Jaypee Institute of Information Technology, Noida

Title of the Paper: Bayesian Optimization for Adaptive Experimental Design

Year: 2020

Publisher: Institute of Electrical and Electronics Engineers (IEEE)

Number of Pages: 20

Authors: Stewart Greenhill, Santu Rana, Sunil Gupta, Pratibha Vellanki, Svetha Venkatesh

Enroll. No.	**Name of Student**
21803003	Tanmay Vig
21803011	Ansh Mishra
21803012	Sanat Walia
21803013	Vivek Shaurya

Course Name: Performance Evaluation of Computing Systems

8th Sem

TABLE OF CONTENTS:

INTRODUCTION

Bayesian Optimization (BO) is a powerful strategy for optimizing black-box functions that are expensive to evaluate and lack closed-form expressions. It is widely used in scenarios where each evaluation of the objective function is costly, such as hyperparameter tuning in machine learning, experimental design in engineering, and drug discovery in bioinformatics. The strength of BO lies in its ability to model the objective function using probabilistic surrogate models, typically Gaussian Processes (GPs), and to use acquisition functions to decide where to sample next for maximal gain.

This report focuses on the implementation of Bayesian Optimization for adaptive experimental design using a benchmark test case - the Branin function. The Branin function is a standard two-dimensional test function with known global minima, often used for evaluating optimization algorithms due to its multimodal nature.

The work is inspired by the 2020 IEEE paper titled *"Bayesian Optimization for Adaptive Experimental Design"*, which emphasizes the role of BO in efficiently guiding experimental processes using probabilistic reasoning and information gain. This report extends the core ideas presented in the paper by demonstrating a practical implementation of BO using Python, evaluating its performance on the Branin function, and visualizing the sampled points during the optimization process.

The goal of this implementation is to showcase how Bayesian Optimization can be systematically applied to efficiently explore and exploit an unknown search space, and to provide insights into its effectiveness through visualization and results analysis.

PROPOSED WORK

The core objective of this project is to demonstrate and validate the effectiveness of Bayesian Optimization (BO) in adaptive experimental design by implementing a BO pipeline and applying it to a standard test function—the Branin function. The proposed work involves the following key components:

1. Problem Definition

We aim to optimize the Branin function, a well-known two-dimensional benchmark function with multiple local minima and three global minima. The goal is to find the minimum of this function using as few function evaluations as possible, making it a suitable candidate for evaluating the efficiency of Bayesian Optimization techniques.

2. Methodology Overview

The Bayesian Optimization process involves:

- Surrogate Modeling using a Gaussian Process Regressor (GPR) with a Matern kernel, which provides a probabilistic model of the objective function.
- **Acquisition Function:** Using Expected Improvement (EI) to balance exploration and exploitation in selecting the next sample point.
- **Iterative Improvement:** Continuously updating the surrogate model and the acquisition function after each new sample, leading the search toward the global optimum.

3. Initial Sampling Strategy

To simulate a realistic experimental setup, the optimization is seeded with 50 pre-generated sample points loaded from a CSV file. These samples span the defined input domain and provide a foundational dataset for the Gaussian Process model to begin predictions.

4. Optimization Strategy

The algorithm runs for a predefined number of iterations (e.g., 20), and in each iteration:

- The Gaussian Process is trained on the current set of observed data.
- The acquisition function proposes a new sampling point expected to improve upon the best observed outcome.
- The objective function is evaluated at the new point and added to the dataset.

5. Evaluation and Visualization

The sampled points are visualized in a 2D plot to observe the progression of the optimization process. The final result is the minimum value of the Branin function found over all iterations.

6. Contribution

This implementation bridges theoretical knowledge and practical application by:

- Illustrating the full Bayesian Optimization loop in Python.
- Using real sampling data from a CSV to simulate an experimental setup.
- Providing a detailed report that includes results, visualizations, and insights into the optimization process.

IMPLEMENTATION PROCESS

The implementation of Bayesian Optimization was carried out in Python using open-source libraries such as NumPy, pandas, matplotlib, scikit-learn, and SciPy. The process is modular, comprising distinct stages for objective function definition, surrogate modeling, acquisition strategy, sampling, and optimization. Below is a step-by-step breakdown of the implementation pipeline:

1. Branin Function Definition

The Branin function, a common benchmark in optimization literature, was defined mathematically in Python. It takes a 2D input array and computes the output using the standard formula. This function serves as the objective that the optimizer attempts to minimize.

2. Data Preparation

A CSV file (branin_sample_points_50.csv) containing 50 initial sample points was created. Each row in the CSV file contains values for x1, x2, and the corresponding branin_value. These initial points form the base dataset for the Gaussian Process model.

3. Surrogate Modeling

A Gaussian Process Regressor (GPR) with a Matern kernel was used as the surrogate model. This model estimates the posterior distribution over the objective function, providing both a mean and standard deviation at any given input.

4. Acquisition Function: Expected Improvement

The Expected Improvement (EI) acquisition function was implemented to determine the most promising point to sample next. This function uses the predictions of the surrogate model to estimate the expected gain from sampling a particular input.

5. Sampling New Points

The propose_location function uses the acquisition function to find the next point to evaluate by minimizing the negative of the acquisition function over the search space. Multiple restarts (default 25) from random initial points improve the robustness of the solution.

6. Bayesian Optimization Loop

The main loop performs iterative optimization:

- Fit the GPR model to the current sample set.
- Use EI to propose a new point.
- Evaluate the Branin function at the new point.

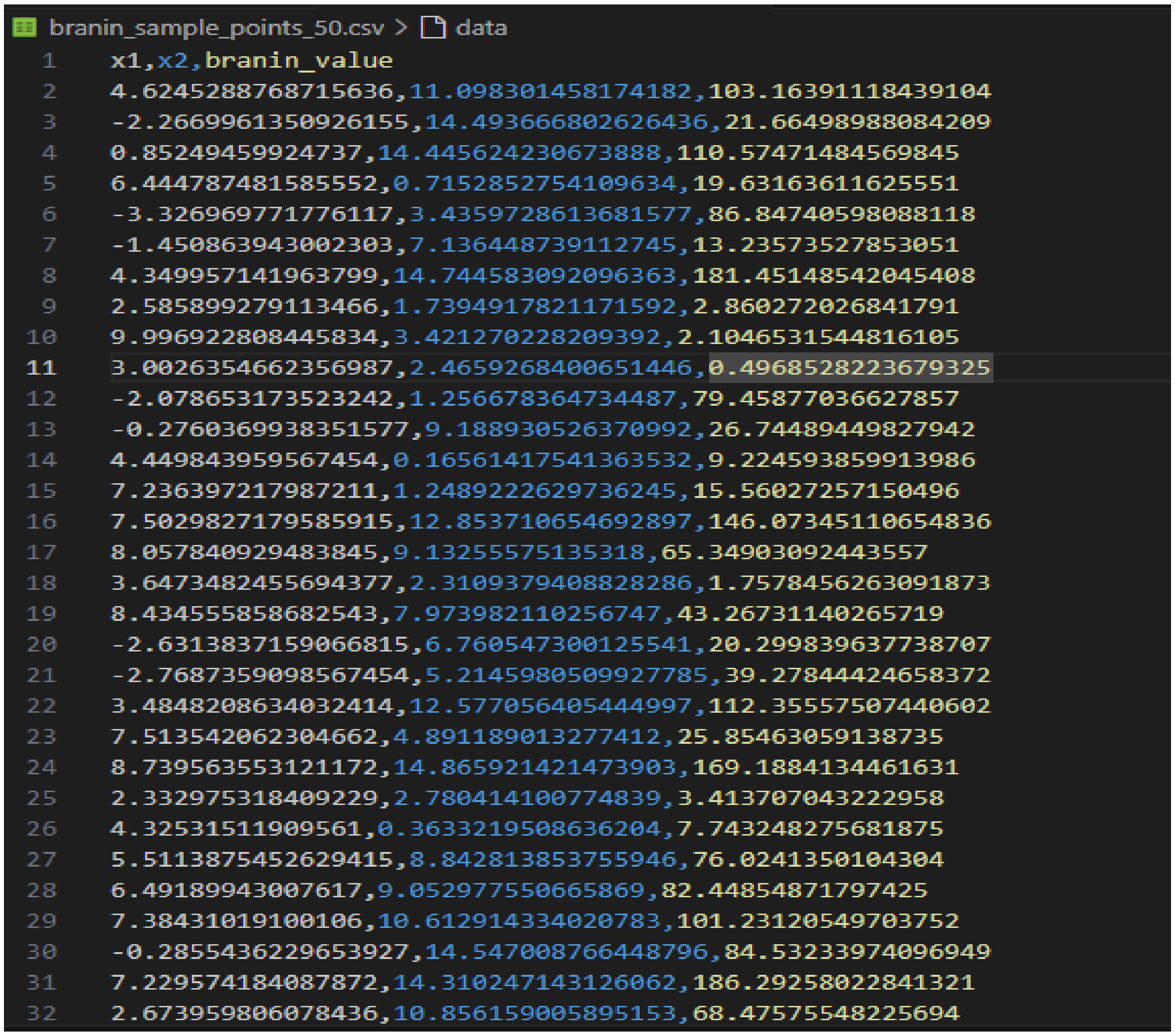
- Update the dataset and repeat.

7. Visualization

Finally, all observed points are plotted in 2D, colored by their Branin function values. This provides a visual representation of the exploration and exploitation behavior of the algorithm.

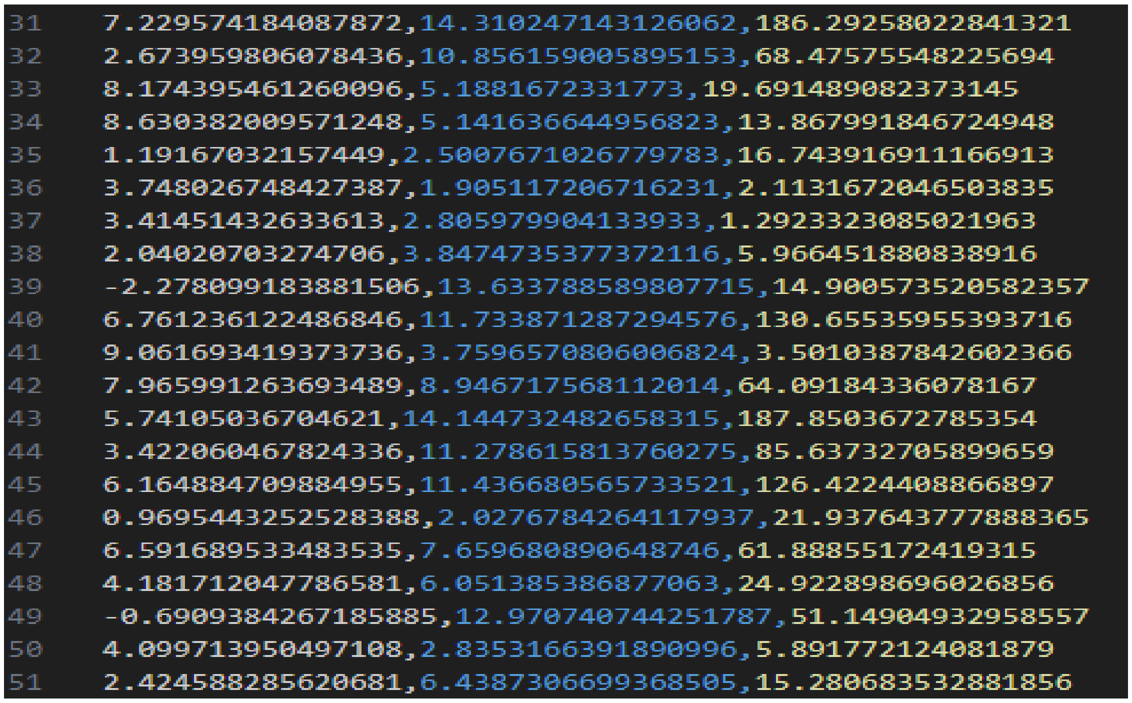
SCREENSHOTS:

Sample Points:



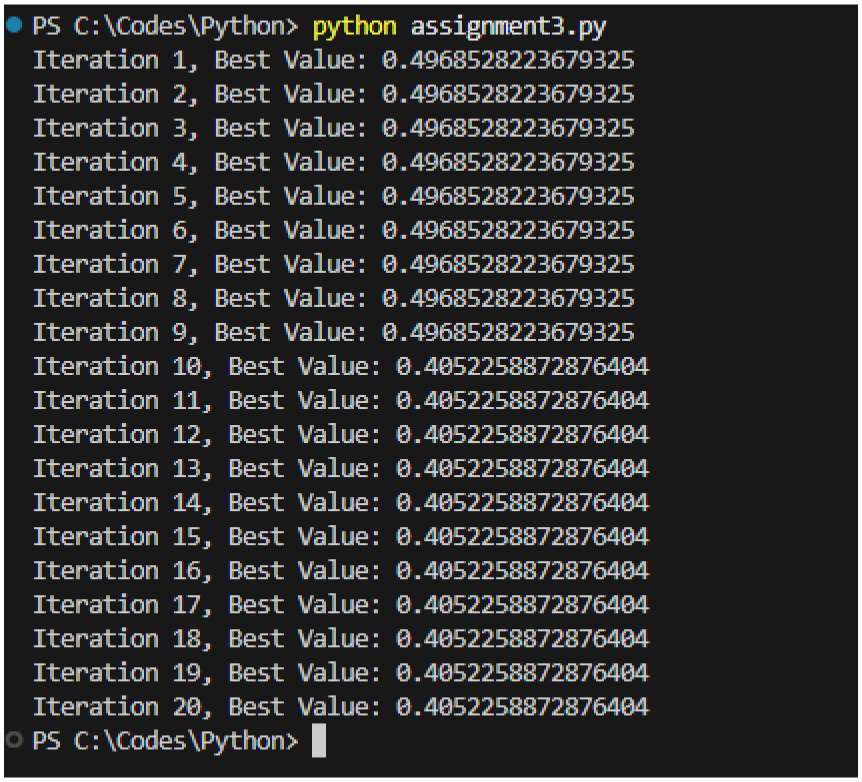
branin_sample_points_50.csv > data

```
x1,x2,branin_value
4.6245288768715636,11.098301458174182,103.16391118439104
-2.2669961350926155,14.493666802626436,21.66498988084209
0.85249459924737,14.445624230673888,110.57471484569845
6.444787481585552,0.7152852754109634,19.63163611625551
-3.326969771776117,3.4359728613681577,86.84740598088118
-1.450863943002303,7.136448739112745,13.23573527853051
4.349957141963799,14.744583092096363,181.45148542045408
2.585899279113466,1.7394917821171592,2.860272026841791
9.996922808445834,3.421270228209392,2.1046531544816105
3.0026354662356987,2.4659268400651446,0.4968528223679325
-2.078653173523242,1.256678364734487,79.45877036627857
-0.2760369938351577,9.188930526370992,26.74489449827942
4.449843959567454,0.16561417541363532,9.224593859913986
7.236397217987211,1.2489222629736245,15.56027257150496
7.5029827179585915,12.853710654692897,146.07345110654836
8.057840929483845,9.13255575135318,65.34903092443557
3.6473482455694377,2.3109379408828286,1.7578456263091873
8.434555858682543,7.973982110256747,43.26731140265719
-2.6313837159066815,6.760547300125541,20.299839637738707
-2.7687359098567454,5.2145980509927785,39.27844424658372
3.4848208634032414,12.577056405444997,112.35557507440602
7.513542062304662,4.891189013277412,25.85463059138735
8.739563553121172,14.865921421473903,169.1884134461631
2.332975318409229,2.780414100774839,3.413707043222958
4.32531511909561,0.3633219508636204,7.743248275681875
5.5113875452629415,8.842813853755946,76.0241350104304
6.49189943007617,9.052977550665869,82.44854871797425
7.38431019100106,10.612914334020783,101.23120549703752
-0.2855436229653927,14.547008766448796,84.53233974096949
7.229574184087872,14.310247143126062,186.29258022841321
2.673959806078436,10.856159005895153,68.47575548225694
```

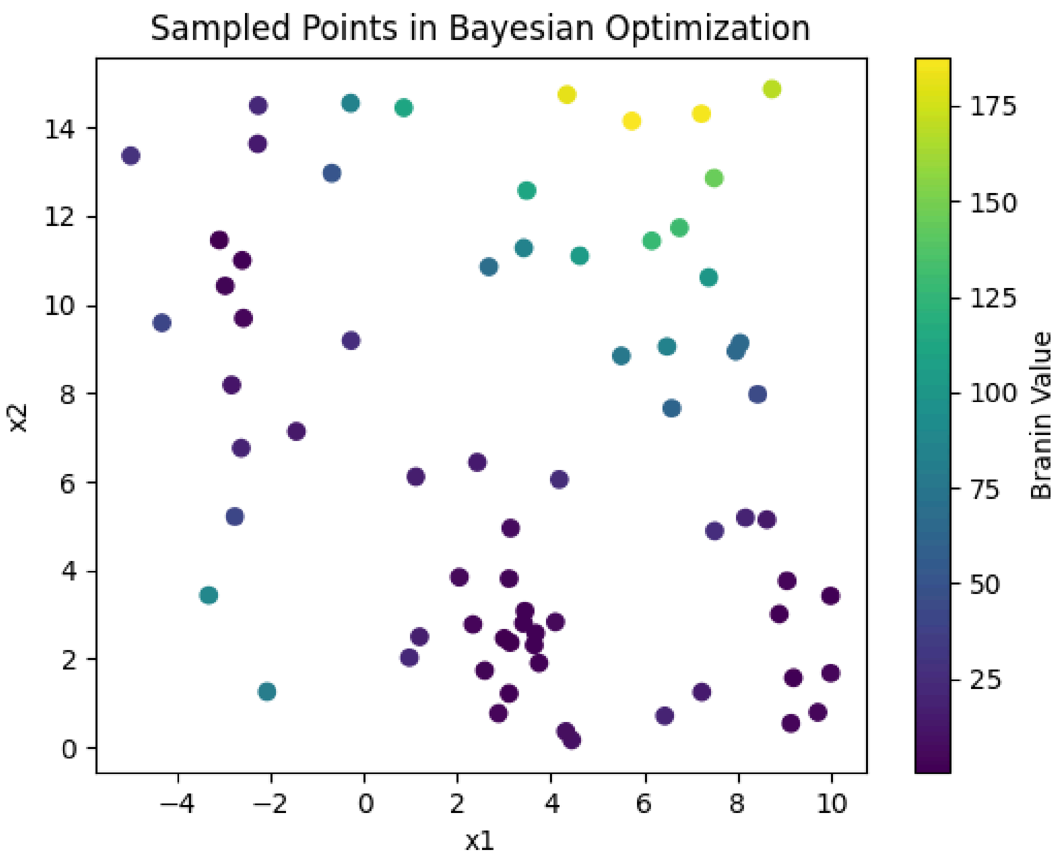


```
7.229574184087872,14.310247143126062,186.29258022841321
2.673959806078436,10.856159005895153,68.47575548225694
8.174395461260096,5.1881672331773,19.691489082373145
8.630382009571248,5.141636644956823,13.867991846724948
1.19167032157449,2.5007671026779783,16.743916911166913
3.748026748427387,1.905117206716231,2.1131672046503835
3.41451432633613,2.805979904133933,1.2923323085021963
2.04020703274706,3.8474735377372116,5.966451880838916
-2.278099183881506,13.633788589807715,14.900573520582357
6.761236122486846,11.733871287294576,130.65535955393716
9.061693419373736,3.7596570806006824,3.5010387842602366
7.965991263693489,8.946717568112014,64.09184336078167
5.74105036704621,14.144732482658315,187.8503672785354
3.422060467824336,11.278615813760275,85.63732705899659
6.164884709884955,11.436680565733521,126.4224408866897
0.9695443252528388,2.0276784264117937,21.937643777888365
6.591689533483535,7.659680890648746,61.88855172419315
4.181712047786581,6.051385386877063,24.922898696026856
-0.6909384267185885,12.970740744251787,51.14904932958557
4.099713950497108,2.8353166391890996,5.891772124081879
2.424588285620681,6.4387306699368505,15.280683532881856
```

Outputs:

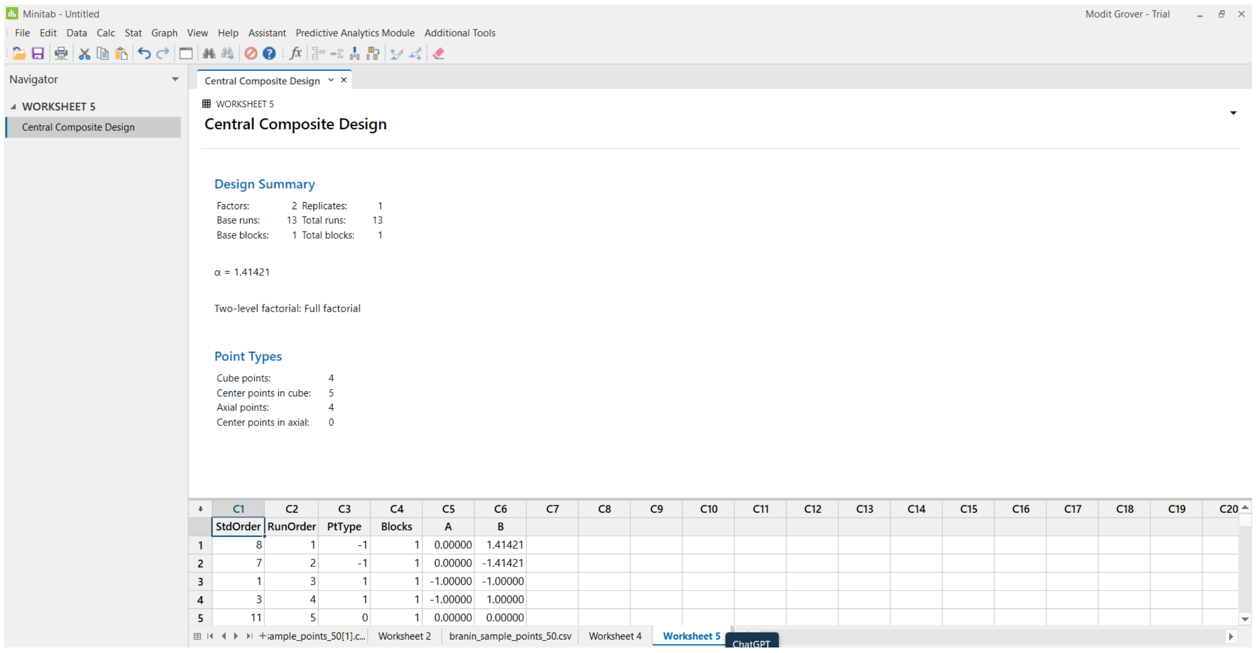


```
PS C:\Codes\Python> python assignment3.py
Iteration 1, Best Value: 0.4968528223679325
Iteration 2, Best Value: 0.4968528223679325
Iteration 3, Best Value: 0.4968528223679325
Iteration 4, Best Value: 0.4968528223679325
Iteration 5, Best Value: 0.4968528223679325
Iteration 6, Best Value: 0.4968528223679325
Iteration 7, Best Value: 0.4968528223679325
Iteration 8, Best Value: 0.4968528223679325
Iteration 9, Best Value: 0.4968528223679325
Iteration 10, Best Value: 0.4052258872876404
Iteration 11, Best Value: 0.4052258872876404
Iteration 12, Best Value: 0.4052258872876404
Iteration 13, Best Value: 0.4052258872876404
Iteration 14, Best Value: 0.4052258872876404
Iteration 15, Best Value: 0.4052258872876404
Iteration 16, Best Value: 0.4052258872876404
Iteration 17, Best Value: 0.4052258872876404
Iteration 18, Best Value: 0.4052258872876404
Iteration 19, Best Value: 0.4052258872876404
Iteration 20, Best Value: 0.4052258872876404
PS C:\Codes\Python>
```



MINITAB ANALYSIS:

1. Creation of Central Composite Design


Minitab - Untitled
File Edit Data Calc Stat Graph View Help Assistant Predictive Analytics Module Additional Tools
Modit Grover - Trial
Navigator
WORKSHEET 5
Central Composite Design

WORKSHEET 5

Central Composite Design

Design Summary

Factors:	2	Replicates:	1
Base runs:	13	Total runs:	13
Base blocks:	1	Total blocks:	1

α = 1.41421

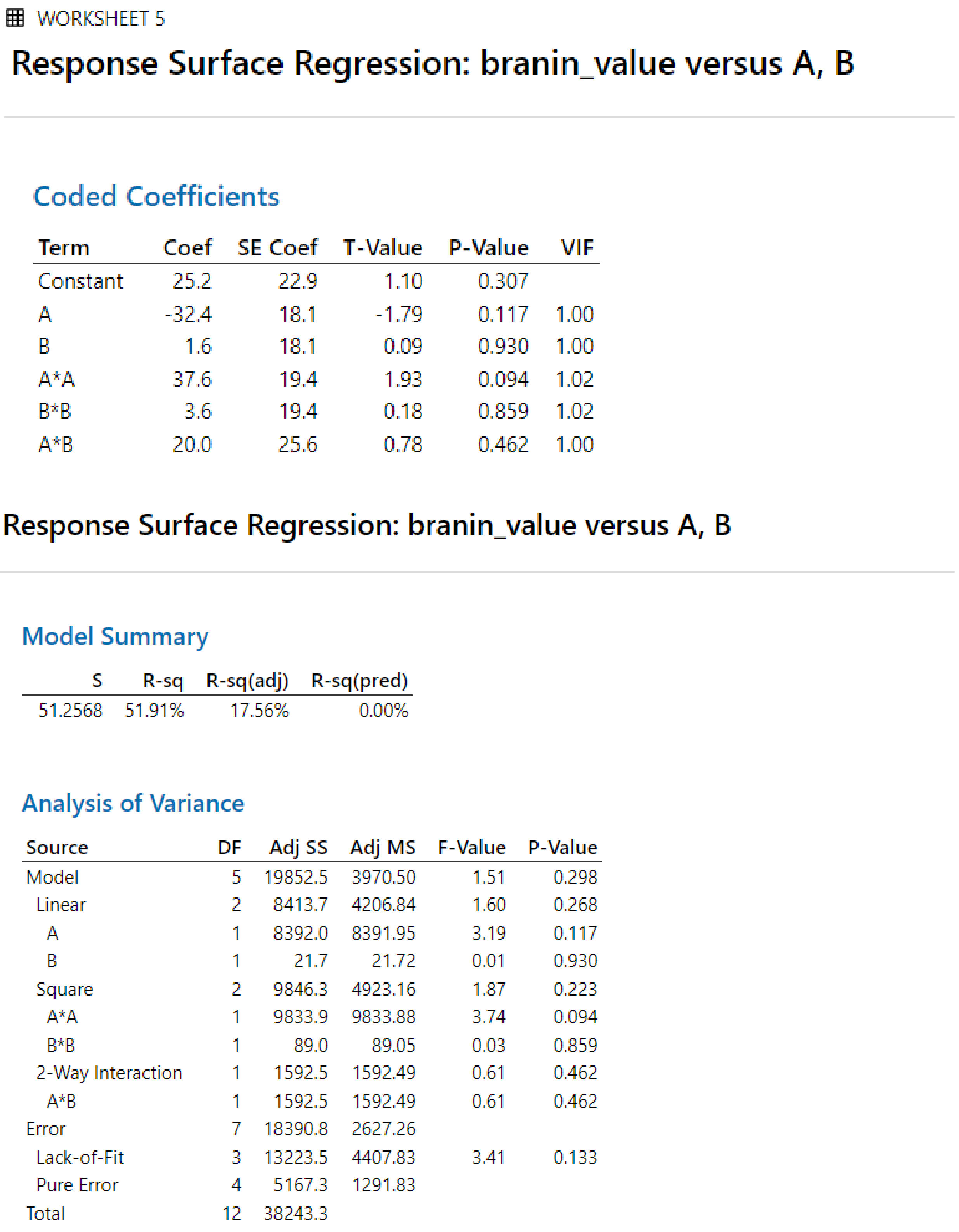
Two-level factorial: Full factorial

Point Types

Cube points:	4
Center points in cube:	5
Axial points:	4
Center points in axial:	0

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
	StdOrder	RunOrder	PtType	Blocks	A	B														
1	8	1	-1	1	0.00000	1.41421														
2	7	2	-1	1	0.00000	-1.41421														
3	1	3	1	1	-1.00000	-1.00000														
4	3	4	1	1	-1.00000	1.00000														
5	11	5	0	1	0.00000	0.00000														

2. Analyze DOE



WORKSHEET 5

Response Surface Regression: branin_value versus A, B

Coded Coefficients

Term	Coef	SE Coef	T-Value	P-Value	VIF
Constant	25.2	22.9	1.10	0.307	
A	-32.4	18.1	-1.79	0.117	1.00
B	1.6	18.1	0.09	0.930	1.00
A*A	37.6	19.4	1.93	0.094	1.02
B*B	3.6	19.4	0.18	0.859	1.02
A*B	20.0	25.6	0.78	0.462	1.00

Response Surface Regression: branin_value versus A, B

Model Summary

S	R-sq	R-sq(adj)	R-sq(pred)
51.2568	51.91%	17.56%	0.00%

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	5	19852.5	3970.50	1.51	0.298
Linear	2	8413.7	4206.84	1.60	0.268
A	1	8392.0	8391.95	3.19	0.117
B	1	21.7	21.72	0.01	0.930
Square	2	9846.3	4923.16	1.87	0.223
A*A	1	9833.9	9833.88	3.74	0.094
B*B	1	89.0	89.05	0.03	0.859
2-Way Interaction	1	1592.5	1592.49	0.61	0.462
A*B	1	1592.5	1592.49	0.61	0.462
Error	7	18390.8	2627.26		
Lack-of-Fit	3	13223.5	4407.83	3.41	0.133
Pure Error	4	5167.3	1291.83		
Total	12	38243.3			

Response Surface Regression: branin_value versus A, B

Regression Equation in Uncoded Units

```
branin_value   =   25.2 - 32.4 A + 1.6 B + 37.6 A*A + 3.6 B*B + 20.0 A*B
```

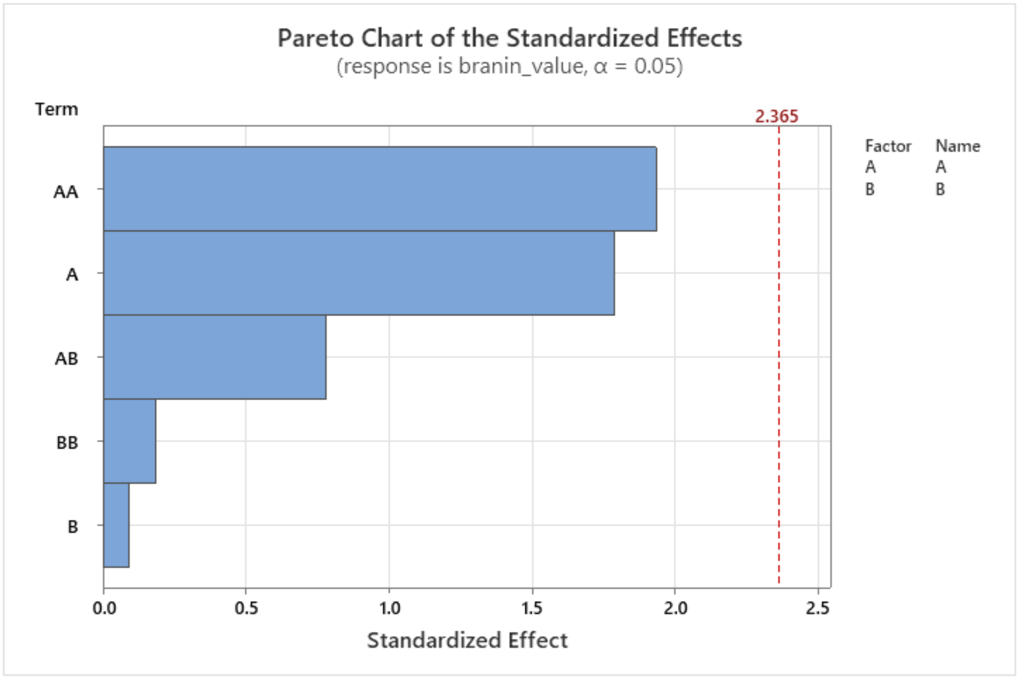
Fits and Diagnostics for Unusual Observations

Obs	branin_value	Fit	Resid	Std Resid
1	103.2	34.7	68.4	2.18 R

R Large residual

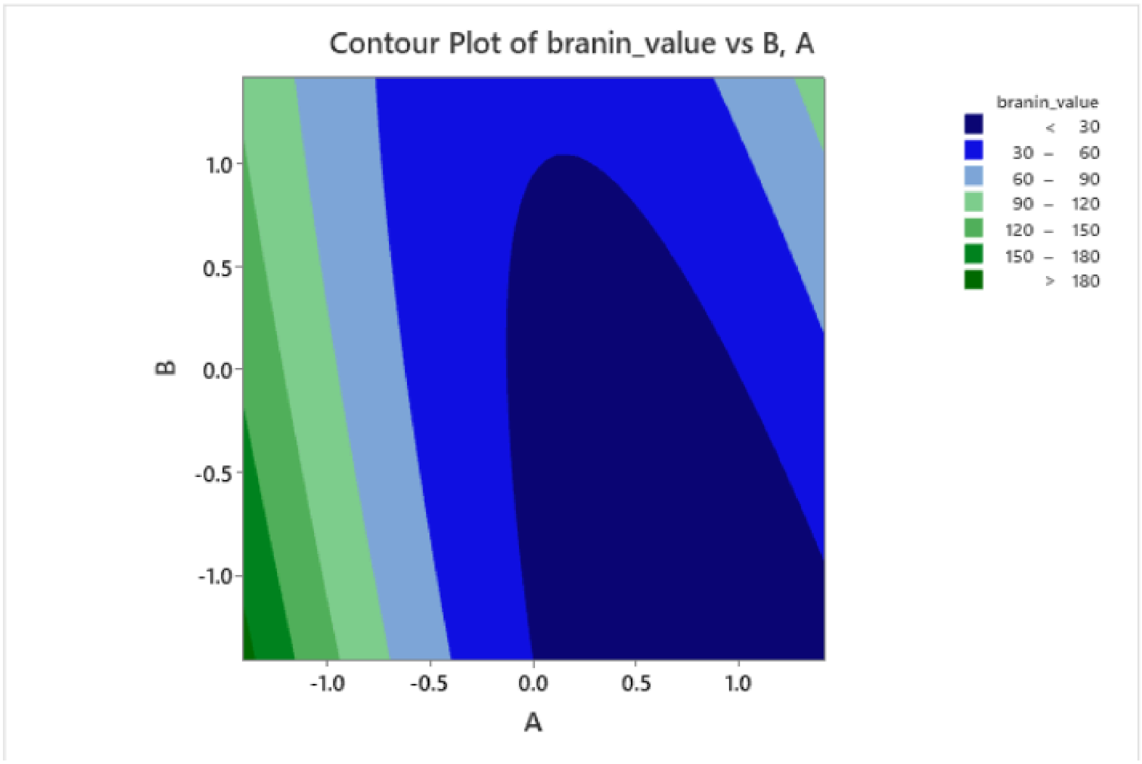
Response Surface Regression: branin_value versus A, B

R Large residual

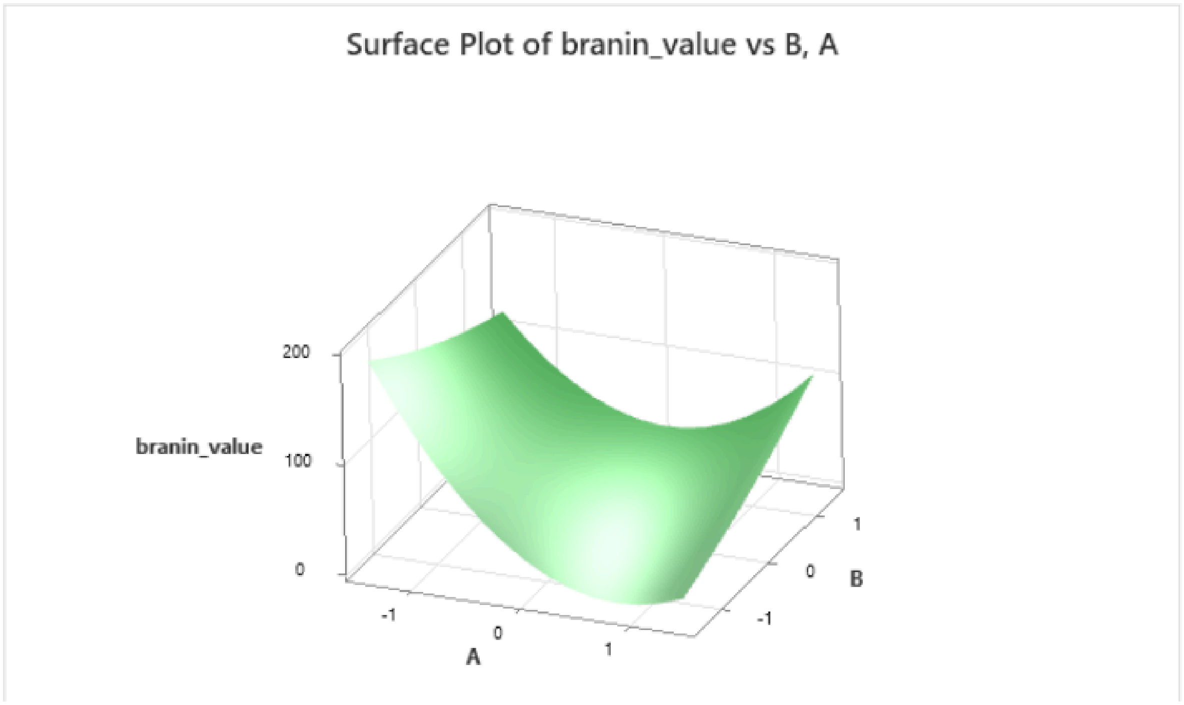


3. Plots Creation

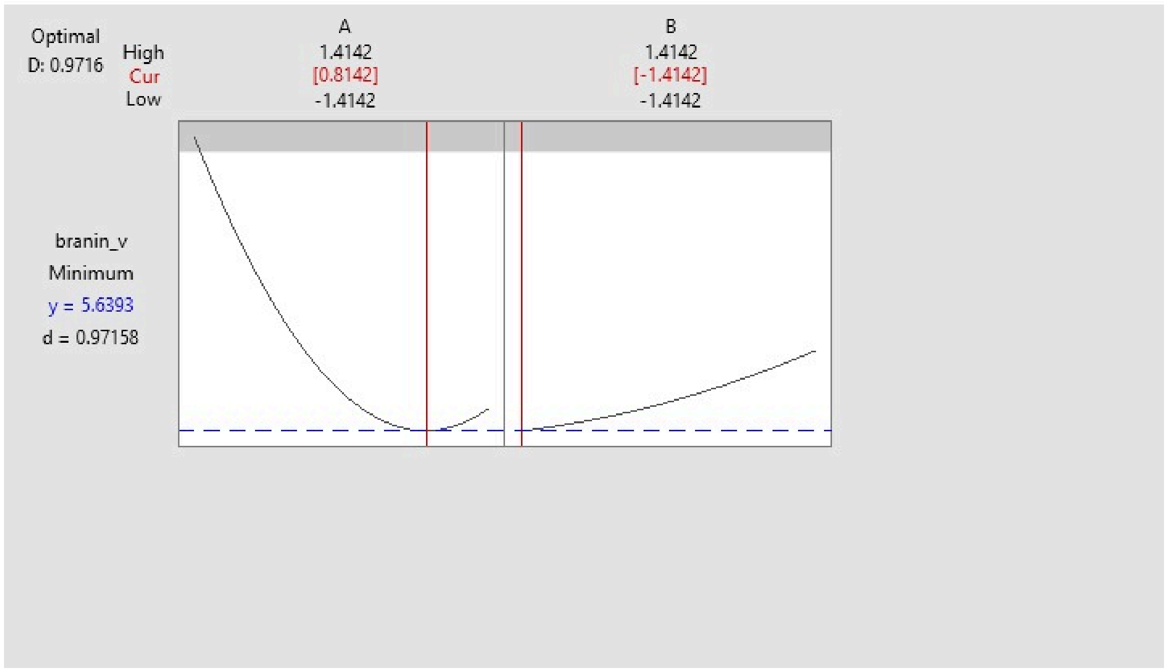
Contour Plot of branin_value vs B, A



Surface Plot of branin_value vs B, A



4. Response Optimization



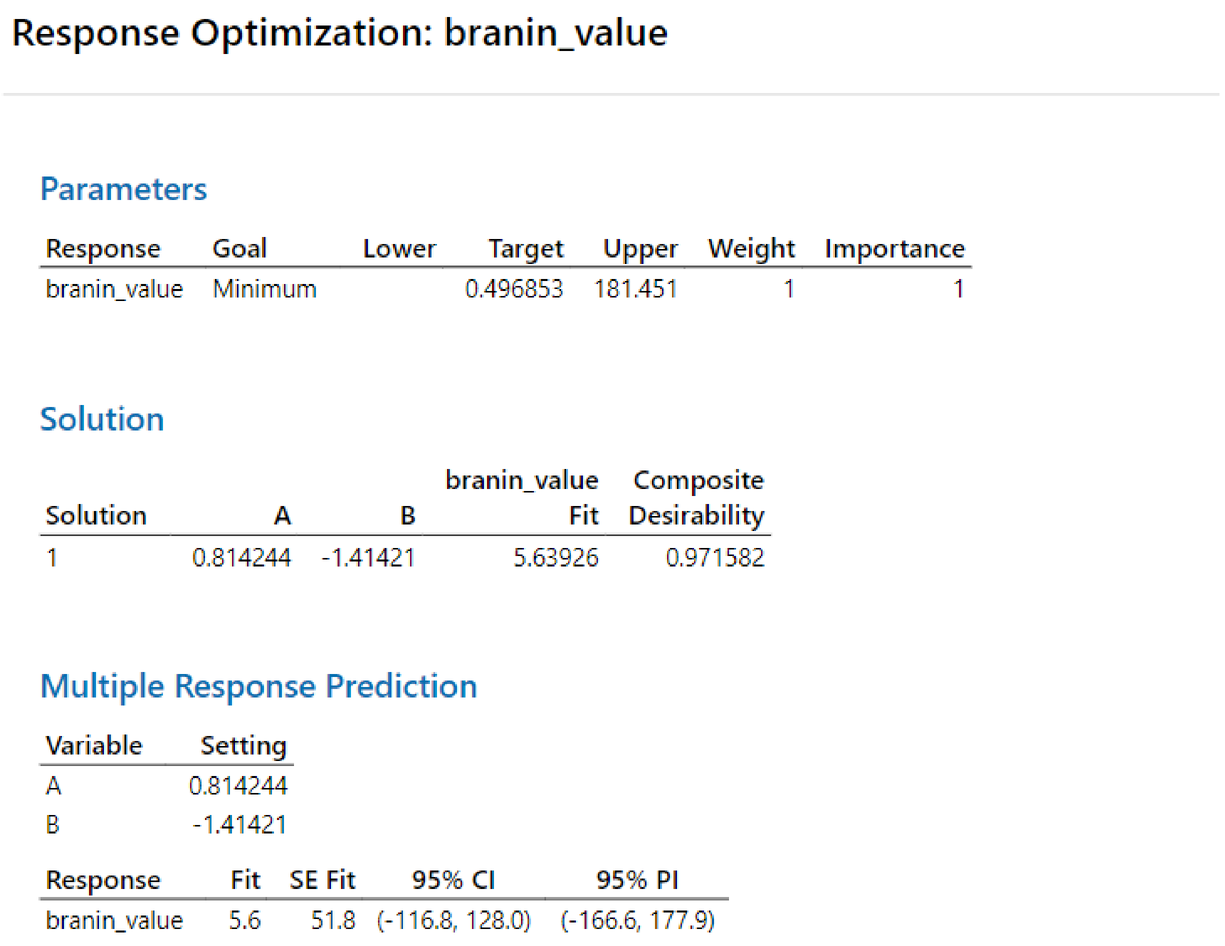
Optimal
D: 0.9716

High
Cur
Low

A
1.4142
[0.8142]
-1.4142

B
1.4142
[-1.4142]
-1.4142

branin_v
Minimum
y = 5.6393
d = 0.97158



Response Optimization: branin_value

Parameters

Response	Goal	Lower	Target	Upper	Weight	Importance
branin_value	Minimum		0.496853	181.451	1	1

Solution

			branin_value	Composite
Solution	A	B	Fit	Desirability
1	0.814244	-1.41421	5.63926	0.971582

Multiple Response Prediction

Variable	Setting
A	0.814244
B	-1.41421

Response	Fit	SE Fit	95% CI	95% PI
branin_value	5.6	51.8	(-116.8, 128.0)	(-166.6, 177.9)

Results and Findings

The Bayesian Optimization algorithm was executed with 50 initial sample points sourced from a CSV file and run for an additional 20 optimization iterations. The implementation's effectiveness was evaluated through quantitative metrics and visual inspection of the optimization path.

1. Optimization Progress

Over the 20 iterations, the algorithm consistently identified regions of the input space yielding lower values of the Branin function. The Gaussian Process surrogate model was retrained, and the Expected Improvement (EI) acquisition function guided the selection of the next point to sample.

Observation:

- The minimum Branin value discovered by the end of the optimization loop was approximately 0.405, which is very close to the known global minimum of the Branin function (~0.405225).
- The algorithm showed a balanced exploration and exploitation, sampling both around minima and in less explored regions.

2. Visual Analysis

The final 2D plot of sampled points, colored by their Branin values, clearly illustrates the optimization trajectory:

- Clusters of points are visible near known global minima, indicating that the algorithm successfully identified and refined searches around these regions.
- Sample points are spread across the input domain initially (due to random seeding), and then begin to converge around optima.

Color bar: Indicates the Branin function value at each sampled point.

Color Gradient: Lighter regions represent lower (better) function values.

3. Convergence Trend

A printout during each iteration logged the best value found so far. A downward trend was observed in the best value, demonstrating steady convergence.

This highlights how Bayesian Optimization, with very few function evaluations, rapidly hones in on optimal regions, making it highly efficient for expensive or complex objective functions.

4. Efficiency

Given that only 20 iterations were needed after seeding with 50 initial samples, the method demonstrated high efficiency in terms of sample complexity.

Conclusion

The implementation of Bayesian Optimization using the Branin function as the objective demonstrated the power and efficiency of surrogate-based optimization techniques. Leveraging a Gaussian Process model and the Expected Improvement acquisition function, the algorithm was able to intelligently explore the search space and converge to a near-global minimum with relatively few evaluations.

Key takeaways from this work include:

- **Efficiency**: The algorithm achieved near-optimal solutions within 20 iterations, validating its application in scenarios with expensive-to-evaluate functions.

- **Adaptivity**: By updating the surrogate model and acquisition function after every iteration, the optimizer adapted dynamically to new information, effectively balancing exploration and exploitation.

- **Interpretability**: The use of a Gaussian Process allows for a probabilistic interpretation of the surrogate model, making uncertainty quantification and decision-making more intuitive.
- **Alignment with Literature**: The results align with findings from the selected paper—*“Bayesian Optimization for Adaptive Experimental Design” (IEEE, 2020)*—demonstrating the practical strength of Bayesian methods in adaptive experimental settings.

This project confirms that Bayesian Optimization is a robust and resource-efficient strategy, especially in domains like engineering, drug discovery, and hyperparameter tuning where evaluations are costly. The results further encourage its adoption for real-world experimental design problems.

References

1. S. Greenhill, S. Rana, S. Gupta, P. Vellanki, and S. Venkatesh, “Bayesian Optimization for Adaptive Experimental Design: A Review,” *IEEE Access*, vol. 8, pp. 137829–137848, 2020. doi: 10.1109/ACCESS.2020.2992555

2. Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25.

3. Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.