparameter convergence reached at iteration 6163.
Acquisition function convergence reached at iteration 194.
177
Hyperparameter convergence reached at iteration 6944.
Acquisition function convergence reached at iteration 161.
178
Hyperparameter convergence reached at iteration 9344.
Acquisition function convergence reached at iteration 190.
179
Hyperparameter convergence reached at iteration 7985.
Acquisition function convergence reached at iteration 173.
180
Hyperparameter convergence reached at iteration 9573.
181
Acquisition function convergence reached at iteration 207.
182
Acquisition function convergence reached at iteration 188.
183
Hyperparameter convergence reached at iteration 7346.
Acquisition function convergence reached at iteration 603.
184
Hyperparameter convergence reached at iteration 7439.
Acquisition function convergence reached at iteration 105.
```

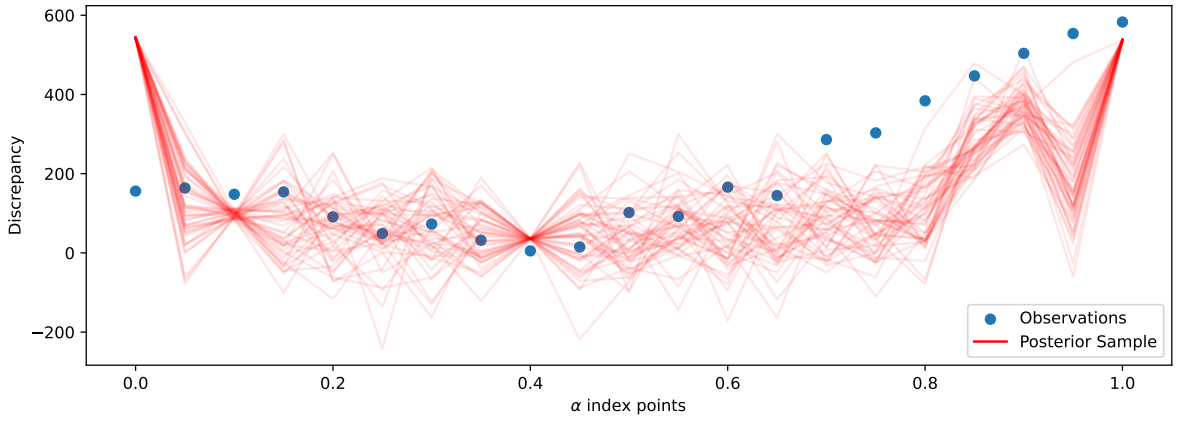
```
185
Hyperparameter convergence reached at iteration 9954.
Acquisition function convergence reached at iteration 151.
186
Hyperparameter convergence reached at iteration 9871.
Acquisition function convergence reached at iteration 679.
187
Hyperparameter convergence reached at iteration 8365.
Acquisition function convergence reached at iteration 437.
188
Hyperparameter convergence reached at iteration 7039.
Acquisition function convergence reached at iteration 189.
189
Acquisition function convergence reached at iteration 1062.
190
Acquisition function convergence reached at iteration 236.
191
Acquisition function convergence reached at iteration 152.
192
Acquisition function convergence reached at iteration 235.
193
Acquisition function convergence reached at iteration 579.
194
Hyperparameter convergence reached at iteration 8320.
Acquisition function convergence reached at iteration 237.
195
Hyperparameter convergence reached at iteration 9433.
Acquisition function convergence reached at iteration 2865.
196
Acquisition function convergence reached at iteration 368.
197
Acquisition function convergence reached at iteration 730.
198
Hyperparameter convergence reached at iteration 7322.
Acquisition function convergence reached at iteration 266.
199
Acquisition function convergence reached at iteration 282.
```

### Fitting the GP Regression across alpha

```
plot_samp_no = 21
gp_samp_no = 50
```

```
GP_seed = tfp.random.sanitize_seed(4362)

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

GP_samples = champ_GP_reg.sample(gp_samp_no, seed=GP_seed)
```

```
plt.figure(figsize=(12, 4))
plt.scatter(
    alpha_slice_index_vals[:, 0], alpha_slice_discrepencies, label="Observations"
)
for i in range(gp_samp_no):
    plt.plot(
        alpha_slice_index_vals[:, 0],
        GP_samples[i, :],
        c="r",
        alpha=0.1,
        label="Posterior Sample" if i == 0 else None,
    )
leg = plt.legend(loc="lower right")
for lh in leg.legend_handles:
    lh.set_alpha(1)
plt.xlabel(r"$\alpha$ index points")
plt.ylabel("Discrepancy")
plt.savefig("champagne_GP_images/new_alpha_slice.pdf")
plt.show()
```




### Fitting the GP Regression across beta

```
champ_GP_reg = tfd.GaussianProcessRegressionModel(
    kernel=kernel_champ,
    index_points=beta_slice_index_vals,
    observation_index_points=index_vals,
    observations=obs_vals,
    observation_noise_variance=observation_noise_variance_champ,
    predictive_noise_variance=0.0,
    mean_fn=quad_mean_fn(),
)

GP_samples = champ_GP_reg.sample(gp_samp_no, seed=GP_seed)
```

```
2024-03-11 13:04:06.549378: W tensorflow/core/kernels/linalg/cholesky_op.cc:56] Cholesky dec
```

```
plt.figure(figsize=(12, 4))
plt.scatter(beta_slice_index_vals[:, 1], beta_slice_discrepencies, label="Observations")
for i in range(gp_samp_no):
    plt.plot(
        beta_slice_index_vals[:, 1],
        GP_samples[i, :],
        c="r",
        alpha=0.1,
        label="Posterior Sample" if i == 0 else None,
    )
leg = plt.legend(loc="lower right")
```

```
for lh in leg.legend_handles:
    lh.set_alpha(1)
plt.xlabel(r"$\beta$ index points")
plt.ylabel("Discrepency")
plt.savefig("champagne_GP_images/new_beta_slice.pdf")
plt.show()
```