Akshata Patil

AA222: Engineering Design Optimization

Date of Submission: 05/26/2023

<u>Project 3:</u>

Part 1: Noiseless Gaussian Process Fitting:

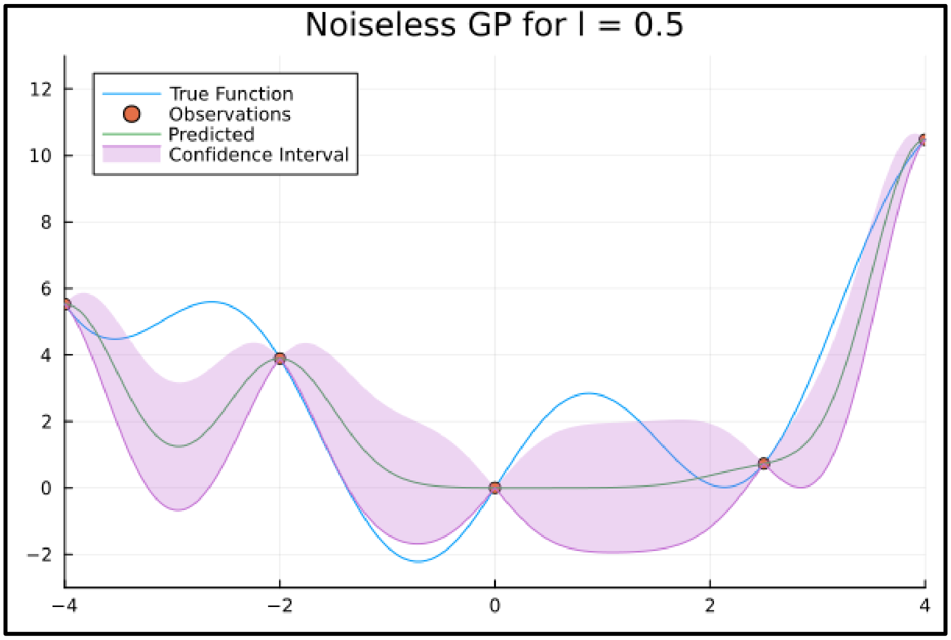

Figure 1: Noiseless Gaussian Process Fitting for characteristic length = 0.5

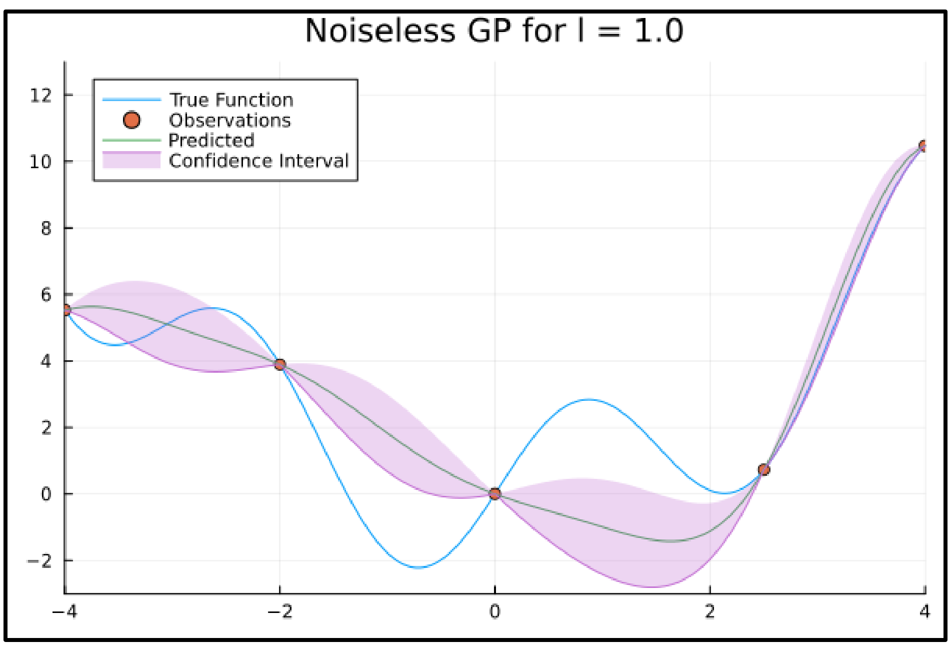

Figure 2: Noiseless Gaussian Process Fitting for characteristic length = 1.0

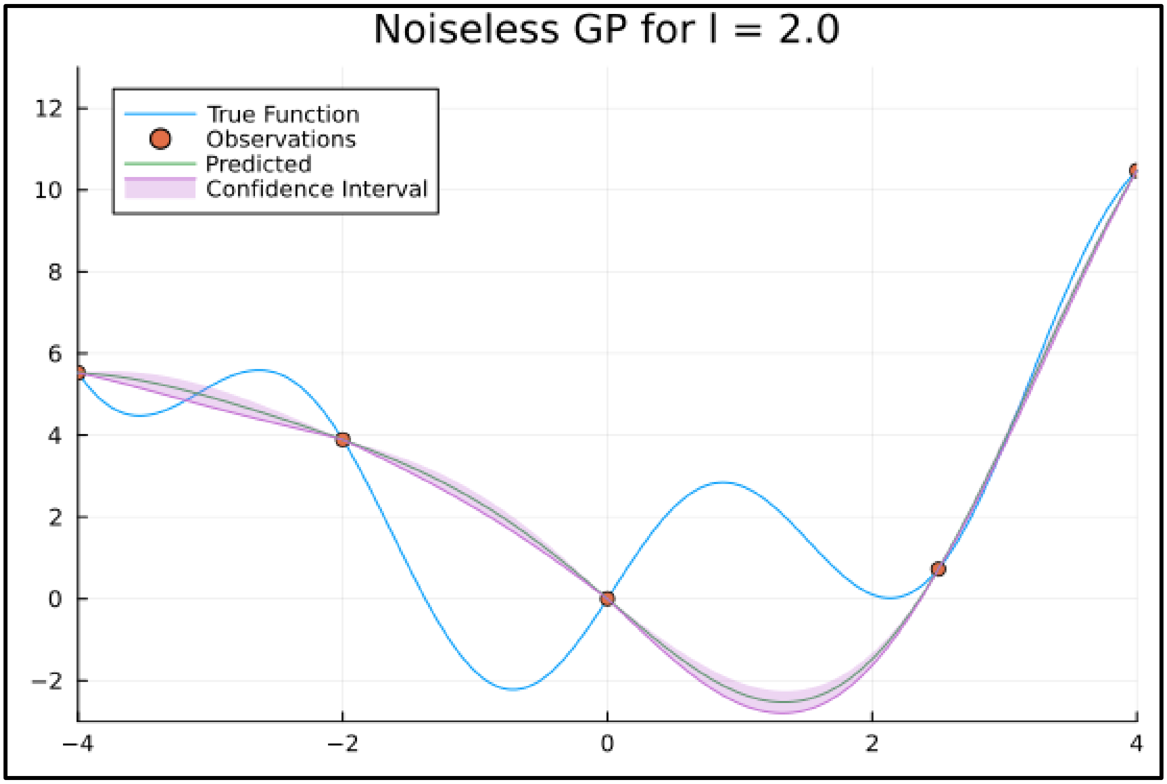

Figure 3: Noiseless Gaussian Process Fitting for characteristic length = 2.0

Part 2: Noisy Gaussian Fitting:

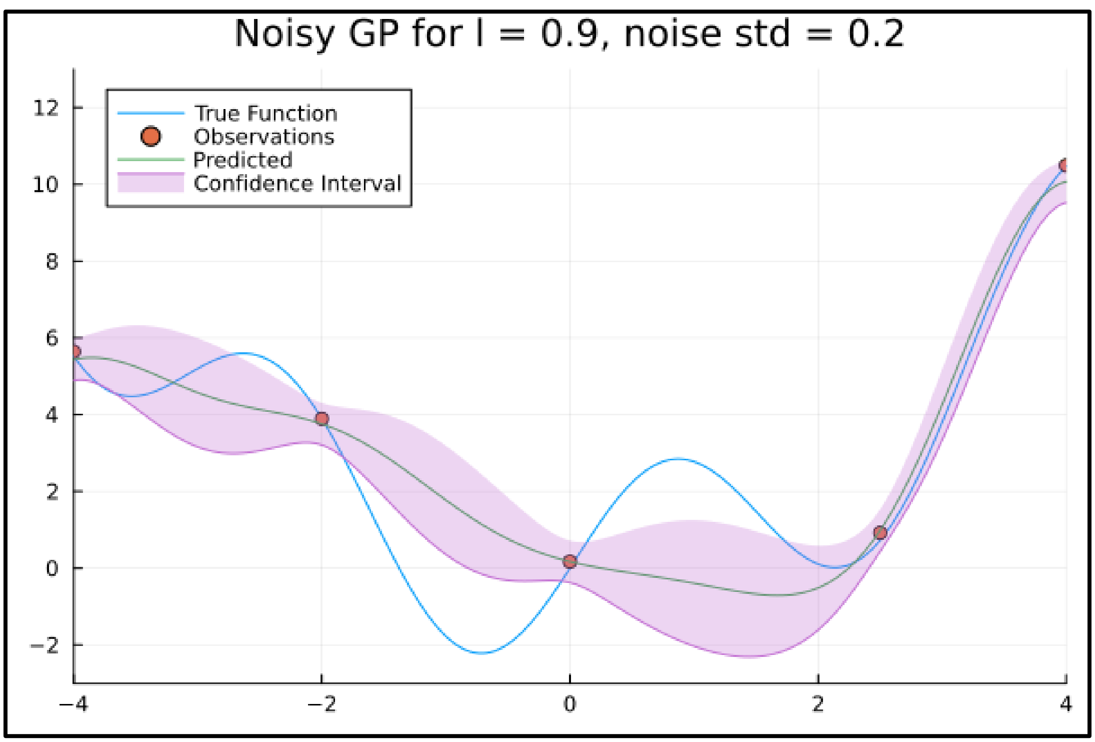

Figure 4: Noisy Gaussian Process Fitting for characteristic length = 0.9

Part 3: Exploration Strategies

Method	Design Point x	Predicted y
Prediction Based	1.67	-0.7080618717419538
Error Based	1.24	-0.5390502526770543
LCB Based	1.44	-0.6477921404833147
	-0.48	0.34463504852758514

Table 1: Function Predictions per different strategies

Comparison:

Various methods can guide an optimization process towards better design points, as demonstrated by the function predictions shown in Table 1. The Prediction-based exploration method selects the design point with the minimum value of the predicted surrogate function. This strategy aims to identify regions of the design space that are likely to have low values of the true function.

In contrast, the Error-based exploration method focuses on increasing confidence in the true function by sampling at a point where the standard deviation between the true function and the predicted function is maximized. By targeting areas with high uncertainty, this strategy aims to gain more information about the true behavior of the function.

The Lower Confidence Bound (LCB) based exploration method selects a design point that minimizes the Lower Confidence Bound of the objective function. This strategy strikes a balance between exploration and exploitation by seeking regions with both low predicted values and high uncertainty.

By employing these different exploration strategies, optimization algorithms can effectively navigate the design space and discover better solutions.

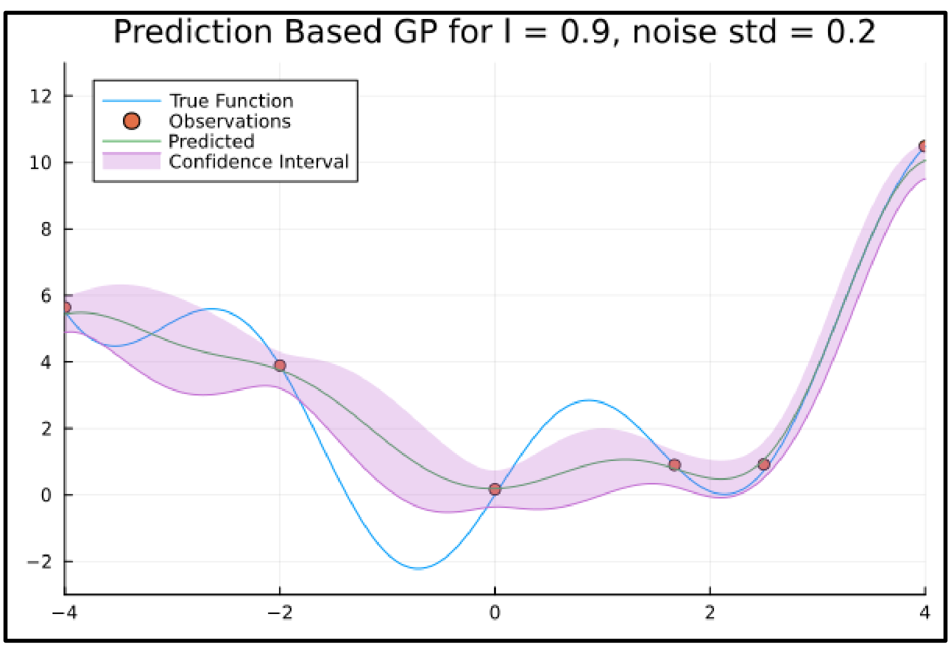

Figure 5: Updated Gaussian Plot from Prediction Based Exploration

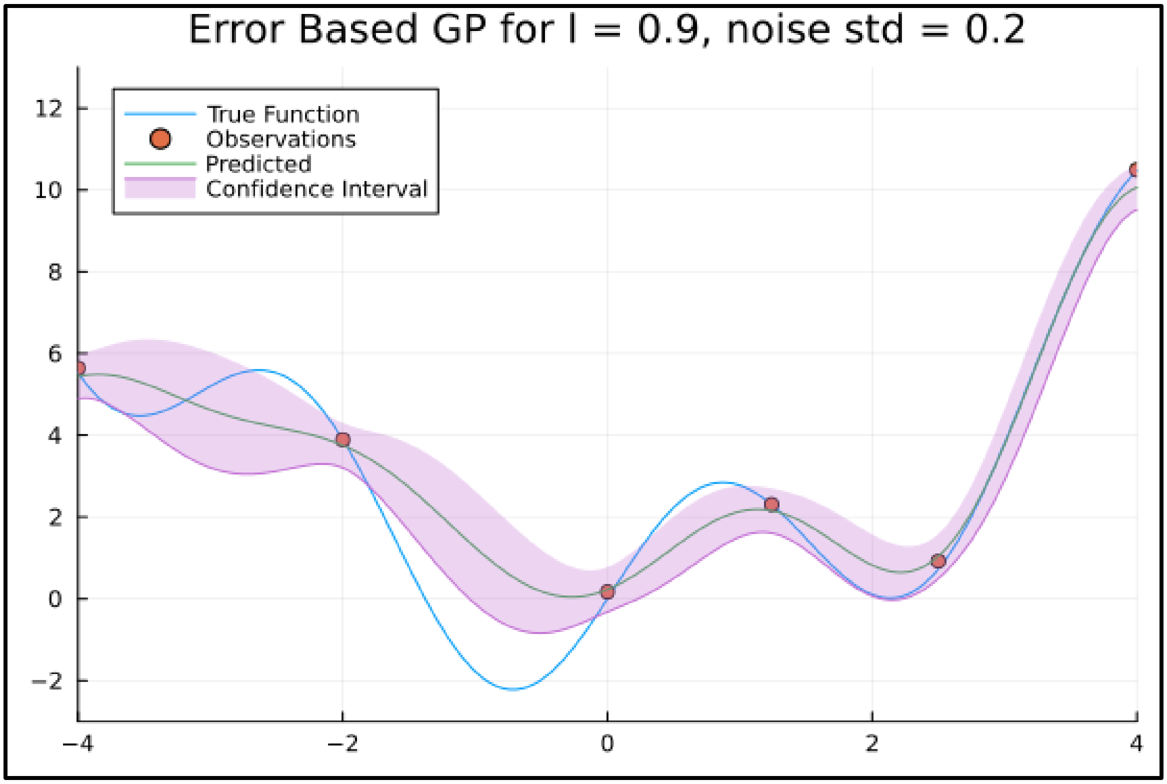

Figure 6: Updated Gaussian Plot from Error Based Exploration

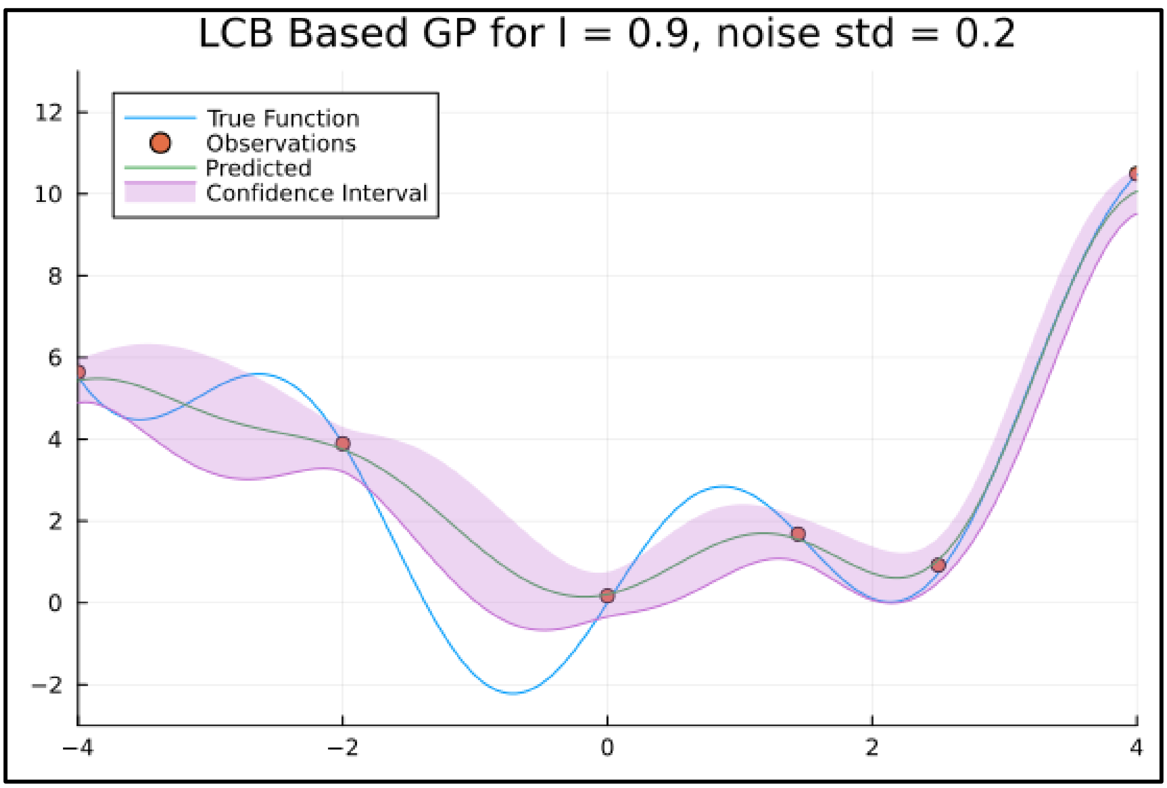

Figure 7: Updated Gaussian Plot from LCB Based Exploration

Code:

project3.jl

```julia
using GaussianProcesses, Plots, Random

# Defining the function:
f(x) = ((x^2)+(5*sin(2*x))) / 2
x_obs = [-4.0, -2.0, 0.0, 2.5, 4.0]
# Known Evals:
y_obs = f.(x_obs)
# True Evals:
x_all = collect(-4.0:0.01:4.0)
y_all = f.(x_all)
# characteristic lengths:
ls_values = [0.5, 1.0, 2.0]
# Gaussian Process:
for ls in ls_values
    m = MeanZero()
    kernel = SE(log(ls), 0.0)
    gp = GP(x_obs, y_obs, m, kernel, -Inf)
    μ, σ² = predict_y(gp, x_all)
    std95 = 1.96 * sqrt.(σ²)

    # Plotting:
    plt = plot(x_all, y_all, label = "True Function")
    scatter!(x_obs, y_obs, label = "Observations")
    plot!(x_all, μ, label = "Predicted")
    plot!(x_all, μ - std95, fillrange = μ + std95, fillalpha = 0.3, label = "Confidence Interval")
    # Setting axes limits:
    xlims!(-4, 4)
    ylims!(-3, 13)
    # Title:
    title!("Noiseless GP for l = $ls")

    # Saving the Plot:
    savefig("GaussianProcesses_l_$ls.png")
end

# Noise:
f(x) = ((x^2)+(5*sin(2*x))) / 2
# known Evals
x_obs = [-4.0, -2.0, 0.0, 2.5, 4.0]
y_obs = [5.64, 3.89, 0.17, 0.92, 10.49]
# True Evals:
x_all = collect(-4.0:0.01:4.0)
y_all = f.(x_all)
ls = 0.9
noise_std = 0.2  # Standard deviation of the additive noise
# Gaussian Process:
m = MeanZero()
kernel = SE(log(ls), 0.0)
gp = GP(x_obs, y_obs, m, kernel, log(noise_std))
μ, σ² = predict_y(gp, x_all)
std95 = 1.96 * sqrt.(σ²)
# Plotting:
plt = plot(x_all, y_all, label = "True Function")
scatter!(x_obs, y_obs, label = "Observations")
plot!(x_all, μ, label = "Predicted")
plot!(x_all, μ - std95, fillrange = μ + std95, fillalpha = 0.3, label = "Confidence Interval")
# Setting axes limits:
xlims!(-4, 4)
ylims!(-3, 13)
# Title:
title!("Noisy GP for l = $ls, noise std = $noise_std")
# Saving the Plot:
savefig("NoisyGaussianProcesses_l_$ls.png")

# Exploration Strategies:

# Prediction Based Exploration:
x_pred_index = argmin(μ)
x_pred = x_all[x_pred_index]
f_pred = μ[x_pred_index]
f_model_pred = f.(x_pred)
println("Next point for evaluation, Prediction Based: x = $x_pred, f(x) = $f_pred")
# Updating Samples:
x_obs_pred = copy(x_obs)
```

```
push!(x_obs_pred, x_pred)
y_obs_pred = copy(y_obs)
push!(y_obs_pred, f_model_pred)
# Refitting the Gaussian Model:
m = MeanZero()
kernel = SE(log(ls), 0.0)
gp = GP(x_obs_pred, y_obs_pred, m, kernel, log(noise_std))
μ_pred, σ²_pred = predict_y(gp, x_all)
std95 = 1.96 * sqrt.(σ²_pred)
# Plotting:
plt = plot(x_all, y_all, label = "True Function")
scatter!(x_obs_pred, y_obs_pred, label = "Observations")
plot!(x_all, μ_pred, label = "Predicted")
plot!(x_all, μ_pred - std95 , fillrange = μ_pred + std95, fillalpha = 0.3, label = "Confidence Interval")
# Setting axes limits:std95
xlims!(-4, 4)
ylims!(-3, 13)
# Title:
title!("Prediction Based GP for l = $ls, noise std = $noise_std")
# Saving the Plot:
savefig("PredictionBased_l_$ls.png")

# # Error Based Exploration:
x_err_index = argmax(sqrt.(σ²))
x_err = x_all[x_err_index]
f_err = μ[x_err_index]
f_model_err = f.(x_err)
println("Next point for evaluation, Error Based: x = $x_err, f(x) = $f_err")
# Updating Samples:
x_obs_err = copy(x_obs)
push!(x_obs_err, x_err)
y_obs_err = copy(y_obs)
push!(y_obs_err, f_model_err)
# Refitting the Gaussian Model:
m = MeanZero()
kernel = SE(log(ls), 0.0)
gp = GP(x_obs_err, y_obs_err, m, kernel, log(noise_std))
μ_err, σ²_err = predict_y(gp, x_all)
std95 = 1.96 * sqrt.(σ²_err)
# Plotting:
plt = plot(x_all, y_all, label = "True Function")
scatter!(x_obs_err, y_obs_err, label = "Observations")
plot!(x_all, μ_err, label = "Predicted")
plot!(x_all, μ_err - std95, fillrange = μ_err + std95, fillalpha = 0.3, label = "Confidence Interval")
# Setting axes limits:
xlims!(-4, 4)
ylims!(-3, 13)
# Title:
title!("Error Based GP for l = $ls, noise std = $noise_std")
# Saving the Plot:
savefig("ErrorBased_l_$ls.png")

# Low confidence Bound:
α = 1.96
LB_index = argmin(μ-α*sqrt.(σ²))
x_LB = x_all[LB_index]
f_LB = μ[LB_index]
f_model_LB = f.(x_LB)
println("Sixth point for evaluation, Lower Bound Based: x = $x_LB, f(x) = $f_LB")
# Updating Samples:
x_obs_LB = copy(x_obs)
push!(x_obs_LB, x_LB)
y_obs_LB = copy(y_obs)
push!(y_obs_LB, f_model_LB)
# Refitting the Gaussian Model:
m = MeanZero()
kernel = SE(log(ls), 0.0)
gp = GP(x_obs_LB, y_obs_LB, m, kernel, log(noise_std))
μ_LB, σ²_LB = predict_y(gp, x_all)
std95 = 1.96 * sqrt.(σ²_LB)
# Plotting:
plt = plot(x_all, y_all, label = "True Function")
scatter!(x_obs_LB, y_obs_LB, label = "Observations")
plot!(x_all, μ_LB, label = "Predicted")
```

```
plot!(x_all, μ_LB - std95, fillrange = μ_LB + std95, fillalpha = 0.3, label = "Confidence Interval")
# Setting axes limits:
xlims!(-4, 4)
ylims!(-3, 13)
# Title:
title!("LCB Based GP for l = $ls, noise std = $noise_std")
# Saving the Plot:
savefig("LCBBased_l_$ls.png")
# Printing the 7th Point:
LB_index = argmin(μ_LB - α * sqrt.(σ²_LB))
x_LB = x_all[LB_index]
f_LB = μ_LB[LB_index]
println("Seventh point for evaluation, Lower Bound Based: x = $x_LB, f(x) = $f_LB")
```