**Code Presentation**

The Bayesian Optimization code is well-structured and includes detailed comments to ensure clarity and ease of implementation. It is divided into modular functions:

1. **black\_box\_function(x)**: Defines the target function to optimize. In this example, it is a noisy sine function.
2. **expected\_improvement(X, X\_sample, Y\_sample, gpr, xi)**: Implements the Expected Improvement acquisition function, which guides the optimization process.
3. **propose\_next\_point(X\_sample, Y\_sample, gpr, bounds, n\_restarts)**: Suggests the next sampling point by maximizing the acquisition function.
4. **bayesian\_optimization(n\_iterations, bounds, noise)**: Runs the main optimization loop, updating the Gaussian Process model and querying new points iteratively.
5. **Plotting Results**: The code includes a visualization of the true function, predicted mean, confidence intervals, and sampled points.

The functions are parameterized to allow flexibility, and the code uses standard Python libraries such as NumPy, Matplotlib, and Scikit-learn. Proper error handling and shape management ensure robustness. This structure makes it easy for others to adapt the code to different problems or datasets. A Jupyter Notebook format would be ideal to present the code, intermediate results, and final output in a step-by-step manner.

more on the optimal regions, improving accuracy while saving resources.