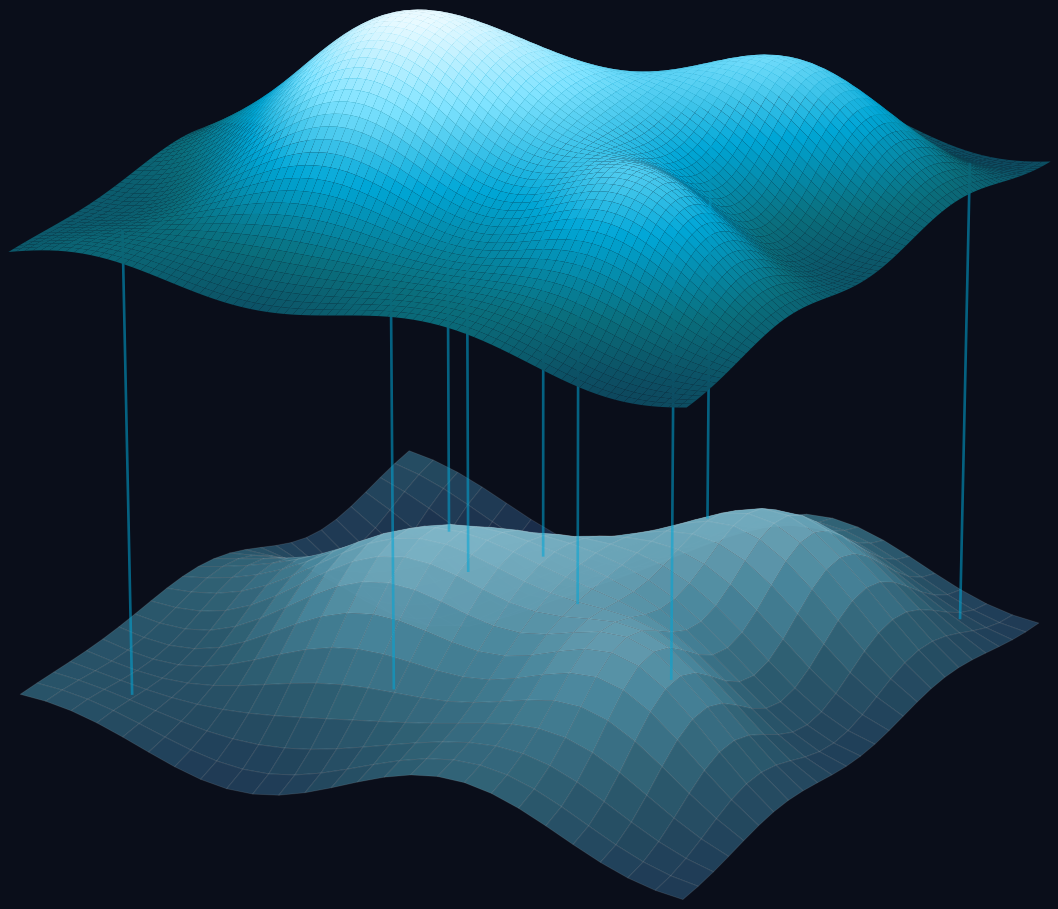
Combining an Asynchronous Multi-Point and a Multi-Fidelity Infill Strategy for Surrogate-Based Optimisation

An Empirical Evaluation on Unconstrained Problems

Ids Oostmeijer

Delft University of Technology



TU Delft

Combining an Asynchronous Multi-Point and a Multi-Fidelity Infill Strategy for Surrogate-Based Optimisation

An Empirical Evaluation on Unconstrained Problems

by

Ids Oostmeijer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 07 July 2026 at 13:00

Student number:	5098335	
Project duration:	March 17, 2025 – July 7, 2026	
Thesis committee:	Dr.ir. Gianfranco la Rocca	TU Delft, Chair
	Dr. Omid Nejadseyfi	TU Delft, Independent examiner
	Dr.ir. Maurice Hoogreef	TU Delft, Responsible supervisor
	Prof.dr.ir. Matthijs Langelaar	TU Delft, Supervisor
	Marek Slebioda	TU Delft, Supervisor

Cover: Multi-fidelity surrogate model landscape adapted from a Claude generated render.

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This thesis marks the end of my MSc Aerospace Engineering at the TU Delft, completed within the Flight Performance and Propulsion track. This has been a longer, more eventful period than anticipated, which I could not have completed on my own.

My interest in optimisation followed from a course on multidisciplinary design optimisation, taught by Dr. ir. Gianfranco la Rocca in the MSc curriculum. Fuelled by curiosity, I delved deeper into the field of optimisation by taking a course on engineering optimisation at the Mechanical Engineering faculty, taught by Prof.dr.ir. Matthijs Langelaar and Dr. Omid Nejadseyfi. The combination of the two courses sparked my interest in pursuing my MSc thesis in this domain.

I would like to express my gratitude towards my supervisors: Dr.ir. Maurice Hoogreef of the Aerospace Engineering faculty and Prof.dr.ir. Matthijs Langelaar and Marek Slebioda from the Mechanical Engineering faculty. I am grateful to each of them for their guidance and insights during our regular meetings, as well as their support and flexibility following an unexpected pause and extension of the project. I would also like to thank Dr.ir. Gianfranco la Rocca and Dr. Omid Nejadseyfi for completing my thesis defence committee.

My years in Delft were shaped in large part by competitive rowing, both as an athlete and, in the past year, as a coach. The friendships, the training sessions, training camps and regattas have left an abundance of unforgettable memories. Combining this with the friendships I made through my studies has shaped me into who I am today. Finally, I want to thank my family for their continued support throughout my studies.

Ids Oostmeijer
Delft, June 2026

Summary

Modern engineering design often relies on optimisation, yet the computationally expensive High Fidelity (HF) analysis tools involved make direct optimisation computationally costly. Surrogate-Based Optimisation (SBO) mitigates this by generating a cheap surrogate from a limited set of training data. This surrogate is refined during the optimisation process by selecting infill points using an infill strategy. Two established ways of reducing the wall-clock time of an SBO run are utilising a Multi-Fidelity (MF) infill strategy, which uses cheaper but less accurate analysis tools, and Multi-Point (MP) infill strategies, which evaluate several infill points concurrently. Both approaches have been studied extensively in isolation, but their combination has received less attention, even though it may offer further performance improvements.

This thesis investigates the performance of a combined MP MF infill strategy. The proposed method combines an asynchronous MP infill strategy based on the scheme of Przysowa et al. with the MF Two-Step infill strategy by Garbo et al. The Two-Step strategy first selects the new sample location on the HF surrogate of a Hierarchical Kriging (HK) model. Using a Jensen-Shannon distance (JSd) threshold, the fidelity to be sampled is then selected as the cheapest one deemed sufficiently accurate. The proposed strategy was benchmarked on eleven two-fidelity numerical test functions, and compared against the baseline Single-Point (SP) Single-Fidelity (SF) Expected Improvement (EI) infill strategy, and the implemented MP and MF strategies separately, using the Expected Runtime (ERT) as the performance metric.

The asynchronous SF MP strategy achieved a statistically significant reduction in ERT of 72.7% over the baseline. The MF aspect, however, degraded performance compared to the SF equivalent, contrary to the hypothesis. This degradation was attributed to the addition of redundant Low Fidelity (LF) samples near the optimum, resulting from reselecting the sample location based on the residual variance of the HK surrogate model. By testing a set of eleven LF functions for the Forrester function, this behaviour was shown to be structural, rather than the result of the LF formulation; the decrease in performance was observed across the whole set. The hypothesis is therefore only partially confirmed: the asynchronous MP strategy delivers a significant reduction in wall-clock time, while the addition of MF partially offsets this performance gain. Dynamically adapting the fidelity-selection threshold during the optimisation, or substituting a different MF criterion, are identified as the most promising routes to realising the potential combined benefit.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms & Abbreviations

ACAS Aggregate-Criteria Adaptive Sampling
AMEI Adapative Multi-Fidelity Expected Improvement
atol absolute tolerance
CAND Candidate points algorithm
CD Critical-Difference
CEI Constrained Expected Improvement
CFD Computational Fluid Dynamics
CGEI Constrained Generalised Expected Improvement
CL Constant Liar
Co-KRG Co-Kriging
CPU Central Processing Unit
DCT Deb's Constraint Tournament
DOE Design of Experiments
EGO Efficient Global Optimisation
EI Expected Improvement
EIR Expected Improvement Reduction
ERT Expected Runtime
FEM Finite Element Method
GEI Generalised Expected Improvement
GEK Gradient-Enhanced Kriging
GIL Global Interpreter Lock
HF High Fidelity
HFM High Fidelity Model
HK Hierarchical Kriging
HPC High Performance Computing
I/O Input/Output
JSD Jensen-Shannon Divergence
JSd Jensen-Shannon distance
KB Kriging Believer
KL Kullback-Leibler
KRG Kriging
LB Lower Bound
LCB Lower Confidence Bound
LF Low Fidelity
LFM Low Fidelity Model
LHS Latin Hypercube Sampling
MF Multi-Fidelity
MFSM Multi-Fidelity Surrogate Model
MinP Minimising Prediction of Surrogate Model
MP Multi-Point
MPI Message Passing Interface
PoF Probability of Feasibility
PoI Probability of Improvement
POTI probability of Targeted Improvement
RAM Random-Access Memory
RBF Radial Basis Function
RMSE Root Mean Square Error
rtol relative tolerance
SBO Surrogate-Based Optimisation
SF Single-Fidelity
SLURM Simple Linux Utility for Resource Management, Scheduler used in the DelftBlue High Performance Computing (HPC).
SMT Surrogate Modelling Toolbox 2.0 [1]
SP Single-Point
STD Standard Deviation
TRBF auto-Tuned Radial Basis Function
UB Upper Bound

Latin Symbols

b Exploitation-Exploration parameter LCB
c Radial basis function constant
e Predicted error
$f(\cdot)$ Function
$\hat{f}(\cdot)$ Surrogate model function
g GEI factor
$g(\cdot)$ Inequality constraint
$G(\cdot)$ Random variable of a constraint
$h(\cdot)$ Equality constraint
$\mathbf{I}$ Identity matrix
$K_{i,j}$ Correlation between sample point i and j
$L(\cdot)$ Likelihood function
n Number of […], in case of infill strategy the number of ranks
p Smoothness parameter Radial basis function
p Gaussian exponential
r Euclidean distance
$\mathbf{R}$ Spatial Correlation in HK
s Mean square error
$\hat{s}(\cdot)$ Mean square error surrogate prediction
T Target value in PoI
w Weight coefficient
$\mathbf{W}$ Weight coefficient matrix
$\mathbf{W}$ Worker rank in asynchronous EGO
$\mathbf{x}$ Input vector
y Output value
$Z(\cdot)$ Random variable

Greek Symbols

β Scaling factor Hierarchical Kriging (HK)
ϵ Statistical Error Polynomial
τ Regression factor
$\delta(\cdot)$ Discrepancy function
μ Mean Kriging surrogate
ϕ Radial basis function
$\mathbf{\Phi}$ Gram matrix
Φ Cumulative distribution function
ϕ Probability density function
ψ Correlation
$\boldsymbol{\psi}$ Correlation vector
$\tilde{\boldsymbol{\psi}}$ Augmented correlation vector
$\mathbf{\Psi}$ Correlation matrix
$\rho(\cdot)$ Adjustment parameter
σ Standard deviation
θ Width parameter Kriging basis function

Number Sets

$\mathbf{S}$ Sample points
$\mathbf{W}$ Available worker ranks
$\mathbf{X}$ Sample points for surrogate generation
$\mathbf{Y}$ Output of sample points for surrogate generation

1

Introduction

In the design of a modern system, optimisation is often involved. For example, the minimisation of fuel burn for an aircraft or the minimisation of the weight of a structure. These optimisation problems often require the use of computationally expensive analysis tools. Utilising these tools in gradient-based optimisation algorithms can lead to extensive computation times as the optimiser requires the analysis tool to be run for a large number of points.
Reducing the computational cost and wall-clock time of these optimisations has been the goal of several studies [2], [3], [4]. One way to reduce this cost is to utilise Surrogate-Based Optimisation (SBO). These SBO solvers sample a significantly smaller number of points, as a surrogate is built from the data points on which the optimum is determined. Since this surrogate is cheap to evaluate, the optimum of this surrogate can be found easily using one of a plethora of optimisation algorithms.
Throughout the optimisation process this surrogate is updated by adding sample points evaluated using the expensive function. The location of these infill points is selected using an infill strategy that selects a point that improves the surrogate's optimality or ensures the surrogate is accurate in a region of interest. A commonly used SBO algorithm is Efficient Global Optimisation (EGO) using the Expected Improvement (EI) strategy on a Kriging (KRG) surrogate model (also called Gaussian process) [5], [6]. This surrogate model can, in addition to the expected value at a point, also return the variance of the prediction at this point. This variance is used in the EI infill criterion, which selects the new sample location by maximising the expected improvement of the optimum, balancing exploration and exploitation.
Studies aiming to improve the performance of these SBO algorithms can do so by changing the infill criterion. Improving the performance remains the goal of recent research. Two distinct approaches have been shown to reduce the optimisation wall-clock time, namely multi-fidelity or multi-point infill strategies [2], [7], [8]. The first utilises the multiple fidelities available to analysis tools. For example, using a coarse and fine mesh for a computational fluid dynamics solver. The Low Fidelity (LF) function will have a lower computational cost, but at the expense of the accuracy of the sampled points. This allows the optimiser to include more LF points, which can show the trend of the High Fidelity (HF) functions, which is updated with the difference between the LF and HF functions, aiming to have an accurate prediction of the HF function.
The second method utilises the increase in computational power by utilising multiple processors to sample multiple new points in the same iteration, increasing the data added to the surrogate for every iteration without an increase in wall-clock time.
Both methods have been shown to reduce the wall-clock time required for an SBO optimisation while converging to the optimum [2], [7], [8].

To the author's knowledge, the combination of multi-fidelity and multi-point algorithms has not been investigated thoroughly, potentially giving way to a further reduction in wall-clock time. Therefore, the aim of this MSc thesis is to investigate the performance change enabled by utilising a Multi-Point (MP) Multi-Fidelity (MF) infill strategy.

This was attempted by combining an existing multi-fidelity and multi-point algorithm, after which the combined infill strategy was tested on a set of 11 numerical test functions, comparing the results against the combined infill strategy alone and against the commonly used single-point single-fidelity EI infill strategy.
Following these numerical test functions, a set of varying LF functions for the Forrester function was investigated to get a better insight into the multi-fidelity aspect of the proposed infill strategy.

First, the notation used in this thesis is introduced in chapter 2, followed by an overview of the current literature on the topic of SBO in chapter 3. Next, the potential performance gains, along with the corresponding research question and hypothesis, are discussed in chapter 4. This chapter also introduces the proposed combined infill strategy. Next, the verification of the implementation is discussed in chapter 5. This is followed by the experimental setup in chapter 6, after which the results of the numerical test functions and the varying LF Forrester functions are presented in chapter 7 and chapter 8 respectively. Lastly, the discussion on the results is outlined in chapter 9, concluding with the conclusion in chapter 10.

2

Notation

In this thesis, bold font is used to represent vectors and matrices. Matrices will be denoted with capitalised symbols, such as $\mathbf{X}$, while vectors will use lowercase letters, such as $\mathbf{x}$.

A specific element in a vector is indicated with a subscript i, for example, x_i. For matrices, subscripts i and j are used, where $x_{i,j}$ represents the entry in the ith row and the jth column. Additionally, the subscript k will be used to denote the iteration number.

Sample points will have a superscript to indicate their respective sample point numbers. For example, $\mathbf{x}^1$ represents the first sample point in the sample set $\mathbf{X} = \left[\mathbf{x}^1, \ldots, \mathbf{x}^N\right]$.

Functions will be denoted with lowercase letters, such as $f(\mathbf{x})$. If a function is a surrogate approximation, this will be indicated with a hat, as in $\hat{f}(\mathbf{x})$.

Optimum values will be indicated with a superscript asterisk, such as x^*.

3

Literature Review

This chapter introduces optimisation and Surrogate-Based Optimisation (SBO), focusing on Multi-Fidelity (MF) and Multi-Point (MP) infill strategies. It begins by briefly introducing optimisation and SBO in section 3.1, followed by a discussion of common surrogate modelling techniques and infill strategies in section 3.2. Next, the chapter explores MF SBO and the current techniques, in section 3.3, with a set of the corresponding MF infill strategies presented in section 3.4. Finally, the current methods employed in MP infill strategies are introduced in section 3.5. Next, a possible improvement to the strategies is discussed in section 3.6. The chapter is concluded with the concluding remarks in section 3.7. The goal of the discussion in this chapter is to provide an overview of the state-of-the-art methods used in MP and MF infill strategies.

3.1. Optimisation

Originally, engineering design was performed using hand-based calculations in a trial-and-error method to determine the best combination of possible design parameters to find the optimum design. This process has since evolved to be performed using computer-based numerical algorithms. However, the general process is the same: From an initial guess, the optimisation process is started, in which parameters are altered to ultimately find the optimum combination for a given objective to be optimised. This objective can be cost, structural weight, a performance metric or any other parameter.
The development of optimisation algorithms replaced the trial-and-error method, speeding up the optimisation process and enabling the solution of problems of increasing complexity. Numerical optimisation first emerged in operations research and was subsequently applied to many other fields where numerical optimisation problems arise, including engineering. This optimisation process can be used in any of the iterative loops in the inherently iterative design process. This first requires the correct formulation of the optimisation and its constraints [6], [9]. This leads to the following general formulation of an optimisation problem:

$$\begin{aligned} &\min f(\mathbf{x}), \\ &\text{s.t.} \quad \mathbf{x}^l \leq \mathbf{x}_i \leq \mathbf{x}^u, \\ &\qquad g_j(\mathbf{x}) \leq 0, \quad j = 1, \dots, m, \\ &\qquad h_j(\mathbf{x}) = 0, \quad j = 1, \dots, p. \end{aligned} \tag{3.1}$$

In which $f(\mathbf{x})$ is the function to optimise, $\mathbf{x} = \left\{x_1, x_2, \dots, x_n\right\}^T$ is the design vector with scalar entries x_n. This is subject to limits to the design vector; x^l and x^u are the lower and upper limits, respectively. In addition, the optimisation can be subject to inequality and equality constraints; $g_j(\mathbf{x})$ and $h_j(\mathbf{x})$.

3.2. Surrogate-Based Optimisation

In engineering optimisation, computationally expensive analysis tools, such as Computational Fluid Dynamics (CFD) simulations or Finite Element Method (FEM) analyses, are often used. Utilising these tools can result in computationally costly optimisation problems. An optimisation technique that tries to

reduce the computational cost is SBO.
In SBO, the optimisation is performed using a surrogate in place of one or several disciplines or part of the optimisation process, or the optimisation is performed on a surrogate constructed from the objective and constraints. These surrogates are created using sampling points from the optimisation or the expensive discipline. In both cases, commonly, additional points are added to the set of sampling points with which the surrogate is updated; these so-called infill points improve the accuracy of the surrogate. This results in the generalised SBO workflow as illustrated in Figure 3.1. As a result of using the surrogate, which is less expensive to evaluate, in the optimisation process, the clock time for the complete optimisation process can be reduced.

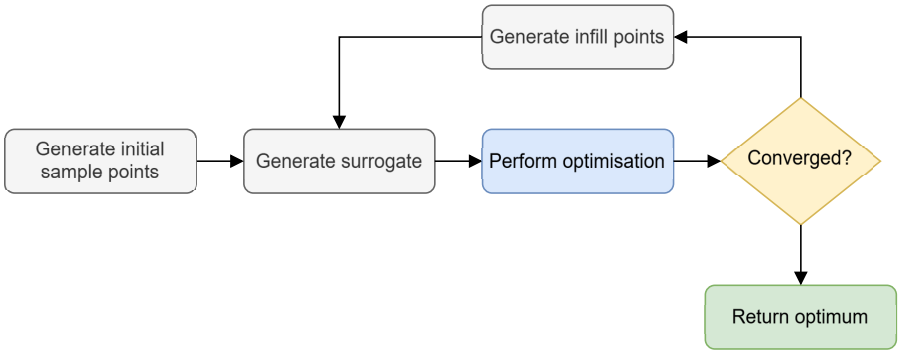


Figure 3.1: The generalised workflow of a SBO problem.

3.2.1. When to use Surrogate-Based Optimisation

Using an SBO algorithm can thus reduce the computational cost of an optimisation. However, not all problems can benefit from an SBO over directly solving the optimisation problem. A case in which an SBO can improve the performance is when the required number of expensive function evaluations to generate the surrogate is lower than the number of evaluations to solve the optimisation problem directly.
Another possible use of SBO is when the optimisation involves noisy models, such as experimental data. This noisy data can be smoothed in the surrogate, possibly improving the performance of gradient-based optimisation methods [6]. Another advantage of using SBO for experimental (and other discrete) data is the ability to create a continuous surrogate from discrete data points. Lastly, data from several sources can be combined in a single surrogate. For example, a surrogate can be made with more readily available numerical results, after which the surrogate is corrected using a (small) number of experimental data points.

3.2.2. Surrogate Generation

Several methods for surrogate construction exist. All methods begin with the selection of initial sampling points, which is often done using Latin Hypercube Sampling (LHS).
After the selection of the initial points, the surrogate is constructed from the training data: $\left(\mathbf{x}^i, y^i\right)$, where $\mathbf{x}^{(i)}$ is the input vector and $y^{(i)}$ is the corresponding output, using interpolation or regression. The use of either of these methods is determined by the data used to fit the surrogate. Interpolation builds a surrogate that exactly matches the data, while regression minimises the error between a smooth trend function and the training data.
Interpolation can be helpful for highly modal, non-noisy data, as a regression model would smooth out variations in the data. This smoothing of the regression model makes it suitable for noisy data, as it will not attempt to fit the exact data points, unlike an interpolation model, which can reduce unwanted oscillations.

One of the simplest surrogates is a polynomial model, also known as a polynomial response surface model. The general form of which is:

$$\hat{f}\left(\mathbf{x}\right) = w_0 + \sum_{i=1}^{m} w_i \mathbf{x}_i + \sum_{i=1}^{m} \sum_{j \geq i}^{m} w_{ij} \mathbf{x}_i \mathbf{x}_j + \ldots + \epsilon. \tag{3.2}$$

With ϵ the statistical error, $\mathbf{x}_i$, the i-th entry in the m-dimensional predictor and w_0, w_i, w_{ij} are the weight coefficients to be determined in the training process. The weight coefficients can be found by solving a least-squares problem. After this, the trained model can be used to make predictions of the estimated function. Note that this estimator can be used outside of the training data. These points are estimated using extrapolation, which might return inaccurate results.
The approximation method essentially is a Taylor series expansion of $f(\mathbf{x})$, truncated at m terms, suggesting that adding terms will improve the approximation. However, the increased number of terms increases the model's flexibility, which in turn increases the chance of overfitting and an excessively 'snaking' polynomial, thereby reducing generalisation. Herein lies one of the shortcomings of this model, since polynomials can not represent highly nonlinear relations [5], [6].
More advanced surrogate models are available. Two commonly used methods are based on Radial Basis Function (RBF) and Kriging (KRG). These methods are discussed in detail below.

Radial Basis Function

A commonly used surrogate in multiple fields is one based on RBFs; these models show good flexibility, simple structures, relatively few calculations, and high efficiency [5], [10]. A radial function is a function that has the Euclidean distance between the sample point and the point to be measured as the independent variable. Using a superposition of these radial functions as the basis functions results in an RBF model.

With noise-free data, an interpolating RBF model can be used. Such an RBF model is constructed with Equation (3.3), given a set of n sample points with $\mathbf{X} = \left\{\mathbf{x^1}, \dots, \mathbf{x^n}\right\}^T$ and corresponding output $\mathbf{Y} = \left\{y^1, \dots, y^n\right\}^T$.

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} w_i \cdot \phi\left(r^i\right) = \mathbf{W^T \Phi}. \tag{3.3}$$

In this equation, $\mathbf{W}$ are the weight coefficients and $\phi(r)$ is a radial function. r^i is the Euclidean distance between point $\mathbf{x}$, the point to be measured, and the sample point, $\mathbf{x^i}$ or $||\mathbf{x} - \mathbf{x^i}||$. Since the model uses interpolation, it holds that the prediction at a sample point is equal to the sampled value, thus:

$$\hat{f}\left(\mathbf{x}^j\right) = y^j \quad \text{where } j = 1, 2, \dots, n. \tag{3.4}$$

From Equation (3.3) it can be seen that the RBF approximation is linear in the weight coefficients w, but the predictor $\hat{f}$ can express nonlinear responses. Combining Equation (3.3) and Equation (3.4) yields Equation (3.5).

$$\begin{pmatrix} y^1 \\ \vdots \\ y^n \end{pmatrix} = \begin{pmatrix} \phi\left(||\mathbf{x}^1 - \mathbf{x}^1||\right) & \dots & \phi\left(||\mathbf{x}^1 - \mathbf{x}^n||\right) \\ \vdots & \vdots & \vdots \\ \phi\left(||\mathbf{x}^n - \mathbf{x}^1||\right) & \dots & \phi\left(||\mathbf{x}^n - \mathbf{x}^n||\right) \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} \tag{3.5}$$

or

$$\mathbf{Y} = \mathbf{\Phi} \cdot \mathbf{W}. \tag{3.6}$$

The weight coefficients can subsequently be calculated using Equation (3.7), given that the Gram matrix, $\mathbf{\Phi}$ is non-singular [5], [10]:

$$\mathbf{W} = \mathbf{\Phi}^{-1}\mathbf{Y}. \tag{3.7}$$

The behaviour of the approximation depends on the radial basis function chosen; several commonly used functions are tabulated in Table 3.1. In this table, the last four basis functions introduce an additional parameter to be estimated, c, improving the model's generalisation properties. This parameter is usually considered constant for all basis functions. It can, however, also be varied between the basis functions; this is, for example, done when using a KRG basis function (discussed later) [10].

For a multi-quadratic basis function, the estimation will show characteristics of global estimation, while using a Gaussian or inverse multi-quadratic function, the model will show local estimation characteristics caused by the influence of the radial function [5].
It can be shown that, under certain conditions, using a Gaussian or inverse multi-quadratic basis function yields a positive definite Gram matrix. This ensures the weight coefficients can be safely computed using

Table 3.1: Commonly used radial functions in RBF models [5], [10].

Function name	Function
Linear	$\phi(r) = r$
Cubic	$\phi(r) = r^3$
Thin-plate splines function	$\phi(r) = \frac{r}{c^2} \ln\left(\frac{r}{c}\right)$, with $c > 0$
Gaussian function	$\phi(r) = \exp\left(\frac{-r^2}{2c^2}\right)$, with $c > 0$
Multi-quadratic function	$\phi(r) = \sqrt{r^2 + c^2}$, with $c > 0$
Inverse multi-quadratic function	$\phi(r) = \frac{1}{\sqrt{r^2+c^2}}$, with $c > 0$

Cholesky factorisation [10]. As discussed in [11], [12], [13], these conditions are the following: First, it is required that $c > 0$ as it is shown that with this condition the Gaussian and inverse multi-quadratic kernel functions are strictly positive definite on $\mathbb{R}^d$ for any dimension $d \geq 1$. Secondly, it is required that all points x^i are distinct. Both conditions result in a full rank $\mathbf{\Phi}$, allowing the use of Cholesky factorisation.

The last condition thus requires the sampling points in $\mathbf{X}$ to be distinct and not in very close proximity, as this can cause ill-conditioning, which will result in failing to compute the weight coefficients. This can become a concern for clusters of infill points in areas of interest in optimisation. In general, this does not pose a problem in the set of initial sample points, if a space-filling sampling plan is used [10].

This is one of the reasons the Gaussian function is the most commonly used basis function in RBF models [5], and results in the following RBF model:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} w_i \exp\left(\frac{-\left(r^i\right)^2}{c^2}\right), \text{ with } c > 0. \tag{3.8}$$

Kriging

Another commonly used surrogate modelling technique is KRG, also called Gaussian process (interpolation) [6]. This technique exhibits a similarity with RBF models, specifically that under certain conditions, KRG can be shown to be an RBF model with a basis function (as given in Equation (3.9)) similar to the Gaussian function (listed in Table 3.1).

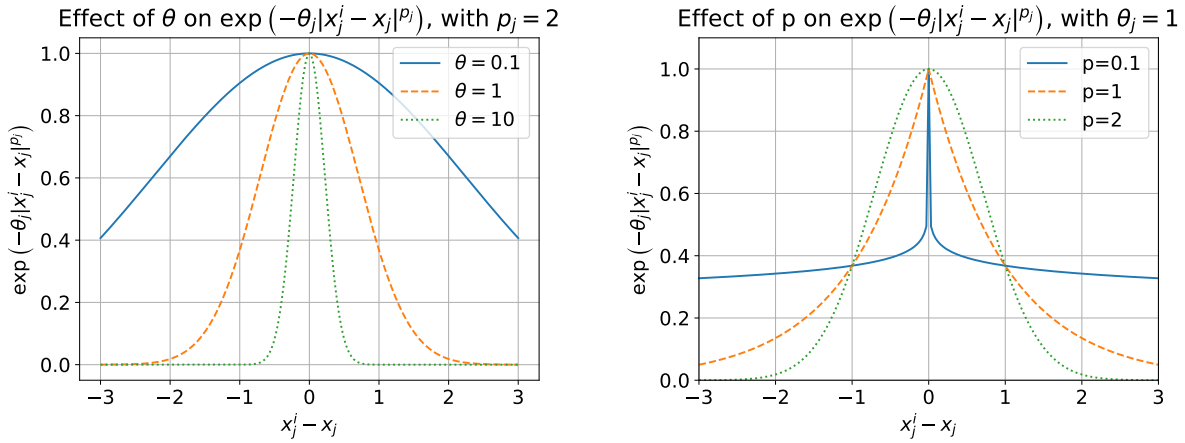
$$\phi\left(r^i\right) = \exp\left(-\sum_{j=1}^{k} \theta_j |\mathbf{x}_j^i - \mathbf{x}_j|^{p_j}\right). \tag{3.9}$$

The first difference between the basis functions is that the KRG basis function allows the width of the basis function to vary. This is controlled by the vector $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_k\}^T$, this parameter is fixed in the Gaussian basis function at $\frac{1}{c^2}$. This parameter defines how far the influence of the sample point extends. A low value indicates that all points have a strong correlation, whereas a large value suggests that the closer sample points will have a greater effect. This effect is visualised in Figure 3.2a. Additionally, examining the θ values for all parameters can provide insight into which parameters are the most important in a design space. This is demonstrated in the example in [10], where a set of measurements of a car's acceleration is introduced, with three variables.

- x_1: The colour of the car.
- x_2: The engine position (for, mid, aft).
- x_3: The engine size.

This results in $\theta_1 < \theta_2 < \theta_3$, which is intuitively correct, since the car's colour does not affect the acceleration and therefore has a small influence factor. The engine position has a greater effect on acceleration, but not as significant as the engine size.
The other parameter that is different in the KRG basis function from the Gaussian basis function is the

variable exponent p_j, which typically is $p_j \in [1, 2]$. The parameter affects the smoothness of the function. With a value of 2, which is the fixed value for a Gaussian basis function, the function is smooth. The function gets increasingly less smooth for lower values of p_j, resulting in almost no correlation between two points for a low value of p. This effect is visualised in Figure 3.2b [6], [10].



(a) The effect of the variable θ_j on the value of the KRG function.

(b) The effect of the variable p_j on the value of the KRG function.

Figure 3.2: The effect of the variable θ_j and p_j in the KRG basis function.

The KRG surrogate method has been popular in engineering and optimisation for several reasons. First, the model can be an interpolating model, which is required when using deterministic simulations. Secondly, the method can be adapted to use noisy data by introducing an error term, thereby effectively using a regression model. Next, the method demonstrates good adaptability to most function types, enabling the process to be applied to fit black-box functions whose intrinsic properties are unknown. Another advantage of the method is that it can be used to integrate data from various stages, due to the probabilistic nature of the modelling technique. Lastly, the prediction errors can be estimated at prediction points [5].
Determining the KRG predictor starts with the following KRG statistical model [6], [10]:

$$\hat{f}(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}), \quad \text{where } Z(\mathbf{x}) \sim \mathcal{N}\left(0, \sigma^2\right). \tag{3.10}$$

In this formula, $\mu(\mathbf{x})$ captures some of the function's behaviour. This can be a function of $\mathbf{x}$. However, most models consider this to be constant, as the random variable, $Z(\mathbf{x})$, can capture the function's behaviour. The exact form of this error term, $Z(\mathbf{x})$, is unknown. However, a reasonable assumption can be made about it.
Given two points, $\mathbf{x}^i$ and $\mathbf{x}^j$, with corresponding $Z\left(\mathbf{x}^i\right)$ and $Z\left(\mathbf{x}^j\right)$. It is assumed that the correlation between $Z\left(\mathbf{x}^i\right)$ and $Z\left(\mathbf{x}^j\right)$ is a function of the distance between the two points. Since, intuitively if $\mathbf{x}$ is close to $\mathbf{x}^j$ it is expected that $Z\left(\mathbf{x}^i\right)$ is close to $Z\left(\mathbf{x}^j\right)$. This correlation is assumed to be a kernel function, of which the most commonly used function is the basis function Equation (3.9), resulting in the following kernel, which models continuous functions:

$$K\left(\mathbf{x}^i, \mathbf{x}^j\right) = \text{corr}\left(Z\left(\mathbf{x}^i\right), Z\left(\mathbf{x}^j\right)\right) = \exp\left(-\sum_{l=1}^{n_d} \theta_l |\mathbf{x}_l^i - \mathbf{x}_l^j|^p\right). \tag{3.11}$$

The next step in determining the KRG estimator is to determine the statistical model parameters, μ, σ^2, $\boldsymbol{\theta}$, and $\mathbf{p}$. This is done using the following likelihood function:

$$L(\mu, \sigma, \boldsymbol{\theta}, \mathbf{p}) = \frac{1}{(2\pi)^{\frac{n_s}{2}} \sigma^{n_s} |\boldsymbol{\Psi}|^{\frac{1}{2}}} \exp\left[-\frac{(\mathbf{Y} - \mathbf{e}\mu)^T \boldsymbol{\Psi}^{-1} (\mathbf{Y} - \mathbf{e}\mu)}{2\sigma^2}\right]. \tag{3.12}$$

In which $\boldsymbol{\Psi}$ is the $n \times n$ correlation matrix of all the data using the kernel function and $\mathbf{Y}$ are the function values of the sample points. The parameters need to be determined so that they maximise this likelihood

function. μ and σ can be determined analytically, by first taking the natural logarithm of Equation (3.12), resulting in:

$$\ln L = -\frac{n}{2}\ln 2\pi - \frac{n}{2}\ln \sigma^2 - \frac{1}{2}\ln |\mathbf{\Psi}| - \frac{(\mathbf{Y}-\mathbf{e}\mu)^T \mathbf{\Psi}^{-1}(\mathbf{Y}-\mathbf{e}\mu)}{2\sigma^2}. \tag{3.13}$$

By setting the derivates with respect to μ and σ to 0, results in the following optimal values:

$$\mu^* = \frac{\mathbf{e}^T\mathbf{\Psi}^{-1}\mathbf{Y}}{\mathbf{e}^T\mathbf{\Psi}^{-1}\mathbf{e}}, \tag{3.14}$$

$$\left(\sigma^*\right)^2 = \frac{\left(\mathbf{Y}-\mathbf{e}\mu^*\right)^T\mathbf{\Psi}^{-1}\left(\mathbf{Y}-\mathbf{e}\mu^*\right)}{n_s}. \tag{3.15}$$

Substituting these optimal values back in Equation (3.13) results in the concentrated likelihood function Equation (3.16).

$$\ln(L) - \frac{n}{2}\ln\left(\left(\sigma^*\right)^2\right) - \frac{1}{2}\ln|\mathbf{\Psi}|. \tag{3.16}$$

This concentrated likelihood function depends only on the kernel functions, thus only on the unknown values θ and $\mathbf{p}$. However, the values for which Equation (3.16) is maximised can not be determined analytically, thus, they are determined using a numerical optimisation technique. Typically, the best values are determined using a global search method, such as a genetic algorithm or simulated annealing. As illustrated in Figure 3.2a, the difference in the plot for $\theta = 0.1$ and $\theta = 1$ is similar to the difference for $\theta = 1$ and $\theta = 10$, thus often the search for θ^* is done using a logarithmic scale.

The last part of the KRG surrogate method determines the KRG predictor, which is used to make predictions at new points. There are many ways to derive this predictor, one of which involves finding the function value $\hat{y}_p$ at the prediction point $\mathbf{x}_p$ that is most consistent with the behaviour of the function depicted by the KRG model. This is done by first adding the artificial point $\left(\mathbf{x}_p, \hat{y}_p\right)$ to the training data. The resulting training data is $\bar{\mathbf{Y}} = \left(\mathbf{Y}, \hat{y_p}\right)$. This turns the likelihood function with the fitted KRG parameters into a function of $\hat{y}_p$. Therefore, the value of $\hat{y}_p$ that maximises this likelihood is the prediction for the function. This optimisation process has a closed-form solution, whose corresponding formula is the KRG predictor.

The derivation starts by defining a vector of correlations between the observed data and the new prediction and an augmented correlation matrix, shown in Equation (3.17) and Equation (3.18) respectively.

$$\boldsymbol{\psi} = \begin{pmatrix} \text{cor}\left[Z\left(\mathbf{x}^1\right), Z(\mathbf{x})\right] \\ \vdots \\ \text{cor}\left[Z\left(\mathbf{x}^n\right), Z(\mathbf{x})\right] \end{pmatrix} = \begin{pmatrix} \psi^1 \\ \vdots \\ \psi^n \end{pmatrix}, \tag{3.17}$$

$$\tilde{\mathbf{\Psi}} = \begin{bmatrix} \mathbf{\Psi} & \boldsymbol{\psi} \\ \boldsymbol{\psi}^T & 1 \end{bmatrix}. \tag{3.18}$$

The last entry in the augmented correlation matrix is 1; this is a continuation of the leading diagonal in $\mathbf{\Psi}$, which is the correlation between a point and itself. Thus where $|\mathbf{x}^i - \mathbf{x}^i| = 0$ and $\text{cor}\left[Z\left(\mathbf{x}^i\right), Z\left(\mathbf{x}^i\right)\right] = 1$. Using the augmented vectors, the ln-likelihood is Equation (3.19). In which $\mathbf{e}\mu$ represents the expected value of the random variables $Z^1, \ldots, Z^2$, which is equal to $[1, \ldots, 1]^T\mu$.

$$\ln(L) = -\frac{n_s}{2}\ln(2\pi) - \frac{n_s}{2}\ln\left(\sigma^{*2}\right) - \frac{1}{2}\ln|\tilde{\mathbf{\Psi}}| - \frac{\left(\bar{\mathbf{Y}}-\mathbf{e}\mu^*\right)^T\tilde{\mathbf{\Psi}}^{-1}\left(\bar{\mathbf{Y}}-\mathbf{e}\mu^*\right)}{2\sigma^{*2}}. \tag{3.19}$$

Since only the last term is dependent on $\hat{y}_p$, the other parts can be omitted, leading to the following maximisation problem:

$$\max_{\hat{y}_p} \quad \ln(L) = -\frac{\left(\bar{\mathbf{Y}}-\mathbf{e}\mu^*\right)^T\tilde{\mathbf{\Psi}}\left(\bar{\mathbf{Y}}-\mathbf{e}\mu^*\right)}{2\sigma^{*2}}. \tag{3.20}$$

Which can be solved analytically, resulting in the mean value of the KRG prediction;

$$\hat{f}(\mathbf{x}) = \mu^* + \boldsymbol{\psi}^T\mathbf{\Psi}^{-1}\left(\bar{\mathbf{Y}}-\mathbf{e}\mu^*\right). \tag{3.21}$$

With a mean square error of;

$$\hat{s} = \sigma^{*2}\left[1 - \boldsymbol{\psi}^T\boldsymbol{\Psi}^{-1}\boldsymbol{\psi} + \frac{\left(1 - \boldsymbol{\psi}^T\boldsymbol{\Psi}^{-1}\mathbf{e}\right)^2}{\mathbf{e}^T\boldsymbol{\Psi}^{-1}\mathbf{e}}\right]. \tag{3.22}$$

The KRG algorithm discussed is an interpolating algorithm; this can be adapted to a regression algorithm by adapting the correlation matrix as follows;

$$\boldsymbol{\Psi}_{reg} = \boldsymbol{\Psi} + \tau\mathbf{I} \quad \text{with } \tau > 0. \tag{3.23}$$

This change introduces a constant along the diagonal, resulting in a model that does not perfectly correlate with the sample points, requiring an additional parameter to be estimated. This change is often adopted in interpolatory KRG models, where τ is set to near machine precision to ensure the correlation matrix is invertible.

The accuracy of the prediction can be improved by including the gradients of the training data; this approach is known as Gradient-Enhanced Kriging (GEK). The methodology is essentially the same as previously discussed, but the observed outputs have been altered. This vector now includes the gradients of the sample points, resulting in Equation (3.24), with a length of $n_s + n_s n_d$, in which n_d is the dimension of $\mathbf{x}$.

$$\mathbf{y}_{\text{GEK}} = \begin{bmatrix} y_1 \\ \vdots \\ y_{n_s} \\ \nabla y_1 \\ \vdots \\ \nabla y_{n_s} \end{bmatrix}. \tag{3.24}$$

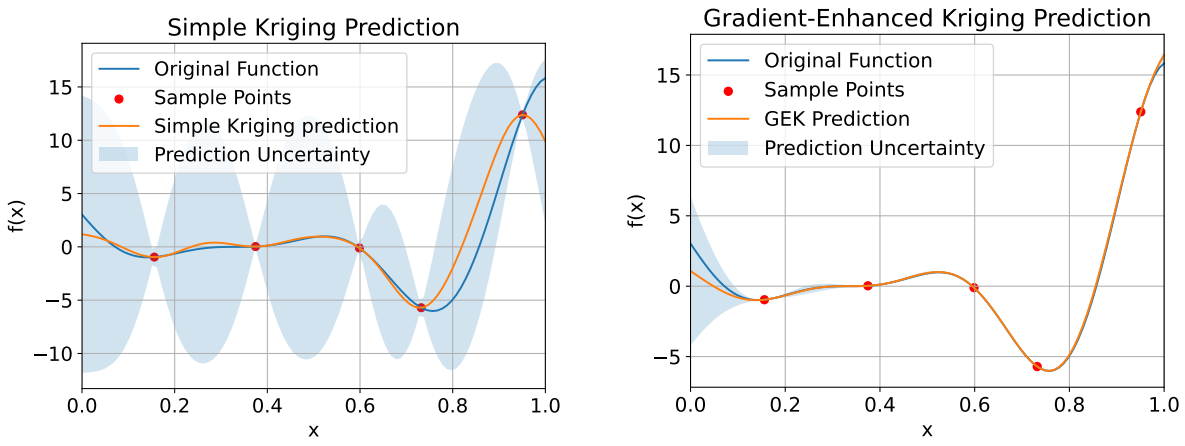
In addition, the mean vector $\mathbf{e}\mu$ needs to be adapted to include the expected values of ∇Z^i, which are all equal to 0. This results in $\mathbf{e}_{GEK} = [1, \ldots, 1, 0, \ldots, 0]$, with the first n_s entries equal to 1 and the remaining entries set to 0.
Lastly, the correlation matrix $\boldsymbol{\psi}$ and augmented correlation matrix $\boldsymbol{\Psi}$ are adapted to include the correlations between function values and gradients, between gradients and function values, and between gradients, as given in Equation (3.25). These adaptations also reveal the downside of GEK compared to simple KRG: the correlation matrix grows more rapidly with an increase in the problem dimension and/or the number of sample points.

$$\begin{aligned}
\text{cor}\left(Z\left(\mathbf{x}^i\right), Z\left(\mathbf{x}^j\right)\right) &= Z\left(\mathbf{x}^i, \mathbf{x}^j\right), \\
\text{cor}\left(Z\left(\mathbf{x}^i\right), \frac{\partial Z\left(\mathbf{x}^j\right)}{\partial \mathbf{x}_l}\right) &= \frac{\partial Z\left(\mathbf{x}^i, \mathbf{x}^j\right)}{\partial \mathbf{x}_l^j}, \\
\text{cor}\left(\frac{\partial Z\left(\mathbf{x}^i\right)}{\partial \mathbf{x}_l}, Z\left(\mathbf{x}^j\right)\right) &= \frac{\partial Z\left(\mathbf{x}^i, \mathbf{x}^j\right)}{\partial \mathbf{x}_l^i}, \\
\text{cor}\left(\frac{\partial Z\left(\mathbf{x}^i\right)}{\partial \mathbf{x}_l}, \frac{\partial Z\left(\mathbf{x}^j\right)}{\partial \mathbf{x}_l}\right) &= \frac{\partial^2 Z\left(\mathbf{x}^i, \mathbf{x}^j\right)}{\partial \mathbf{x}_l^i \partial \mathbf{x}_l^j}.
\end{aligned} \tag{3.25}$$

Using the updated variables Equation (3.21) and Equation (3.22) can be used as the predictors. This algorithm for GEK is discussed in more detail in [6].
An example of the resulting predictions is illustrated in Figure 3.3; in this figure, the improvement of GEK is seen compared to simple KRG by the prediction of the function as well as the reduced uncertainty of the predictions.

3.2.3. Optimisation and Infill Strategies

Once the initial surrogate is generated, the next steps in the optimisation process outlined in Figure 3.1 are the optimisation and updating of the surrogate. This optimisation is not performed on the objective function f, but on the surrogate $\hat{f}$. During the design process, new points are commonly added to the



(a) The resulting simple KRG prediction.

(b) The resulting GEK prediction.

Figure 3.3: The difference in prediction between simple KRG and GEK.

surrogate; these points are referred to as infill points, and with the addition of the latest point(s), the surrogate is updated. The selection of these infill points is based on an infill strategy that can utilise one of two approaches: exploitation and exploration.
Exploitation uses the current surrogate's optimum as the infill point. This point can be found using any optimisation algorithm. This approach locally updates the surrogate at the region of interest, where the current optimum is found. However, this region may not be the true optimum for highly modal functions, which could result in the found optimum being a local one.
The other approach is exploration, which focuses on discovering new regions of interest. A combination of both methods is often used in combination with KRG, where the uncertainty of the prediction can be estimated [2], [6], [14].

Probability of Improvement

An infill strategy that utilises the uncertainty provided by a KRG surrogate is Probability of Improvement (PoI). This infill strategy gives the probability that a value is improved upon, given the surrogate estimate and uncertainty. This value to improve upon has an offset from the current best value, $T < f_{min}$. The PoI is defined as:

$$\text{PoI} = \Phi\left(\frac{T - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right). \tag{3.26}$$

In Equation (3.26) Φ is the normal cumulative distribution function.
The behaviour of this infill strategy is heavily dependent on the choice of T. If this improvement is too small, the search will be highly local to the current best optimum. Likewise, if the improvement is set to a large value, the search will be mostly explorative. Two solutions are proposed; the first is to evaluate the PoI infill strategy for several values of T during every iteration and select the best infill point from the multiple points proposed. The second solution led to the Efficient Global Optimisation (EGO) algorithm with an Expected Improvement (EI) infill strategy [15], [16], [17].

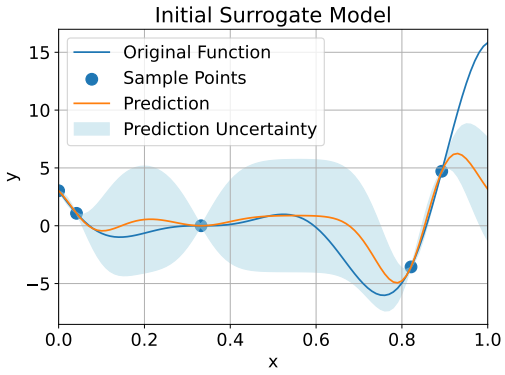
Efficient Global Optimisation with Expected Improvement

EGO with EI is a Bayesian optimisation approach that balances exploitation and exploration, requiring the surrogate to provide estimates for uncertainty. This approach aims to sample new points at the point where the expected improvement is highest. This does not mean that the point is sampled at the point where the uncertainty is highest, as this might be far from the region of interest. This thus results in Equation (3.27) for a probability distribution with the optimum at $f^* = f\left(\mathbf{x}^*\right)$.

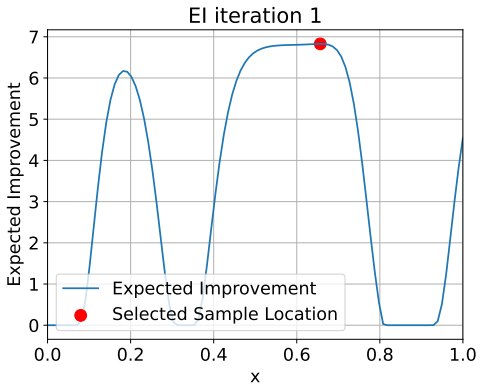
$$\max \text{EI}(\mathbf{x}) = \mathbb{E}\left(\max\left(f^* - f(\mathbf{x}), 0\right)\right). \tag{3.27}$$

This can be found analytically for a KRG surrogate using Equation (3.28), with Φ and ϕ the cumulative distribution function and probability density function and μ_f and $\hat{s}$ found from the surrogate [5], [6]. An example is shown in Figure 3.4.

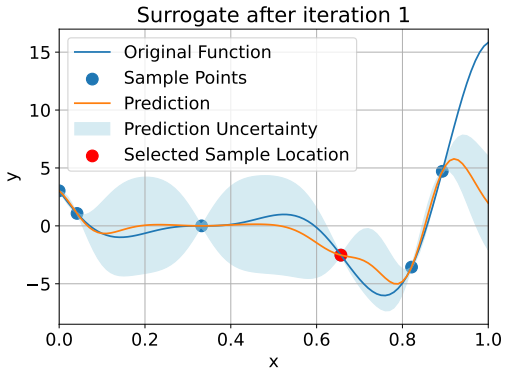
$$\max \mathrm{EI}(\mathbf{x}) = \left(f^* - \hat{f}(\mathbf{x})\right) \Phi\left(\frac{f^* - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right) + \hat{s}(\mathbf{x})\, \phi\left(\frac{f^* - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right). \tag{3.28}$$



(a) Initial surrogate.



(b) Corresponding EI distribution.



(c) Updated surrogate with selected infill point.

Figure 3.4: Example of one iteration of efficient global optimisation and expected improvement.

This implementation is valid for an unconstrained optimisation problem and can be extended to constrained problems. If the constraints do not rely on expensive calculations, they can be added to the maximisation problem for the EI, resulting in:

$$\begin{aligned} &\max \mathrm{EI}(\mathbf{x}), \\ &\text{s.t. } g_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, m. \end{aligned} \tag{3.29}$$

With $g_j(\mathbf{x})$ the constraints of the original optimisation problem.

Another method suggested by Schonlau [18] is the Constrained Expected Improvement (CEI) [5]. This includes a Probability of Feasibility (PoF) (Equation (3.30), in which $\hat{g}_j$ is the surrogate of the j-th constraint, e_j is the predicted error and the constraint function is considered a random variable; $G_j(\mathbf{x}) \sim \mathcal{N}\left(\hat{g}_j(\mathbf{x}), e_j(\mathbf{x})\right)$, resulting in the CEI to maximise; Equation (3.31).

$$\mathrm{PoF} = \prod_{j=1}^{m} \Pr\left(G_j(\mathbf{x}) \leq 0\right) = \prod_{j=1}^{m} \Phi\left(\frac{-\hat{g}_j(\mathbf{x})}{e_j(\mathbf{x})}\right), \tag{3.30}$$

$$\max \mathrm{CEI}(\mathbf{x}) = \mathrm{EI}(\mathbf{x}) \times \prod_{j=1}^{m} \Phi\left(\frac{-\hat{g}_j(\mathbf{x})}{e_j(\mathbf{x})}\right). \tag{3.31}$$

The first part of this equation is the same as Equation (3.28) and will thus favour a high EI, whereas the second term favours points which have a high PoF. Multiplying these terms results in an infill approach which tends to select an infill point that satisfies all constraints and has a large expected improvement.

Generalised Expected Improvement

Schonlau argued that the previously discussed EI criterion placed too much emphasis on the local search near the optimum of the surrogate. This would be the case when the correlation parameters were poorly estimated. To combat this he introduced a version of EI that searches more globally in [18]; Generalised Expected Improvement (GEI). This version includes an additional parameter g, which determines how much the infill strategy tends to search globally, with a larger g, resulting in more emphasis on global search. This resulted in the GEI as defined in (3.32).

$$\mathrm{EI}(\mathbf{x}) = (\hat{s}(\mathbf{x}))^g \sum_{k=0}^{g} (-1)^k \binom{g}{k} \left(\frac{f^* - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})} \right)^{g-k} T_k. \tag{3.32}$$

With T_k from the recursive equation;

$$T_k = -\phi \left(\frac{f^* - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})} \right) \left[\frac{f^* - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})} \right]^{k-1} + (k-1) T_{k-2}, \tag{3.33}$$

with starting $T_0 = \Phi\left(\frac{f^* - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right)$ and $T_1 = \phi\left(\frac{f^* - \hat{f}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right)$. The special case where $g = 1$, the GEI simplifies to the EI criterion. For $g = 2$, the function can be simplified to:

$$\mathrm{GEI}_{g=2} = [E(I)]^2 + VAR(I). \tag{3.34}$$

This is a monotone transformation of the EI infill criterion and the variance of the improvement. This variance tends to be larger the further away from a sample point it is calculated, thus representing a global component. The selection of a new sample point using (3.35) is thus a tradeoff between a small improvement with a large probability (exploitation) and a large improvement with a small probability (exploration). By changing the g, this relative importance is altered.

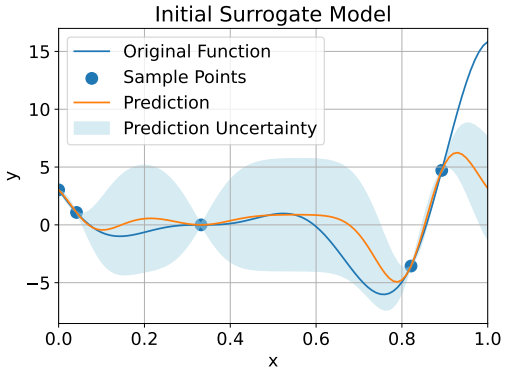
$$\begin{aligned} &\max \mathrm{GEI}(\mathbf{x}), \\ &\quad \text{s.t. } g_j(\mathbf{x}) \leq 0, \quad j = 1, \ldots, m. \end{aligned} \tag{3.35}$$

Lower Confidence Bound

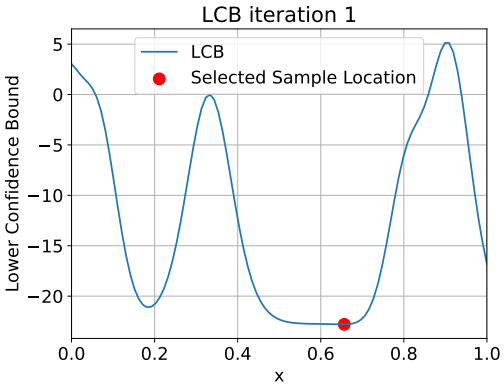
Another infill strategy, which balances exploitation and exploration, is the Lower Confidence Bound (LCB) criterion in Equation (3.36).

$$\min \mathrm{LCB}(\mathbf{x}) = \hat{y}(\mathbf{x}) - b\hat{s}(\mathbf{x}). \tag{3.36}$$

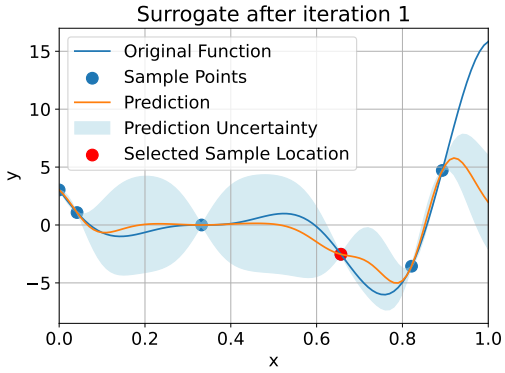
This method requires user input to define the equilibrium constant b, which determines the relative importance of exploitation. If this parameter is chosen as 0, the infill point will be a pure exploitation point. In contrast, if b is chosen to be $b \to \infty$, the point will be a pure exploration point. $\hat{y}(\mathbf{x})$ is the response value from the KRG surrogate and $\hat{s}(\mathbf{x})$ is the predicted standard deviation [5], [6]. An example of the infill strategy is displayed in Figure 3.5.



(a) Initial surrogate.



(b) Corresponding LCB distribution for $b = 1$.



(c) Updated surrogate with selected infill point for $b = 1$.

Figure 3.5: Example of one iteration of efficient global optimisation and lower confidence bound.

3.3. Multi-Fidelity Surrogate

For most engineering simulations, several fidelity levels exist, for example, a CFD simulation compared to a panel method or a simulation using a fine grid versus a coarse grid. By combining these fidelities into one surrogate, the computational cost of the optimisation can be further reduced.
There are several ways to generate MF surrogates; these can be broadly divided into three categories. First, adaptation: this strategy enhances the Low Fidelity Model (LFM) with High Fidelity (HF) samples throughout the optimisation. This correction can be additive, multiplicative or a combination of the two, resulting in the following general form for a two-fidelity model:

$$f_{\text{HFM}}(\mathbf{x}) = w\rho(\mathbf{x})\, y_{\text{LFM}}(\mathbf{x}) + (1 - w(\mathbf{x}))\left[y_{\text{LFM}}(\mathbf{x}) + \delta(\mathbf{x})\right]. \tag{3.37}$$

In this correction, w is the weighting factor, which can be adjusted throughout the optimisation. $\rho(\mathbf{x})$ is an adjustment parameter, which can vary throughout the design space. The last parameters a, b, c are unknown constants, and $\delta(\mathbf{x})$ is a discrepancy function. For a weight factor $w = 0$, this correction will be additive, for $w = 1$ and $c = 0$ the correction will be multiplicative and for either $w \notin \{0, 1\}$ or $w = 1$ and $c \neq 0$ it will be a combination [2], [14], [19].
This method can be adapted for the use of multiple LFM, using weighted average models. For n LFM, this will result in Equation (3.38) [2].

$$f_{\text{HFM}}(\mathbf{x}) = \sum_{i=1}^{n} \rho_i(\mathbf{x})\, f_{\text{LFM}_i}(\mathbf{x}) + \delta(\mathbf{x}). \tag{3.38}$$

A further distinction can be made within the adaptive MF surrogates based on how $\rho(\mathbf{x})$ and $w(\mathbf{x})$ are modelled: deterministic or probabilistic. The deterministic approach models functions as linear combinations of a finite number of basis functions, such as radial basis functions. The probabilistic approach models the functions using a stochastic process, often a Gaussian process [2].

The next MF surrogate strategy is based on the fusion of the HF and Low Fidelity (LF) data. This approach combines the LF and HF samples, by combining a small number of HF evaluations to obtain an unbiased estimator, with a larger number of LF evaluations to get a low variance estimator [14]. An example of this approach is Co-Kriging (Co-KRG), which is discussed in detail later.
The last category in this categorisation of MF surrogate methods is filtering. This approach runs the LF function, and based on the output of this LF function, a filter is invoked. This filter can be of several natures; for example, it might invoke the running of the HF function if the LF sample is deemed uncertain or if a new sample point is found that meets specific criteria [14].

3.3.1. Co-Kriging

A commonly used MF surrogate technique is Co-KRG. This technique is a form of KRG that correlates multiple sets of data. Therefore, the method is similar to simple KRG as discussed in subsection 3.2.2, with several changes to accommodate the multiple sets of data. The first change is to concatenate the HF and LF sample points, resulting in;

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_{\mathrm{LF}} \\ \mathbf{X}_{\mathrm{HF}} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{\mathrm{LF}}^{1} \\ \vdots \\ \mathbf{x}_{\mathrm{LF}}^{n_{\mathrm{LF}}} \\ \mathbf{x}_{\mathrm{HF}}^{1} \\ \vdots \\ \mathbf{x}_{\mathrm{HF}}^{n_{\mathrm{HF}}} \end{pmatrix}, \tag{3.39}$$

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}\right) \\ \mathbf{Y}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{HF}}\right) \end{pmatrix} = \begin{pmatrix} \mathbf{Y}_{\mathrm{LF}}\left(\mathbf{x}_{\mathrm{LF}}^{1}\right) \\ \vdots \\ \mathbf{Y}_{\mathrm{LF}}\left(\mathbf{x}_{\mathrm{LF}}^{n_{\mathrm{LF}}}\right) \\ \mathbf{Y}_{\mathrm{HF}}\left(\mathbf{x}_{\mathrm{HF}}^{1}\right) \\ \vdots \\ \mathbf{Y}_{\mathrm{HF}}\left(\mathbf{x}_{\mathrm{HF}}^{n_{\mathrm{HF}}}\right) \end{pmatrix}. \tag{3.40}$$

The following steps in the method use the auto-regressive model of Kennedy and O'Hagan [5], [10], [20]. This implies that the HF data is assumed to be true; thus, nothing can be learned at a sample point $\mathbf{x}$ from the LF simulation if this sample point is generated with the HF simulation. Thus, the inaccuracies are solely in the LF simulation. This results in an approximation of the HF simulation with LF data.

$$Z_{\mathrm{HF}}(\mathbf{x}) = \rho Z_{\mathrm{LF}}(\mathbf{x}) + Z_d(\mathbf{x}), \tag{3.41}$$

with $Z_d(\mathbf{x})$ the difference between $Z_{\mathrm{HF}}(\mathbf{x})$ and $\rho Z_{\mathrm{LF}}(\mathbf{x})$.
The next change is the new method used to construct the covariance matrix.

$$\begin{aligned} \mathrm{COV}\left\{\mathbf{Y}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}\right), \mathbf{Y}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}\right)\right\} = \mathrm{COV}\left\{\mathbf{Z}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}\right), \mathbf{Z}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}\right)\right\} = \\ \sigma_{\mathrm{LF}}^2 \boldsymbol{\Psi}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}, \mathbf{X}_{\mathrm{LF}}\right), \end{aligned} \tag{3.42}$$

$$\begin{aligned} \mathrm{COV}\left\{\mathbf{Y}_{\mathrm{HF}}\left(\mathbf{X}_{\mathrm{HF}}\right), \mathbf{Y}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}\right)\right\} = \mathrm{COV}\left\{\rho \mathbf{Z}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}\right) + Z_d\left(\mathbf{X}_{\mathrm{LF}}\right), \mathbf{Z}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}\right)\right\} = \\ \rho\sigma_{\mathrm{LF}}^2 \boldsymbol{\Psi}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{LF}}, \mathbf{X}_{\mathrm{HF}}\right), \end{aligned} \tag{3.43}$$

$$\begin{aligned} \mathrm{COV}\left\{\mathbf{Y}_{\mathrm{HF}}\left(\mathbf{X}_{\mathrm{HF}}\right), \mathbf{Y}_{\mathrm{HF}}\left(\mathbf{X}_{\mathrm{HF}}\right)\right\} = \mathrm{COV}\left\{\rho \mathbf{Z}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{HF}}\right) + Z_d\left(\mathbf{X}_{\mathrm{HF}}\right), \rho \mathbf{Z}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{HF}}\right) + Z_d\left(\mathbf{X}_{\mathrm{HF}}\right)\right\} = \\ \rho^2\sigma_{\mathrm{LF}}^2 \boldsymbol{\Psi}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{HF}}, \mathbf{X}_{\mathrm{HF}}\right) + \sigma_d^2 \Psi_d\left(\mathbf{X}_{\mathrm{HF}}, \mathbf{X}_{\mathrm{HF}}\right). \end{aligned} \tag{3.44}$$

The correlations are of the same form as Equation (3.11); however, there are two correlations ψ_{lf} and ψ_d. These two correlations require more parameters to be estimated: $\boldsymbol{\theta}$ and $\mathbf{p}$ for both the LF data and the discrepancy between LF and HF, as well as the adjustment parameter ρ. The ln-likelihood, similar to Equation (3.15) can be used to find the parameters for the LF data as well as the μ_{LF} and σ_{LF}^2. This is essentially the same approach for the LF data as used in the derivation of simple KRG. The parameters for the difference ($\mu_d, \sigma_d^2, \boldsymbol{\theta}_d, \mathbf{p}_d$ and ρ) are estimated as follows, starting with;

$$\mathbf{d} = \mathbf{y}_{\mathrm{HF}} - \rho \mathbf{y}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{HF}}\right), \tag{3.45}$$

In which $\mathbf{y}_{\text{LF}}\left(\mathbf{X}_{\text{HF}}\right)$ are the values of the LF dataset sample points, which are also sampled in the HF simulation. If no LF samples are sampled at the location of HF samples, ρ can be estimated using KRG estimates; $\hat{\mathbf{y}}_{\text{LF}}\left(\mathbf{X}_{\text{HF}}\right)$, using $\hat{\theta}_{\text{LF}}$ and $\hat{\mathbf{p}}_{\text{LF}}$.
The method for determining the predictor is the same as the approach used in simple KRG, utilising the previously modified variables. This method is described in more detail in [10] and [5]. An example of a Co-KRG surrogate of the Forrester function and its LF function [10], [21] is illustrated in Figure 3.6.

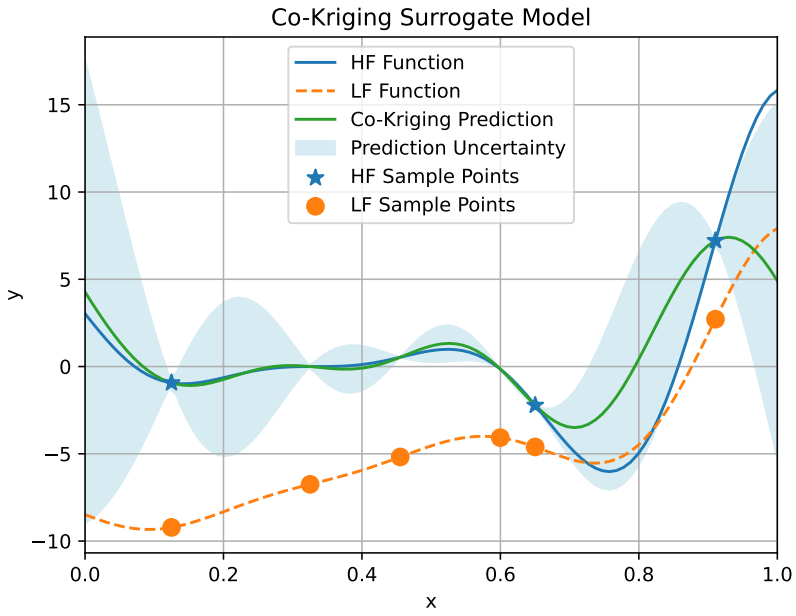


Figure 3.6: Co-KRG surrogate of the Forrester function using three HF and six LF samples.

3.3.2. Hierarchical Kriging

Another form of MF KRG is Hierarchical Kriging (HK) as introduced by Han and Görtz [22]. In this method, an LF KRG model is scaled by a scalar, which together serves as the global trend or mean function. A stationary random process error term Z is added to this mean hierarchically. A similar model exists in the geostatistics community: KRG with external drift. There are two differences: first, HK does not require nested samples, which KRG with external drift (and Co-KRG) does. A nested sample means that for a sample at a fidelity level, all lower fidelity models should include this sample location in their training data. Secondly, KRG with external drift utilises variograms, whereas HK employs correlation functions.
For a two-fidelity HK model, the method works as follows: first, the LFM is created using a KRG algorithm. Next, the HK model of the HF model is determined, starting with the definition of the random process that describes the function:

$$Y(\mathbf{x}) = \beta_0 \hat{y}_{\text{LF}}(\mathbf{x}) + Z(\mathbf{x}). \tag{3.46}$$

In this the LFM, $\hat{y}_{\text{LF}}(\mathbf{x})$, is scaled by an unknown factor β_0. This serves as the mean and the random processes, $Z(\mathbf{x})$ having a zero mean and the following covariance;

$$\text{Cov}\left[Z(\mathbf{x}), Z\left(\mathbf{x}'\right)\right] = \sigma^2 R\left(\mathbf{x}, \mathbf{x}'\right). \tag{3.47}$$

In which σ is the variance and $\mathbf{R}$ is the spatial correlations, depending on the Euclidean distance. Assuming that the HF function can be approximated as a linear response of the HF data leads to the predictor in Equation (3.48). Which can be derived to be Equation (3.49).

$$\hat{y}(\mathbf{x}) = \mathbf{w}^T \mathbf{Y}, \tag{3.48}$$

$$\hat{f}(\mathbf{x}) = \beta_0 \hat{y}_{\text{LF}}(\mathbf{x}) + \mathbf{r}^T(\mathbf{x}) \underbrace{\mathbf{R}^{-1}\left(\mathbf{Y} - \beta_0 \mathbf{F}\right)}_{=:\mathbf{v}_{HK}}. \tag{3.49}$$

Where:

$$\begin{aligned} \mathbf{F} &\coloneqq \left[\hat{y}_{\mathrm{LF}}\left(\mathbf{x}^1\right), \ldots, \hat{y}_{\mathrm{LF}}\left(\mathbf{x}^2\right)\right]^T, & \tilde{\mu} &= \frac{\mu}{2\sigma^2}, \\ \mathbf{R} &\coloneqq \left(R\left(\mathbf{x}^i, \mathbf{x}^j\right)\right)_{i,j} \in \mathbb{R}^{n \times n}, & \mathbf{r} &\coloneqq R\left(\mathbf{x}^i, \mathbf{x}\right)_i \in \mathbb{R}^n, \\ \beta_0 &= \left(\mathbf{F}^T \mathbf{R}^{-1} \mathbf{F}\right)^{-1} \mathbf{F}^T \mathbf{R}^{-1} \mathbf{Y}. \end{aligned} \tag{3.50}$$

The scaling factor β_0 indicates the degree of correlation between the HF and LF data. The vector $\mathbf{V}_{HK}$ is solely dependent on training data, thus to determine the predicted value for a new point only $\hat{y}_{\mathrm{LF}}\left(\mathbf{X}_{\mathrm{new}}\right)$ and $\mathbf{R}^T\left(\mathbf{x}_{\mathrm{new}}\right)$ have to be recalculated.
The uncertainty for a new point can be found using:

$$\hat{s} = \sigma^2 \left\{ 1 - \mathbf{r}^T \mathbf{R}^{-1} \mathbf{r} + \left[\mathbf{r}^T \mathbf{R}^{-1} \mathbf{F} - \hat{y}_{\mathrm{LF}}(\mathbf{x})\right] \left(\mathbf{F}^T \mathbf{R}^{-1} \mathbf{F}\right)^{-1} \left[\mathbf{r}^T \mathbf{R}^{-1} \mathbf{F} - \hat{y}_{\mathrm{LF}}(\mathbf{x})\right]^T \right\}. \tag{3.51}$$

This uncertainty includes the behaviour of the LFM, which improves the determination of the uncertainty compared to traditional KRG. In addition, the HK model does not need to model the cross-correlation between the LF and HF formula, leading to a smaller correlation matrix and model complexity. The hyperparameters of the model are trained in a comparable way to the previously discussed KRG models. An example of an HK surrogate is illustrated in Figure 3.7.
This method can be expanded to include more fidelity levels; a discussion of this and a more detailed derivation can be found in [22].

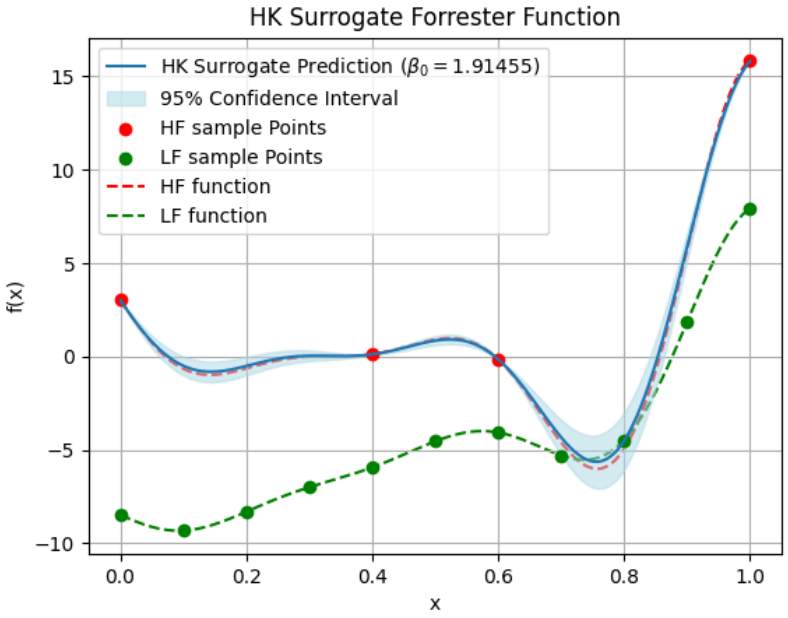


Figure 3.7: HK surrogate of the Forrester function using four HF and eleven LF samples.

3.4. Multi-Fidelity Infill Strategies

As discussed in subsection 3.2.3, surrogates are often improved during the optimisation process according to an infill strategy. The strategies discussed in subsection 3.2.3 can be applied to Single-Fidelity (SF) surrogates. However, MF surrogates require an additional decision to be made; in addition to the location of the new sample, the fidelity level at which to sample has to be determined. Do and Zhang [2] divide these methods into three categories.
The first category includes strategies that do not consider fidelity. These methods determine a new sample location and feed it to one or several pre-determined fidelity levels. The selection of the fidelity level is thus not taken into account when selecting the infill point.
The next category of strategies is the heuristic approach. In this approach, the new sample point location

and fidelity are determined simultaneously. These strategies often include an auxiliary function which considers the computational cost of each fidelity level and how each will affect the accuracy improvement of the surrogate.
The last category is sequential selection. This approach first determines the sample location, after which the fidelity level is selected. The main difference between the heuristic approach and this method is that the choice of fidelity level and sample location is independent, whereas this is not the case for the heuristic approach.
For each category, several infill strategies are introduced. An overview per category is tabulated in Table 3.2. Note that in this table, if the required surrogate is 'any', this refers to any MF surrogate that can return the uncertainty for the High Fidelity Model (HFM).

Table 3.2: Classification of discussed MF infill strategies.

MF infill	Base infill	Surrogate	Fidelity Selection	Additional note
No Fidelity Consideration				
Always HF [23]	Any SF	Any	Always use HF	-
Every n-th HF	Any SF	Any	Use HF for very n-th point, else LF	-
Heuristic approach				
Improvement per fidelity level [24]	SF EI	Nested HK	Maximise surrogate improvement for location and fidelity	Nested, thus addition of HF point requires the addition of LF point
Adapted EI [25]	SF EI	Non-nested MF	EI per fidelity scaled with cost-ratio	Favours HF, LF is selected if the cost-ratio is competitive
Minimisation of further improvement [26]	SF EI	MF	Reduction in max EI scaled with cost ratio	-
Variable fidelity EI [27]	SF EI	HK	Considers uncertainty due to lack of HF or LF sample	-
Sequential approach				
AMEI [28]	SF EI	HK	Based on RMSE between HFM and LFM and threshold for $AMEI_L$	Improves upon Variable fidelity EI by including optimal result of LFM
EIR [29]	SF EI	Any MF	The reduction in EI for one HF sample or n LF samples. With n based on the cost-ratio	The n LF samples are selected using LHS in trust-region around the HF infill location
filter-GEI [30]	SF GEI	Co-KRG	Use LF unless surrogate prediction is above the dynamic filter threshold	The filter threshold is based on the current optimum, the weighted average sample value and the adaptive relative importance of the two.
Two-Step [31]	Any SF	HK	Select location using SF infill, use cheapest sufficiently accurate fidelity	-

Continued on next page

Table 3.2 – continued from previous page

MF infill	Base infill	Surrogate	Fidelity Selection	Additional note
MF LCB [32]	SF LCB	Any MF	Maximum of uncertainty vector	The uncertainty vector is defined as the fidelity uncertainty divided by the cost ratio between the fidelity and HF functions.
MF ACAS [33]	ACAS	Any MF	Maximum of the uncertainty vector	Equal to MF LCB, with different base infill.

3.4.1. No Fidelity Consideration

As stated earlier, infill strategies that do not consider fidelity levels will not determine at which fidelity level to sample based on the infill location. This can result in a computationally more expensive optimisation run. An example of this approach is to include every new sample point as an HF sample, as used by Dong et al. [23]. In addition, Zhang et al. [27] note that this is the only possible strategy when employing the EI strategy as is, or another unadapted infill strategy, for a MF surrogate.
A possible method that has no fidelity consideration based on the sample point is to include an LF sample at every sample location, except every n-th sample point, which is sampled using the HF function.
This method and other methods that fall in this category are relatively simple. They do not require a decision about the fidelity level during the infill point selection process. However, this might lead to the inclusion of HF samples, which could be sampled at LF or to an excessive number of LF samples. Both may increase the computational cost or affect the optimum.

3.4.2. Heuristic Approach

The next of the three categories of approaches considers the selection of the fidelity level when choosing infill points. These methods have been studied in the literature, of which four are introduced below. The four discussed infill strategies are in order:

1. Improvement per fidelity level [24].
2. Adapted EI [25].
3. Minimisation of further improvement [26].
4. Variable fidelity EI [27].

The first method, as introduced by Zhang et al. [24], starts from the SF EI infill strategy. This EI method is adapted to include the improvement of the surrogate per fidelity level. The method utilises an HK surrogate, with nested sample data; thus, the HF sample locations are a subset of the LF samples. Given this assumption, the lower-fidelity models are accurate at the HF sample sites. However, errors in the LF surrogate can propagate to higher-fidelity surrogates. Combining this with the interpolative HK, the model error due to the lack of LF samples can be calculated. This results in an infill strategy that maximises the improvement of the HF surrogate for a sample location and sample fidelity level. Note, however, that due to the nested nature of the nested surrogate used, an LF sample must also be added at the new sample location for every new HF sample.

The adapted EI is introduced by Sacher et al. [25]. This method uses a non-nested surrogate; consequently, the LF and HF samples are not co-located. This method begins by adapting an SF augmented EI infill strategy. This SF augmented EI infill strategy is an EI infill strategy with an added penalisation term that favours areas with high prediction uncertainty. A method is suggested to adapt this approach to include a measure of the induced reduction in predictive variance of the MF surrogate for every fidelity level. This method will favour HF samples, as they usually reduce the uncertainty most effectively. However, since the LF samples are computationally less expensive, these samples can still be beneficial. To account for this, the new infill criterion was scaled by the computational cost of the fidelity level (which is assumed to be constant throughout the design space).

The next discussed infill strategy is the minimisation of further improvement. This method extends the EI infill strategy, and was proposed by Shu et al. [26]. This method aims to minimise further improvement. Further improvement is defined as the EI after a sample is added at a fidelity level. This will thus be zero if an HF sample is added, but remains non-zero for an LF sample, since an error is involved in the LF data. By calculating the reduction in EI between the current and expected EI for the next iteration at every fidelity level and scaling the terms according to the computational cost associated with the model, a new sample location and fidelity are selected. This method thus takes into account both the cost and benefit of each fidelity level.

Lastly, the variable fidelity EI was introduced by Zhang et al. [27]. This study proposes a variable fidelity EI infill strategy for an HK surrogate, which extends the EI infill strategy to consider not only the uncertainty due to the lack of an HF sample but also the uncertainty due to the lack of an LF sample. This additional uncertainty can be analytically derived for an HK surrogate and might not be possible or be difficult for other surrogates, such as Co-KRG.
This proposed infill strategy would tend to include more LF samples if the LF data correlates well with the HF data; on the other hand, if this correlation is worse, more HF samples are added. Additionally, in regions where the LF function is undersampled, HF samples would be added, and vice versa.

3.4.3. Sequential Approach

The last category of MF infill methods is the sequential approach. This approach separates the selection of the infill point's location and fidelity. First, the location is selected, after which the fidelity level is determined separately based on the infill location. Six infill methods using this approach are discussed below; these are in order:

1. Adapative Multi-Fidelity Expected Improvement (AMEI) [28]
2. Expected Improvement Reduction (EIR) [29]
3. filter-GEI [30]
4. Two-Step [31]
5. MF LCB [32]
6. MF Aggregate-Criteria Adaptive Sampling (ACAS) [33]

The first of these methods is the method from Hao et al. [28]. This method builds upon the previously described strategy using EI and HK from Zhang et al. [27]. Starting from the shortcomings identified in this method by Hao et al. [28]; as the formulation of this method does not incorporate the optimal result of the LFM, it yields an unreliable EI of the HFM when an LF is added, which possibly results in more LF samples to be added than necessary. To address this shortcoming, Hao et al. [28] proposed a new infill strategy. This new method decouples the selection of the sample location from the fidelity level, thereby making it a sequential approach.
This method defines the infill criterion (AMEI) discussed in [27] for both the HFM ($AMEI_H$) and LFM ($AMEI_L$) separately. From these formulae, the location of the sample point can be determined. After introducing the Root Mean Square Error (RMSE) of the LFM, $RMSE_L$, which measures the similarity between the LFM and HFM, the fidelity level can be determined. This is determined based on two conditions:
The first condition is based on the $RMSE_L$ compared to $RMSE_\delta$. In which $RMSE_\delta$ is the RMSE of the bridge function. If $RMSE_L > 2RMSE_\delta$, an HF sample is selected as the LF can not approximate well. And if $RMSE_\delta > RMSE_L$, the bridge function is less accurate than the LFM, an HF sample is added to guarantee an improvement in the bridge function and thus in the HFM.
The second is based on the $AMEI_L$. If the $AMEI_L$ exceeds a certain threshold, an LF sample is selected, as the LFM has the potential to improve. This threshold is determined by the difference between the maximum and minimum values of the initial sample.

Another sequential method that uses an adapted EI criterion is proposed by Yang et al. [29]. This method determines the fidelity level of the sample based on a measure of the maximum EIR. This

measure gives an insight into how much the surrogate can be improved after the current iteration.
This infill strategy first selects the new sample location using EI. Following this, the fidelity level is selected. For the fidelity level, there are two options. The first of which is to add an HF sample at the selected infill location. The second option considered is to add n LF in a trust-region around the selected infill location. These LF samples are placed in the trust-region using LHS. The number of LF samples to be added per iteration depends on the cost ratio. To determine whether to include one HF or n LF samples, the reduction of EI is calculated. This is done by adding the hypothetical sample point(s) to the surrogate and recalculating the maximum EI value. The value(s) for the candidate sample point(s) are taken to be the prediction of the current surrogate (either the LFM or HFM) at the point(s). The method that shows the largest reduction in maximal EI for the updated surrogate is selected.

A method based on the GEI introduced by Schonlau [18] is outlined by Guo et al. [30]. Schonlau created GEI after he observed that the classical EI places too much emphasis on the local search near the optima of the surrogate. To combat this, he introduced GEI, which includes a parameter that has to be set by the user. This parameter determines the relative importance of global search and local search, with a higher value resulting in an increased importance for global search. This infill strategy often places an infill point in the neighbourhood of the optima or the area with the maximum prediction uncertainty. Taking this into account GEI was extended to filter-GEI, to account for Multi-Fidelity Surrogate Model (MFSM) [30].
This MF method determines the infill location using GEI and evaluates this sample using the LF simulation, unless the new sample location has a better estimated objective function value than the filter threshold. This filter threshold depends on the current best optimal solution, a weighted average function value over all training samples and an adaptive weight coefficient that weighs the relative importance of the other two parameters.
This weight coefficient is based on the correlation between the HFM and LFM. If this correlation is high, the infill strategy will use HF samples focused on the local exploitation. On the other hand, if the correlation is low, it will lead to HF samples being sampled in a larger area, avoiding the misleading of the LF simulations.

A method that can use any SF infill strategy and more than two levels of fidelity is discussed by Garbo et al. [31]. This method first determines the location of the next infill point using an SF infill strategy on the highest fidelity surrogate model. In the paper, they use an EI with PoF. The next step is to determine which fidelity levels are accurate enough to sample at the next infill location. The last step is to sample the new sample point using the fastest fidelity level in the set of accurate enough fidelity levels.
Several ways of determining if a sample is sufficiently accurate exist and can be used; the Jensen-Shannon distance (JSd) metric is used by Garbo et al. [31].
The Jensen-Shannon Divergence (JSD) quantifies how similar two probability distributions are [34]. It is constructed from the Kullback-Leibler (KL) divergence, which quantifies how much a distribution P differs from a reference Q, using Equation (3.52) [35].

$$D_{\mathrm{KL}}(P\|Q) = \int P(x)\, \log_2 \frac{P(x)}{Q(x)}\, \mathrm{d}x. \tag{3.52}$$

This KL divergence has two shortcomings, which are resolved by JSD. The first of which is that the divergence is not symmetric ($D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$). The second shortcoming is that the divergence measure is undefined whenever Q assigns zero probability to an event that P does not. Both are solved in JSD by measuring how far each distribution deviates from their mixture $M = \frac{1}{2}(P+Q)$. This results in the JSD formulated as [34]:

$$\mathrm{JSD}(P,Q) = \tfrac{1}{2}\, D_{\mathrm{KL}}(P\|M) + \tfrac{1}{2}\, D_{\mathrm{KL}}(Q\|M), \text{ with } \quad M = \frac{P+Q}{2}. \tag{3.53}$$

The JSD is not a proper metric, since it does not satisfy the triangle inequality. The JSd is a proper metric and is defined as the square root of the JSD, with a value closer to 0 indicating an increasingly similar distribution [36]. The JSd measure is used in the Two-Step infill strategy to compare the predictive distribution of a fidelity level to that of the HFM at the selected infill location. A small value for the JSd

signals that the fidelity level is deemed reliable locally, while a high value indicates that the fidelity level is inaccurate at the selected point.

Lastly, two similar approaches are introduced by Pellegrini et al. [32] and Wackers et al. [33]. Both methods start by selecting a sampling location, using the LCB [32] or the ACAS infill strategy (introduced by Serani et al. [37]) [33].
The next step is to select the fidelity level. This is done by first defining the cost ratio between a fidelity and the highest fidelity, which is used to define the surrogate uncertainty vector, whose entries are the model uncertainties divided by their cost ratio.
The fidelity level, with the maximum value in this prediction uncertainty vector, is selected to sample the new sample point.

3.5. Multi-Point Infill Strategies

Another way to reduce the clock time of the SBO problem is to sample multiple points per iteration. Since the sample points are independent, multiple samples can be evaluated simultaneously. This is enabled by the increase in computing power and access to High Performance Computing (HPC) services. The majority of the commercial HPC services charge per time of provisioned machines, not their utilisation level [38]. Reducing the clock time by parallelising the calculation of infill points can, in these cases, thus result in a reduction of clock time and cost.
Several MP infill strategies from the literature are introduced in detail below; the key characteristics of these strategies are tabulated in Table 3.3. As can be seen in this table, the discussed strategies can be divided into three categories: Single-Point (SP) infill strategy adaptations, combining elements and asynchronous. The first of these adapts an existing SF infill strategy to select multiple infill points per iteration. The second category combines multiple surrogates and/or infill strategies to find multiple infill points. The third category selects a new infill point, which is calculated asynchronously, while a new sample point is selected simultaneously.

Table 3.3: Classification of discussed MP infill strategies.

MP infill	Points per iteration	Base infill	MP mechanism	Additional note
SP infill strategy adaptation				
EIR [29]	1 or n	SF EI	Either 1 HF or n LF new sample points are added per iteration.	The number of LF points is determined based on the cost ratio.
Elite archive [39]	Variable	Design of Experiments (DOE) with elite archive	Number of points is based on the performance compared to the elite archive	-
LCB	Arbitrary	LCB	Select points with different relative importance for exploration and exploitation.	The number of points equals the number of unique relative-importance settings used.
POTI [40]	Arbitrary	PoI	Sequentially select point with PoI with exclusion zone(s) around selected location(s)	-
q-EI [18]	Arbitrary	EI	Closed form q-EI [41] or KB or CL heuristic	The closed form q-EI can be used if q < 10, else KB or CL should be used.

Continued on next page

Table 3.3 – continued from previous page

MP infill	Points per iteration	Base infill	MP mechanism	Additional note
Combining elements				
Multiple infill strategies [42]	4	MinP, EI, LCB, PoI	One sample point per infill strategy.	Can be extended by adding infill strategies
Multiple surrogates Beaucaire 1 [43]	2	DCT on TRBF and CEI on KRG	Select one point using each combination.	-
Multiple surrogates Beaucaire 2 [43]	2	DCT on TRBF and KRG	Select one point using each combination.	-
Multiple surrogate Viana	n	Any	Select one new sample point on every surrogate.	The number of sample points selected is equal to the number of surrogates used.
Asynchronous				
Asynchronous [38]	1 per rank	SP CAND	New point per free rank	Possibly select a new point using KB or CL heuristic if required.

The first of these strategies has been introduced by Yang et al. [29]. This method has been discussed in section 3.4. As discussed in that section some MF infill strategies can select multiple LF sample points per iteration. The strategy by Yang et al. [29] selects n samples vian LHS in a trust region around the sample location determined using EI. The points themselves are evaluated sequentially, with the number determined by the ratio of the computational cost of the LF and HF functions and the surrogate generation time. This algorithm can thus benefit from sampling these LF points in parallel. However, this infill strategy only adds n samples if the LF function is evaluated, not when the HF function is evaluated, which may lead to underutilisation of the available computational power.

A method which adds a variable number of infill points per iteration is discussed by Wang et al. [39]. This method uses an inaccurate search and an elite archive to determine the new infill points.
The first step in this optimisation approach is to set up a DOE, which also serves as the initial elite archive. Next, a surrogate is generated based on this DOE. The infill strategy starts by selecting one exploitation point and a variable number of exploration points. These exploration points are determined using an inaccurate search with differential evaluation based on the elite archive. Through mutation, crossover and selection, a new sample point is selected.
Next, this new sample point is accepted if it outperforms a random sample from the elite archive in exploitation or exploration. In this, the exploitation performance is determined based on the value of the elite archive point and the prediction of the surrogate at the new sample point location. The explorative performance is calculated based on the minimum Euclidean distance of the point to any other point in the surrogate training dataset. This measure is to be maximised to place the points across the design space.
If the new point is accepted, it is evaluated and subsequently added to the surrogate training dataset. If this evaluated point outperforms the comparison point in the Elite archive, it will replace the comparison point in the elite archive.
This process is performed for n samples per iteration. The number of explorative points, n, per iteration is changed based on the performance of the new sample points. If the best and worst points in the new sample points perform worse than the best and worst points in the elite archive, respectively, the exploration points perform poorly. This results in a reduction in the number of exploration points added in the next iteration. On the other hand, if the best and worst points outperform the best and worst points in the elite archive, the number of exploration points in the next iteration is increased. If either the best or the worst point outperforms the respective elite archive point, the performance of the exploration point is equivalent to the elite archive. In this case, the number of exploratory points in the next iteration

will remain unchanged.

Another possible way to select multiple points in an iteration is the use of the LCB infill strategy. This infill strategy utilises a parameter that can be tuned to determine the relative importance between exploitation and exploration. By determining infill points using LCB, each with a different value for this parameter, several distinct new infill points can be generated.

Chaudhuri and Haftka [40] introduced an infill strategy based on the SP probability of Targeted Improvement (POTI) infill strategy. The proposed method can select an arbitrary number of new samples in one iteration.
The method determines the first new sample point using PoI, after which an exclusion radius, eps, is placed around the point on the PoI curve. Following this, a new local maximum is found, resulting in the second infill point. Repeating this process an arbitrary number of times will thus result in an arbitrary number of unique infill points per iteration.
The size of this exclusion radius is chosen as $eps = 0.1\sqrt{n_{\text{dim}}}$ on a normalised design space.

A method based on SF EI is q-EI. This method selects q samples locations, such that the EI is maximised, and was first introduced by Schonlau [18]. An analytical method to determine two sample locations was presented by Ginsbourger et al. [44]. For a larger number of samples, however, a Monte Carlo Simulation was necessary. Later, a closed-form formula to solve for q points was presented by Chevalier and Ginsbourger [41]. This formula allows for fast determination of q samples, if q is reasonably low, typically less than 10. For larger q, heuristic approaches are typically used.
Two such heuristics are Kriging Believer (KB) and Constant Liar (CL), and were introduced by Ginsbourger et al. [44]. Both methods determine the q sample points sequentially, by selecting a new infill point, after which the surrogate is adapted and a new point is selected. The primary difference between the KB and CL approaches lies in how the surrogate is adapted.
KB uses the predicted value of the surrogate at the new sample location to update the surrogate. This will set EI to 0 at this point for an interpolating KRG surrogate. This updated temporary surrogate can than be used to find a new infill location using EI.
CL includes a fixed value, referred to as the 'lie', at the sample location. Again, the surrogate is changed, resulting in a change in the EI. Commonly, one of three values is used for the lie: the minimum of the current sample point dataset, the mean of this dataset or the maximum of the dataset. The larger this lie is chosen, the more explorative the infill strategy will be. Of these three lies, the most exploitative was the most promising approach on the Branin-Hoo test function used by Ginsbourger et al. [44].

This method, however, has a drawback [42]. Primarily, the EI criterion is highly multimodal, making it difficult to find a global maximum for a new sample. A solution to this was proposed by Liu et al. [45]. This method generates four sample points per iteration. These four new sample locations are generated using four different infill strategies. From the comparison of SP infill strategies by Liu et al. [46], the four best-performing infill strategies were selected. Namely, Minimising Prediction of Surrogate Model (MinP), EI, LCB, and PoI. MinP is a strategy that performs optimisation on the objective surrogate, considering the constraints if applicable. The found feasible optimum is used as the new infill point, thus being a pure exploitation point.
Wang et al. [42] extend this model to be able to sample n new samples, by adding a KB strategy, with which every infill strategy used by Liu et al. [47] can obtain an arbitrary number of infill points.

Another method that uses multiple infill strategies is discussed by Beaucaire et al. [43]. This method, however, differs from the previously mentioned infill strategies in that it uses multiple surrogates. The paper introduces two methods: first, an approach that samples one infill point using Deb's Constraint Tournament (DCT) selection with an auto-Tuned Radial Basis Function (TRBF) surrogate, and a second point using CEI on a KRG surrogate.
A TRBF is an RBF model which uses a multiquadratic or Gaussian basis function. The user does not select this basis function; instead, it is chosen during the model's training. In addition, the width hyperparameter is automatically tuned in this process [43], [48].
The DCT selection [49] uses a tournament selection operator that compares two points at a time. To select which sample point is preferred, the following three criteria are used:

1. A feasible point is preferred over an infeasible point.
2. For two feasible points, the point with a better objective function is preferred.
3. For two infeasible points, the point with the smaller constraint violation is preferred.

The second strategy introduced determines two infill points, both using DCT selection, but one on the TRBF surrogate and one on the KRG surrogate. Both methods show similar improvements in performance compared to SP infill strategies.

Viana et al. [50] introduced a different infill strategy, which utilises multiple surrogates. This method utilises multiple surrogates, each with a native or imported error estimate. A surrogate with an imported error estimate, for example, a TRBF model, is combined with the error estimate from a KRG surrogate on the same data.
A new infill point is selected using an infill strategy on each surrogate separately, resulting in a maximum of n new sample points for n surrogates.

All previously discussed MP infill strategies are synchronous processes. Thus, the new infill points are first selected and evaluated, after which the surrogate is updated and a new iteration is run. An asynchronous approach, on the other hand, does not wait for the new infill point(s) to be evaluated before selecting new infill points. An example of such a parallel asynchronous method is presented by Przysowa et al. [38].
This method utilises the SP Candidate points algorithm (CAND) [51], [52] to select new sample locations. This infill technique selects new sample locations in two ways: first, by applying a small perturbation to the best found solution. Secondly, by adding points across the design space. Next, the new sample locations are evaluated on the current surrogate, after which the algorithm selects the best new sample point. This point is evaluated using the full model.
Przysowa et al. [38] implemented this method in an asynchronous optimisation process. This process included an element that keeps track of available resources on the HPC service. If resources are available, a new sample point will be determined and evaluated asynchronously. This process thus does not wait for the previous sample point to be evaluated before adding a new sample point. This can lead to a new sample location having to be determined without an update to the surrogate. This is solved by temporarily including values predicted by the surrogate for previously selected, but not yet evaluated, samples, thus using the KB heuristic.

3.6. Further Improvements

Both the MF SBO and the parallel MP infill strategy approaches exhibit a decrease in computational cost compared to the SF SP SBO. However, a possible further decrease in clock time could be achieved through the combination of MF and MP infill strategies [24], [25]. This combined approach is studied less than the MF and MP approaches separately.
One example of an infill strategy discussed earlier that benefits from the combination is the filter-GEI algorithm proposed by Guo et al. [30]. This algorithm selects multiple points in every iteration, which can be sampled using either the LF or HF functions.
Another example of a combination is the infill strategy by Schouler et al. [8]. This method adds one HF sample using one infill strategy and two LF new samples using LCB. This infill strategy might benefit from the additional determination if a new sample should be evaluated using an HF or LF simulation.
This gap in MF MP infill strategies is thus promising to investigate, aiming to further reduce the cost and clock time of expensive optimisation problems.

3.7. Concluding Remarks

The strategies reviewed in section 3.3 demonstrate that combining HF and LF samples can increase the performance of an SBO run. These methods can be broadly divided into three groups, namely, no-fidelity consideration, heuristic approach, and sequential approach. Most of the reviewed methods in these categories include at most one HF sample per iteration, thus, possibly not fully utilising computational resources.

The MP infill strategies discussed in section 3.5, on the other hand, sample multiple points per iteration. These methods, however, do not include the possible multi-fidelity analysis tools available for engineering optimisation problems. This thus results in both approaches not fully utilising the resources available, leaving possible further performance increases unrealised.

The discussed MF and MP infill strategies served as the building blocks for the combined strategy investigated in this thesis. In particular, the Asynchronous MP approach by Przysowa et al. [38], combined with the sequential MF infill strategies, is identified as promising. The formalisation of this gap and resulting research questions are discussed in chapter 4, where the proposed MF MP infill strategy is also introduced.

4

Gap and Proposed Improvement

As discussed in chapter 3, several ways of reducing the wall-clock time of a Surrogate-Based Optimisation (SBO) run have been investigated. However, as briefly touched upon, further improvements are possible; this and the resulting research questions are discussed in section 4.1. Following this, the hypothesis for the research question is introduced in section 4.2. Next, the requirements for the combined infill strategy were synthesised in section 4.3, followed by the selection of a suitable Multi-Point (MP) infill strategy in section 4.4 and a Multi-Fidelity (MF) infill strategy in section 4.5. Finally, this chapter is concluded with a discussion of the resulting proposed MF MP SBO algorithm in section 4.6.

4.1. Gap

As discussed in chapter 3, both the use of MF and MP infill strategies show an encouraging decrease in wall-clock time for SBO runs. However, the introduced new infill strategies focus on one of the two methods. From this, the question arose whether further improvement could be achieved by combining the MP and MF infill strategies into a MF MP infill strategy. This results in the main question for this thesis:

> How does a parallel multi-point multi-fidelity infill strategy affect the performance of a Surroagte-Based optimisation run?

To answer this question, either a new MP, MF infill strategy must be created or a MP and a MF infill strategy have to be combined. This raises the question: how to decide on the number of points and their fidelity to sample each iteration? And how to ensure the computational resources are utilised as best as possible?
Since, in a sequential Efficient Global Optimisation (EGO), a High Fidelity (HF) sample may be computationally twice as expensive as a Low Fidelity (LF) sample, it is possible that, on a 2-core system, in one iteration one HF and two LF samples can be computed. However, in most MF infill strategies, the sample's fidelity cannot be selected a priori; it is not trivial to add an LF sample to the pending set of one HF and one LF sample. This results in the following sub-questions:

> How to decide the sample location and fidelity level for a new sample point?
> How to decide on the number of new high-fidelity and low(er)-fidelity samples per iteration?
> How can the available computational resources be maximally used?

4.2. Hypothesis

The resulting MF MP infill strategy will combine a MF Single-Point (SP) infill strategy with a Single-Fidelity (SF) MP infill strategy, which in turn are both adapted from SP SF infill strategies. Since the base MP SF and SP MF infill strategies are shown to converge to the optimum of a function, it is expected that

the resulting optimum from the combined infill strategy will not differ substantially.
If a difference in the resulting optimum is observed, this is expected to be due to the addition of LF samples, since these points provide less accurate information. The addition of these points could also yield a better solution in using a MF MP infill strategy. Caused by utilising the MF MP strategy, which allows for adding more points per iteration (increasingly so when a larger fraction of the sample points are LF), and these points could possibly be more exploratory points. This could open the possibility to sample more of the design space, potentially overcoming local optima.
Combining the MP infill strategy's ability to sample multiple points per iteration, distributed across several Central Processing Units (CPUs), with the MF infill strategy's ability to sample points at different computational costs will likely reduce wall-clock time.
Combining these expectations leads to the hypothesis for the main research question that a MF MP infill strategy will yield improved performance compared to SF MP, MF SP, and SF SP infill strategies. The expected improvement is expected to stem from a reduction in the time required per optimisation run, due to the increase in information available for training the surrogate at time t, leading to earlier convergence of the optimiser. With the addition of more infill points, the optimiser is expected to achieve a higher success rate, thereby improving Expected Runtime (ERT).

4.3. Requirements

Next, the proposed MF MP infill strategy was created by combining an existing MF and MP infill strategy. To select suitable infill strategies to be combined, first, a set of requirements was generated:

REQ-CONSTR-1: The method shall be able to solve unconstrained and constrained problems.
REQ-FID-1: The method shall be able to accept n fidelity levels, with $n \geq 1$.
REQ-NPTS-1: The method shall be able to sample n samples per iteration, which is scaled with the available computation resources, which are kept constant throughout the optimisation.

The first requirement, REQ-CONSTR-1, ensures that the infill strategy is applicable to a range of functions and problems, both constrained and unconstrained. In this work constrained problems are not considered, but it is desirable that the infill strategy can handle constrained problems or be easily adapted to do so, to have a more practical application for engineering optimisation problems. The next requirement, REQ-FID-1, ensures that the method scales to the available resources for the problem. Adding multiple fidelities could result in fidelity 2 (a higher number is a lower fidelity, with 0 being HF), sampled more than fidelity 1, reducing the computation time. If only fidelities 2 and 0 are to be included, this could cause the infill strategy to choose fidelity 0, whereas otherwise fidelity 1 would be chosen, increasing the sampling time. This behaviour was shown by Przysowa et al. [38]. The last requirement ensures the resulting infill strategy is not restricted in the number of samples that can be added per iteration. If this requirement is not satisfied, the final method may not optimally use available resources by failing to assign new sample locations to free ranks, ultimately resulting in a reduction in performance.
Following the requirements, the goal of the combination is formulated:

The aim of combining the infill strategies is to reduce the optimisation wall-clock time by utilising MP and MF samples while achieving the same or an equivalent optimum as the separate infill methods.

To achieve this goal, the computational resources should be utilised as efficiently as possible. This means minimising the time CPUs are idling. A benefit for a potential infill strategy can therefore be to not include the computational cost of the fidelity levels. Since this is not always available accurately and is not necessarily constant throughout the design space, this could lead to samples added that would not be added using a different cost ratio.
This cost ratio can, however, be used to determine at which fidelity level to sample for a MP sequential EGO approach. Since the sequential EGO approach first selects the sample locations and their fidelities, and then computes all points concurrently, knowing the cost ratio can improve the use of computational resources. For example, in the case where an HF sample is twice as expensive as an LF sample, and the optimisation is run with two ranks available. The first sample could be selected to be an HF sample, knowing the cost ratio, this can open up the possibility to select two LF samples for the other rank,

minimising the idle time of both ranks.
However, this poses a problem in that for most MF infill strategies, it is not known at which fidelity level the next sample is generated before generating a new sample location.
Thus, based on the considerations, the likely combination of infill strategies is an SF MP algorithm that performs asynchronous computation of sample locations, and a MF SP infill strategy that does not base the fidelity decision on the cost-ratio.

4.4. Multi-Point Infill Methods

The first step in developing a new infill method was to select the MP infill strategy to adapt or combine. The methods introduced in section 3.5 can roughly be divided into three categories; SP infill strategy adaptation, combining elements and asynchronous.
The first of these categories, SP infill strategy adaptation, adapts an SP infill strategy to select multiple samples per iteration. A discussed example of this is the q-EI infill strategy as introduced by Schonlau [18]. This infill strategy adapts the Expected Improvement (EI) infill criterion such that an arbitrary number of sample points can be selected. For approximately 10 points or more, however, a heuristic approach should be used. Most infill strategies in this category can select an arbitrary number of samples, complying with REQ-NPTS-1 and can be adapted to handle constraints using, for example, Probability of Feasibility (PoF).

A downside, however, is that the number of samples to select is non-trivial in that it would require being adapted to use a MF infill strategy for which the fidelity level to sample at can be selected. This is needed to optimally use the computational power available. To aid the decision on what fidelity level is required, the cost ratio is required to fill the still available computational power per iteration with the maximum useful new information.

The second category, which can be distinguished, is the category that combines multiple infill strategies and/or surrogate models. For example, the method introduced by Viana et al. [50] combines several surrogates, each of which is used to select a unique new sample point. This has the benefit that it can utilise the strengths of multiple surrogate models. In addition to the downside the single point adaptation category has, these methods can not scale easily to an arbitrary number of new sample points. Since the number of sample points is determined based on the number of infill strategies, surrogate model combinations are used. This does not comply with REQ-NPTS-1, since it can only scale to the possible combinations.

The last category is the asynchronous approach, exemplified by Przysowa et al. [38]. These methods do not select the infill points simultaneously, after which all points are calculated concurrently. These approaches select one new sample point, which is calculated immediately asynchronously. While this point is calculated, a new sample point is selected for another rank. This process is repeated until all ranks are evaluating a point. Once an infill point is evaluated, it is added to the surrogate, after which the free rank receives a new infill point and computes it. This process is repeated until convergence, essentially generating an Asynchronous EGO algorithm. The added benefit of this category is that implementing a MF infill strategy does not require the cost ratio between the fidelities to maximise the utilisation of the computational power, since available ranks receive an infill point asynchronously.
A downside of this method is that a situation can occur, like the start of the optimisation, where no new sample points are added to the surrogate, and a new infill point is to be selected. Since the surrogate is not updated, selecting a new sample point would lead to the previously selected point being reselected. To overcome this issue, the selection of the new sample point must be performed using a heuristic method, such as Kriging Believer (KB) or Constant Liar (CL). KB adds the surrogate prediction of the pending point(s) to the surrogate after which it is retrained, and a new sample location can be selected. CL works similarly but does not add the surrogate's predicted value; it instead adds the minimum, mean, or maximum of the sampled points.
One possible downside of the asynchronous approach relative to the other two categories is that the surrogate must be updated for each sample point, potentially increasing the number of times it must be trained and the computational cost. However, training the surrogate is often orders of magnitude cheaper

than computing the sample points, so this effect is expected to be small. If training the surrogate is as expensive or more expensive than the function evaluations themselves, utilising a non-SBO optimisation algorithm might be worthwhile. This asynchronous approach can satisfy REQ-CONSTR-1 and REQ-FID-1, depending on the choice of SP infill strategy. Since the asynchronous approach can accommodate any SP infill strategy, this method does not pose a risk of failing to satisfy these requirements. The last requirement, REQ-NPTS-1, should also be satisfied by this approach, since the number of sample points can be easily scaled, possibly using the KB or CL heuristic to select a new sample point if required.

Each of the discussed categories presents advantages and disadvantages, which are summarised in Table 4.1. Considering the goal of minimising wall-clock time by maximising use of computational resources and avoiding the need for a computational cost ratio, the asynchronous approach was selected as the basis for the MP component of the combined infill strategy.

Table 4.1: Advantages and disadvantages of the MP infill strategy categories.

MP category	Advantages	Disadvantages
SP infill adaptation	• Arbitrary number of sample points • Constraint handling with certain infill strategies	• Non-trivial to adapt to MF • Requires the cost ratio to optimally use computational resources
Combining elements	• Exploit strengths of multiple surrogates and/or infill strategies • Fixed number of sample points • Can easily be extended to MF	• Requires the cost ratio to optimally use computational resources
Asynchronous	• Arbitrary number of points, limited by resources available • Any SP MF infill strategy can be used • No cost ratio required	• More frequent surrogate retraining

4.5. Multi-Fidelity Infill Methods

As discussed, the selected asynchronous EGO method can accept any SP infill strategy. The asynchronous EGO can thus be readily extended to handle a MF optimisation process by using a MF infill strategy. Of the three categories of MF infill strategies introduced in section 3.4, only infill strategies from the last two categories, the heuristic and sequential approach, were considered.
The first category, no-fidelity consideration, could lead to a suboptimal use of available resources by selecting too many LF or HF samples at possibly wrong locations for the fidelity level and was therefore not considered. The next consideration in selecting suitable MF infill strategies was the requirement of an available cost ratio between fidelity levels. Since it cannot be guaranteed that this is known a priori for every problem, it would be beneficial if the MF infill strategy did not rely on this ratio. Taking these criteria into account, the following four promising infill strategies were identified:

- Variable fidelity EI [27]
- Adapative Multi-Fidelity Expected Improvement (AMEI) [28]
- Filter-Generalised Expected Improvement (GEI) [30]
- Two-Step [31]

The first of which is the variable fidelity EI as introduced by Zhang et al. [27]. This method maximises the expected improvement of the HF surrogate given a point is sampled at fidelity l, finding the sample location and sample fidelity simultaneously. This variable-fidelity EI is defined for two fidelity levels; thus, it must be extended to accommodate an arbitrary number of fidelity levels.

The next promising infill strategy was introduced by Hao et al. [28] and improves on the previously

discussed method. This method improves upon the strategy introduced by Zhang et al. [27] by including the best LF sample point. Not considering this point has been reported to cause too many LF points to be sampled, thereby reducing performance and possibly leading to overfitting. This method is again defined for two fidelity levels and should be adapted to handle constraints.

The third infill strategy that was identified was the strategy introduced by Guo et al. [30]. This method uses the GEI infill strategy as a base. This infill strategy was created by Schonlau [18] to combat the emphasis on local search that EI can tend to. This was done by introducing the parameter g, which determines the relative importance of exploration and exploitation. This method was used by Guo et al. [30] for the filter-GEI. This method determines the sample location using GEI, after which the fidelity level is sampled. If the expected value of the sample point is below the filter threshold, the point is sampled using HF; otherwise, an LF sample is used. This threshold is changed throughout the optimisation and depends on the correlation between the LF and HF surrogate. Again, this method is defined for two fidelity levels and has to be extended to handle constraints.

The last promising infill strategy is the Two-Step infill strategy of Garbo et al. [31]. This method selects the sample location using an SF infill strategy determined by the user, followed by selecting the fidelity level to sample at. This is done by first determining all fidelities which are accurate enough at location x, using the Jensen-Shannon Divergence (JSD) as an accuracy metric. From this list, the computationally cheapest (thus lowest-fidelity) fidelity level is selected. Since this method can accept any SF infill strategy, it can handle constraints by selecting a suitable SP SF infill strategy. In addition, the way in which the fidelity level is selected allows the infill method to use an arbitrary number of fidelities.

The key characteristics of these four infill strategies are summarised in Table 4.2. Of these promising infill strategies, the Two-Step infill strategy was selected to be used in the MF MP infill strategy, as this infill method can natively support constraints and generate infill samples for an arbitrary number of fidelity levels. The single fidelity infill strategies used in combination were the commonly used EI [5], [6] and its more globally sampling adaptation GEI [18]. To adapt this method to be able to handle constraints the Constrained Expected Improvement (CEI) and Constrained Generalised Expected Improvement (CGEI) counterparts can be used.

Table 4.2: Comparison of the four candidate MF infill strategies.

Property	**Variable fidelity EI** [27]	**AMEI** [28]	**Filter-GEI** [30]	**Two-Step** [31]
Location & fidelity selection	Heuristic	Sequential	Sequential	Sequential
Fidelity levels	Two	Two	Two	Arbitrary
Constraint handling	Requires adaptation	Requires adaptation	Requires adaptation	Using appropriate SF infill strategy
Cost ratio required	No	No	No	No

4.6. Proposed Optimisation Algorithm

Combining the selected MP infill strategy and the MF infill strategy yields an asynchronous EGO utilising the Two-Step MF infill strategy. The flow of this method is illustrated in Figure 4.1.
This resulting method starts by selecting a set of initial samples, S_0, with coordinates X_0 via Latin Hypercube Sampling (LHS). On this set of initial samples, a Hierarchical Kriging (HK) model is trained. After the optimisation is invoked, a set of available workers is generated. At the start of the optimisation, this consists of all workers available to the optimiser. Next, a new infill location is selected using an SF infill strategy, after which, using the Two-Step method, the fidelity level to be sampled at is selected. This point is then sent to the first free worker, who starts the calculation of the point. This process is repeated until no workers are available to receive a task, at which point every rank is calculating a new

sample point. When a rank finishes the calculation of the new point, this is added to the HK surrogate, which is retrained and the worker is added again to the set of available workers. In some cases, it could be possible that the surrogate has not yet been updated with a new sample point if a new point is to be selected. In those cases, the new sample point is selected using the KB heuristic.
The used SF infill strategy in the Two-Step infill strategy can be any SF infill strategy. In this work, the EI and GEI are used.

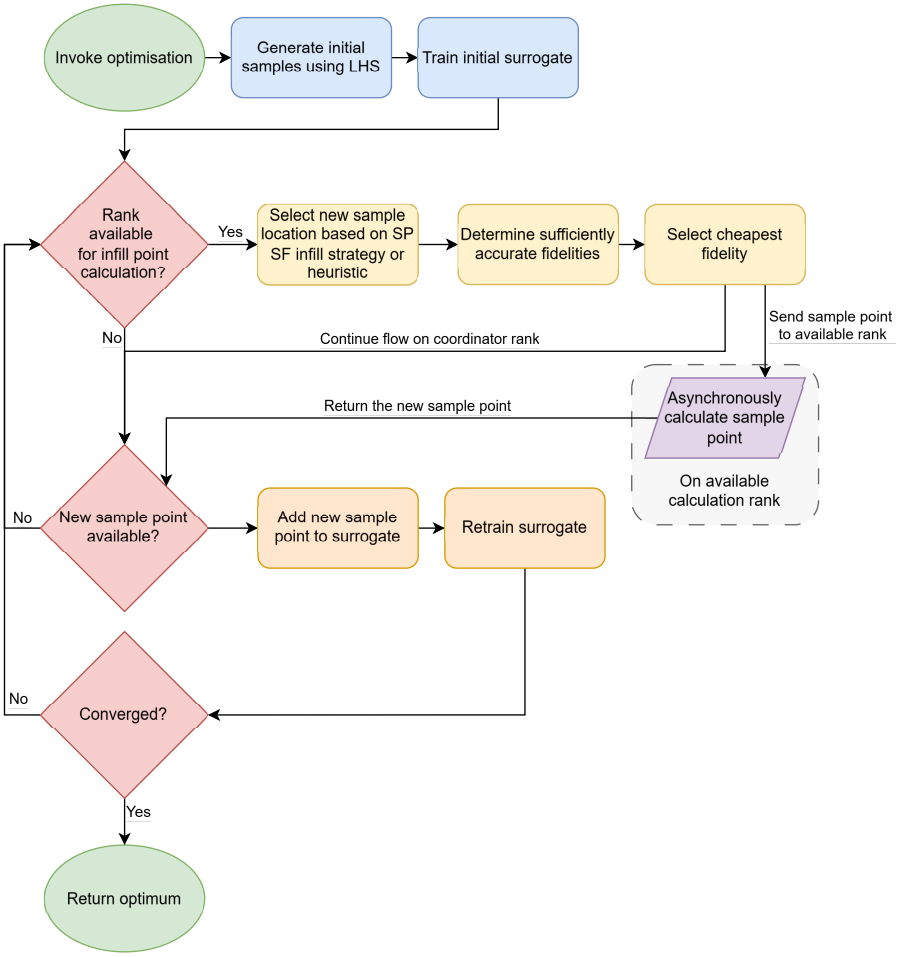


Figure 4.1: The proposed optimisation algorithm.

5

Verification of Implementation

This chapter introduces the framework's implementation. The implementation details can be found in Appendix A, as well as a description of the measures taken to ensure consistency and repeatability. The verification starts with the surrogate models (Kriging (KRG) and Hierarchical Kriging (HK)) in section 5.1, followed by the infill strategies (random, Expected Improvement (EI) and Generalised Expected Improvement (GEI)) in section 5.2. Lastly, the verification of the implemented optimisation algorithms is discussed in section 5.3.

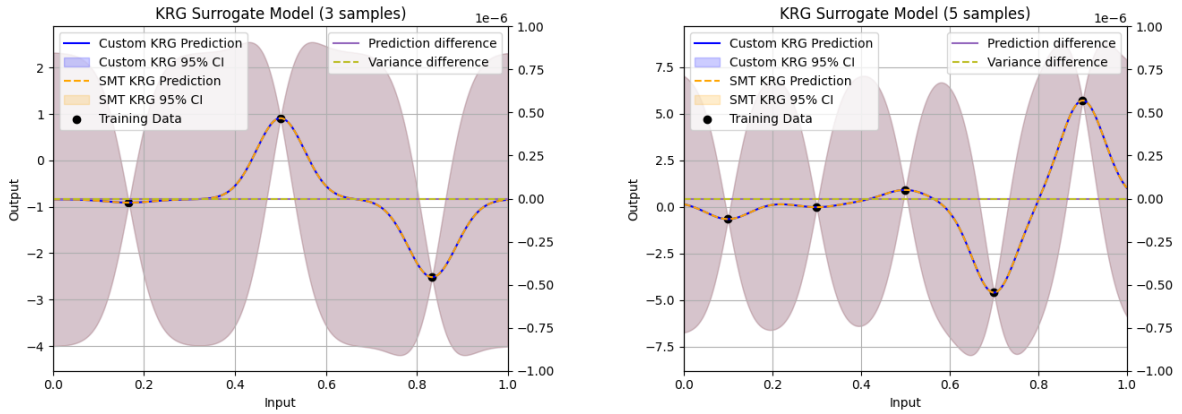
5.1. Surrogate Models

The first components used in the optimisation, which were implemented and verified, are the surrogate models. The selected surrogate models are the Single-Fidelity (SF) KRG surrogate and the Multi-Fidelity (MF) HK model. The KRG surrogate is used in the SF reference test and as a base model for the MF HK model.

5.1.1. Kriging

The first of these surrogate models is the KRG. Since the surrogate-based optimisation framework uses Surrogate Modelling Toolbox 2.0 (SMT) [1] as a base, the implemented KRG model is a wrapper of the SMT KRG model. To ensure that the wrapper does not alter the resulting surrogate, a comparison was made between the original model and the model obtained from the wrapper across several functions (included in the MF2 Python library [21] and discussed in more detail in chapter 7) and varying numbers of sample points. In all cases, the relative difference between the wrapper and SMT implementation for both the mean prediction and variance is lower than 1×10^{-7}. Two of the test cases for the 1-dimensional Forrester (5.1) function are illustrated in Figure 5.1.

$$f_{\text{HF}}(x) = (6x - 2)^2 \sin(12x - 4). \tag{5.1}$$



(a) KRG surrogate for 3 samples.

(b) KRG surrogate for 5 samples.

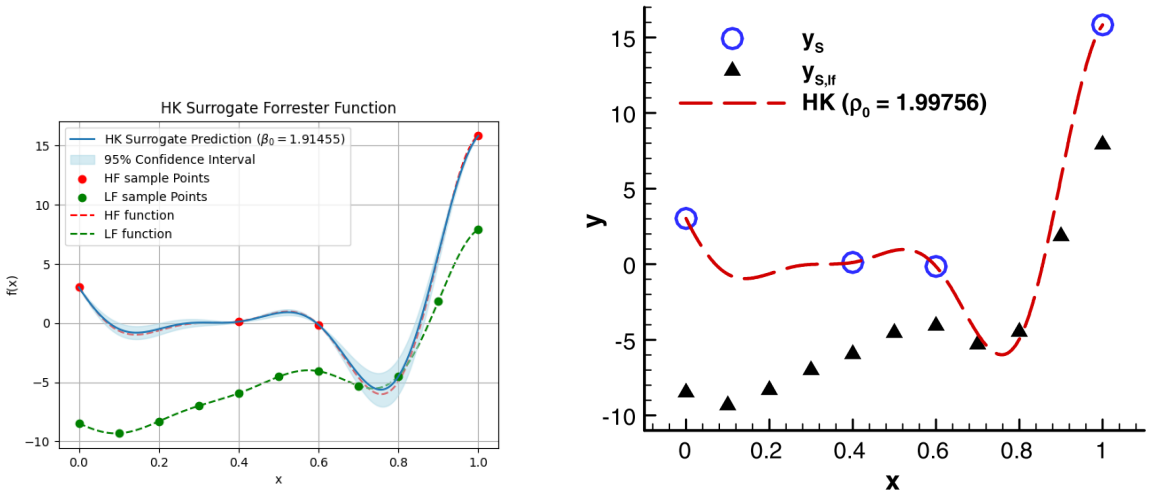
Figure 5.1: Comparison of SMT and wrapper KRG surrogate on 1-dimensional Forrester function.

5.1.2. Hierarchical Kriging

The next surrogate model verified is the MF HK model introduced by Han and Görtz [22] and discussed in subsection 3.3.2. The implementation includes both the Gaussian exponential and the cubic-spline correlation introduced by Lophaven et al. [53]. The implemented HK model was verified against the two reference cases used by Han and Görtz [22]. The first of the two cases used the High Fidelity (HF) Forrester function (5.1), with a commonly used Low Fidelity (LF) function (5.2). With the following nested sample set: $x_{\text{LF}} = \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ and $x_{\text{HF}} = \{0.0, 0.4, 0.6, 1.0\}$.

$$f_{\text{LF}}(x) = 0.5 f_{\text{HF}}(x) + 10(x - 0.5) - 5. \tag{5.2}$$

The resulting models are presented in Figure 5.2, in which Figure 5.2a is the custom implementation and Figure 5.2b the reference.



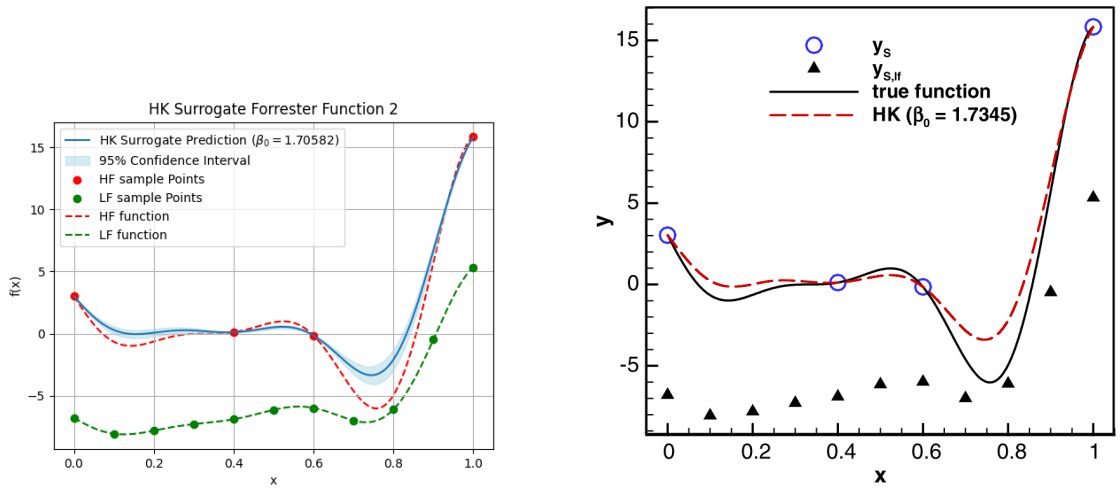
(a) Resulting HK model from custom implementation.

(b) Resulting reference HK model [22].

Figure 5.2: Comparison of custom HK implementation and reference [22] for the Forrester verification case.

The second verification case examined uses the same sample set and HF function as the previously discussed case, but uses a modified LF function (5.3). This new formulation introduces a more complicated correlation between the high- and low-fidelity functions. The resulting surrogates are illustrated in Figure 5.3.

$$f_{\text{LF}}(x) = 0.4 f_{\text{HF}}(x) + 10\left(0.4x^4 + 0.1x^2 + 0.2x + 0.2\right) - 10. \tag{5.3}$$



(a) Resulting HK model from custom implementation.

(b) Resulting reference HK model [22].

Figure 5.3: Comparison of custom HK implementation and reference [22] for the Forrester verification case with updated LF function.

As shown in the illustrated results for both test cases, there is a difference between the trend coefficient β_0 (whose values are tabulated in Table 5.1) and a slight difference in the shape of the HF prediction. The slight difference in the predictor is most pronounced near the function's optimum, where the implementation attains a higher value.

Table 5.1: Comparison of trend coefficients for different LF Forrester functions.

Item	Forrester	Updated Forrester
Reference [22]	1.997 56	1.7345
Implementation	1.914 55	1.705 82

This difference was concluded to be the result of a different width parameter, θ found in the training process. This hyperparameter is tuned by maximising the log-likelihood. Since this log-likelihood can be highly multi-modal, this was done using Scipy's differential evolution [54]. Han and Görtz [22] use a genetic algorithm to overcome this issue.

Differential evolution is a stochastic population-based optimisation algorithm, which does not use gradient information to find a minimum. It can search large regions of the design space, but often requires many function evaluations. The optimisation starts with a population of candidates, or vectors, with $\bar{v} \in \mathbb{R}^n$. In each iteration, these vectors are mixed according to a selected strategy, resulting in the mutated vector. In the next step, crossover is applied between the mutated vector and the reference vector to increase population diversity. This is achieved by combining the two vectors and selecting the corresponding element from each. This is done by taking n numbers from a binomial distribution on $[0, 1)$. For each element of the vector, the value from the mutated vector is taken if the corresponding value from the binomial distribution is less than the set recombination constant. Otherwise, the element is taken from the reference vector. This results in the candidate vector. Next, the fitness of the trial vector is determined. Namely, if the trial is better than the original vector, the original vector is replaced by the trial vector. In the original algorithm from Storn and Price [55], at the end of each full iteration, the best solution is updated from the new population, so it is updated once per iteration. The implementation in Scipy (1.15.3), however, continually updates the best solution throughout an iteration. This can lead to faster convergence, since the trial vectors can immediately benefit from the improved solution.

The conclusion regarding the difference in predictors was based on a simple numerical test case, which showed no errors in the deterministic parts of the implementation. This shows the difference may have resulted from the non-deterministic optimisation of the log-likelihood. This analysis is discussed in more detail in Appendix B. As a last step to verify the conclusion, a reverse calculation of the reference

β_0 and trained β_0 to find the optimum θ for each was performed. This resulted in the found values as tabulated in Table 5.2. From the found θ_{opt} and their corresponding log-likelihood, it can be concluded that the previous conclusion is correct. As the table shows, the reference found a θ corresponding to a worse local maximum, whereas the implementation found a better optimum, since the log-likelihood is maximised.

Table 5.2: Comparison of found optimal θ.

Item	β_0	θ_{opt}	log-likelihood
For HF (5.1) and LF (5.2)			
Reference [22]	1.997 56	0.000 94	−10.1501
Implementation	1.9145	0.0387	−9.7567
For HF (5.1) and LF (5.3)			
Reference [22]	1.7345	1.5527	−15.920
Implementation	1.7058	0.0703	−4.372

5.2. Infill Strategies

The implementation of the selected infill strategies, discussed in section 4.6, is discussed next. To verify the infill strategies, the previously verified surrogate models were used. This discussion of the verification begins with the random-point infill strategy.

5.2.1. Random Point

As a baseline reference, a random infill strategy was implemented. This infill strategy illustrates a worst-case scenario in which a random point is added to the set of sample points until convergence is achieved.

To ensure the correct functioning of this method, a large number of samples were selected on $(x_1, x_2) \in [(-10, -10), (10, 10)]$ for a varying number of fidelities. If the implementation of the infill strategy is correct, the entire sample design space should be sampled without duplicates. In addition, the samples should be uniformly distributed in both sample location and sample fidelity. The case for $n_{fid} = 5$ with 2000 sample locations is figured in Figure 5.4, which shows the correct working of the random infill method.

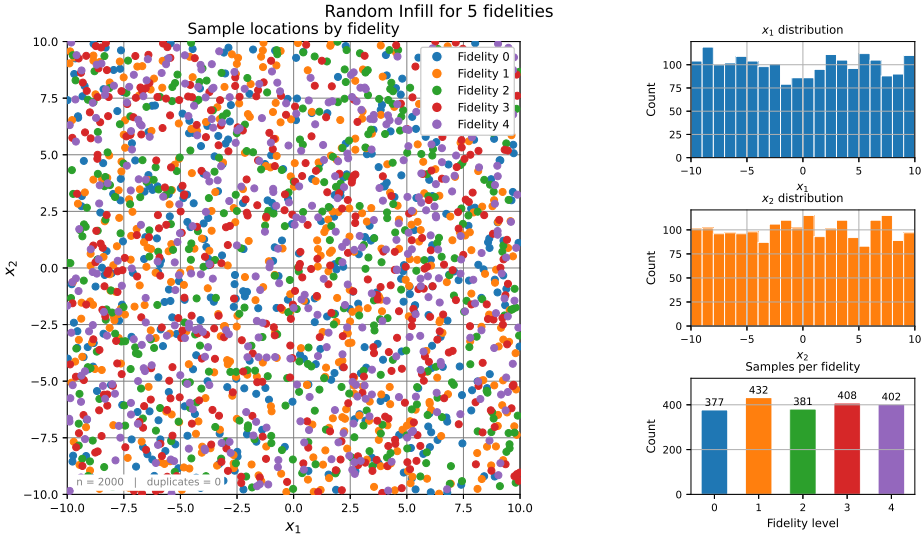


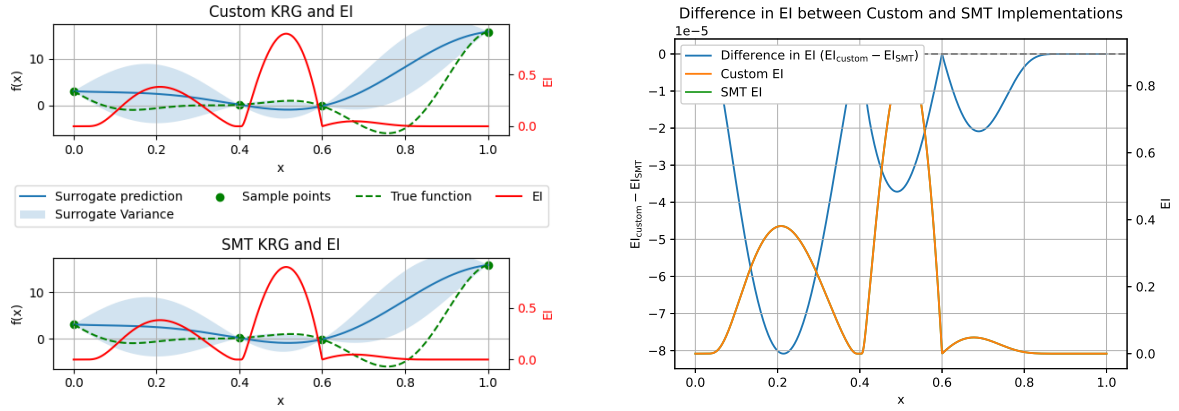
Figure 5.4: Verification of the random infill strategy.

5.2.2. Expected Improvement

The next verified infill strategy is the EI infill strategy (discussed in more detail in subsection 3.2.3). This infill strategy selects a point at the location $\bar{x}$ that maximises expected improvement over the design

space, thereby balancing exploration and exploitation. Since the EI-value, calculated using (3.28), can be highly multi-modal, the optimum was found using the same differential evolution algorithm as used in subsection 5.1.2.

To ensure that the EI infill strategy works correctly, it was compared to the EI implementation in the SMT toolbox [1]. Both the SMT and custom EI infill were calculated over a KRG surrogate of the Forrester function. These surrogates were equal in both cases. If the custom EI implementation is correct, both implementations should yield identical EI values throughout the design space. The resulting EI and differences are illustrated in Figure 5.5.



(a) SMT and custom EI values.

(b) Difference SMT and custom EI values.

Figure 5.5: SMT and custom EI values for KRG surrogate of Forrester function.

In Figure 5.5a, the EI values look equal; however, a difference can be observed in Figure 5.5b, but these differences are small. The maximum absolute error and maximum relative difference, with the corresponding other value, are tabulated in Table 5.3. Since these values are small, the implementation was deemed to be verified.

Table 5.3: Maximum differences SMT and custom EI in Forrester test case.

Maximal Error	Absolute Error	Relative Error	Corresponding x
Maximum absolute error	8.082	2.126×10^{-4} %	[0.214]
Maximum relative error	2.266×10^{-289}	0.0949 %	[0.400]

5.2.3. Generalised Expected Improvement

The GEI was verified in Two-Steps; first, a test was performed comparing the returned infill value for a $g = 1$, with the values of the previously verified EI infill strategy. As in the special case in Equation (3.32), this is a special case for the GEI infill strategy where the infill strategy can be simplified to the EI infill strategy. The test yielded equal infill values, verifying that the GEI infill strategy works correctly for $g = 1$.

The next step in verifying the correct functioning of this infill strategy was plotting the infill value and each maximum for different values of g. What is expected is that the maximum (and thus selected infill location) moves to a more explorative point, with an increase in g. The result of this test is visualised in Figure 5.6, where the expected behaviour is visible.

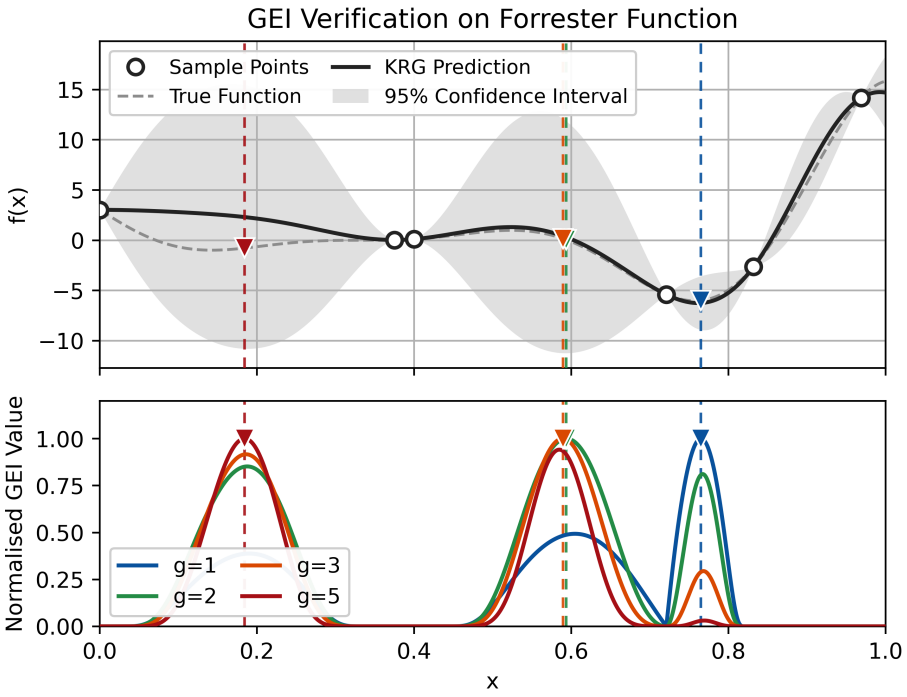


Figure 5.6: Verification of the GEI infill strategy on the Forrester function.

The last verification step was to verify the implemented closed-form GEI to the integral form. The integral form of GEI is [18]:

$$E\left[I^g\right] = \hat{s}^g \int_{-\infty}^{u} (u - z)^g \, \phi(z) \, dz, \tag{5.4}$$

with $z = \frac{y-\hat{y}}{\hat{s}}$, $u = \frac{f_min-\hat{y}}{\hat{s}}$. A comparison was made for $g \in [1, 2, 3, 5]$ at several x locations. These tests all showed a difference below the relative difference threshold of 1×10^{-5} and the absolute difference threshold of 1×10^{-10}. The combination of these three tests showed that the implemented GEI works correctly.

5.2.4. Two-Step

The Two-Step infill strategy [31] was verified in four steps; first, it was ensured that the sample location selected is correct. This was done by selecting the infill location for a set of different surrogates and selecting a sample point on the same surrogate with the selected infill strategy. Both strategies selected the same sample location, thus verifying that the sample location did not change. The next step was to determine if the list of sufficiently accurate fidelities was created correctly. This was done by generating the list of sufficiently accurate fidelities for a set of surrogates with an HF and an LF prediction, which differed by a different amount. The results are tabulated in Table 5.4. In this table, the expected and actual outputs are equal, demonstrating the correct functioning of this part of the implementation. The

third test performed was to ensure that for the Jensen-Shannon distance (JSd) test, the correct fidelity level was selected. In the implementation, the HF has an index of 0, with higher indices corresponding to lower fidelity levels. Thus, the selected fidelity level should be max(accurate enough fidelities). This behaviour is again shown in Table 5.4.

Table 5.4: Accurate enough fidelities and selected fidelities for Two-Step using a JSd threshold of 0.7.

μ_{HF}, μ_{LF}	LF JSd Expected	LF JSd Actual	Fidelities Expected	Fidelities Returned	Selected Fidelity
(0, 0)	0.0	0.0	[0, 1]	[0, 1]	1
(0, 1σ)	0.4	0.400	[0, 1]	[0, 1]	1
(0, 2σ)	0.697	0.697	[0, 1]	[0, 1]	1
(0, 3σ)	0.872	0.872	[0]	[0]	0
(0, 50σ)	1.0	1.0	[0]	[0]	0

The last verification step combined the selection of the infill location and the fidelity level. This resulted in the location being equal to that selected by the SF infill strategy alone, with the correct fidelity level, verifying the implementation.

5.3. Optimisation Algorithms

Lastly, the verification of the optimisation algorithms is discussed. Two optimisation algorithms are implemented, namely a sequential Efficient Global Optimisation (EGO) as a reference and the selected Multi-Point (MP) algorithm, as introduced in section 4.4, resulting in an asynchronous EGO. To verify both cases, the previously verified KRG surrogate and EI infill strategy were used. The first optimisation algorithm discussed is the sequential EGO.

5.3.1. Sequential Efficient Global Optimisation

To verify the working of the sequential EGO, it was tested using a KRG surrogate and EI infill strategy on several functions. The functions used are the HF functions included in the Python library MF2 [21], which are discussed in more detail in chapter 7. The optimisation was run with 15 random seeds for each function to ensure that the initial set of samples, created using Latin Hypercube Sampling (LHS), changes, thereby increasing the likelihood of finding the function's optimum. In addition, it was ensured that none of the found optima, for any seed, is a better solution than the known optimum. This resulted in the optima tabulated in Table 5.5, which are all minimisation problems, except for the Currin function.

Table 5.5: Verification of sequential EGO.

Function	Known Optimum	Found Optimum	Difference
Forrester (1D)	-6.0207	-6.0207	1.29×10^{-10}
Bohachevsky (2D)	0	0.000 376	3.76×10^{-4}
Booth (2D)	0	6.31×10^{-29}	6.31×10^{-29}
Branin (2D)	-333.916	-333.9196	6.74×10^{-5}
Currin (2D), maximisation	13.799	13.799	-9.11×10^{-7}
Himmelblau (2D)	0	1.45×10^{-4}	1.45×10^{-4}
Six Hump Camelback (2D)	-1.032	-1.031	2.01×10^{-4}
Park91A (4D)	2.718×10^{-8}	2.718×10^{-8}	0
Park91B (4D)	0.667	0.667	0
Hartmann6 (6D)	-3.042	-3.041	1×10^{-3}
Borehole (8D)	7.820	7.821	8.4×10^{-4}

From this table, it is clear that the difference between the found and actual optimum is very small. Combining this with the diagram of the optimisation process for the Forrester function in Figure 5.7, it was determined that the implementation of the sequential EGO works correctly.

The diagram of the optimisation process for the Forrester function using an initial sample set $x = \{0, 0.4, 0.6, 1\}$. For each iteration, it is visible that the new sample location is selected based on the infill criterion. This is repeated until convergence is reached, based on the criterion for the minimum threshold for the infill value, which was set to 1×10^{-6}.

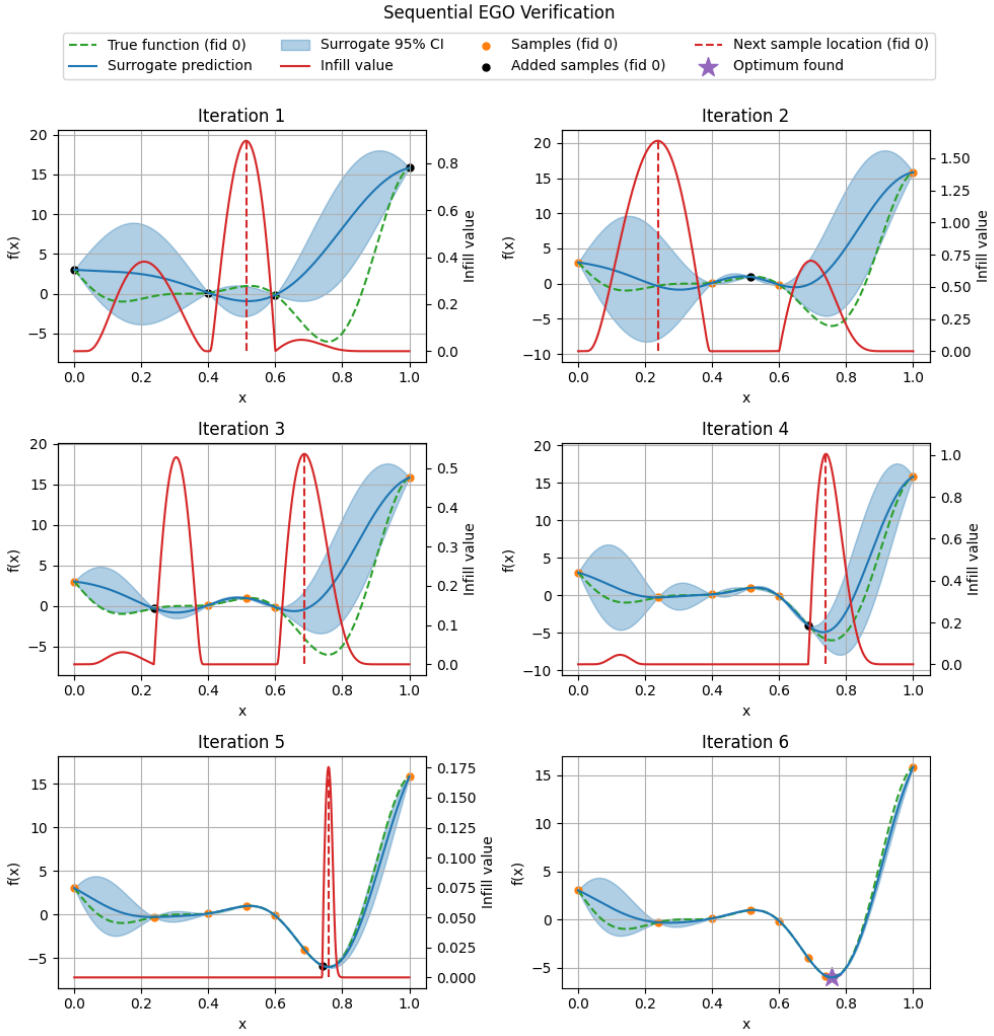


Figure 5.7: Sequential EGO process for the Forrester function.

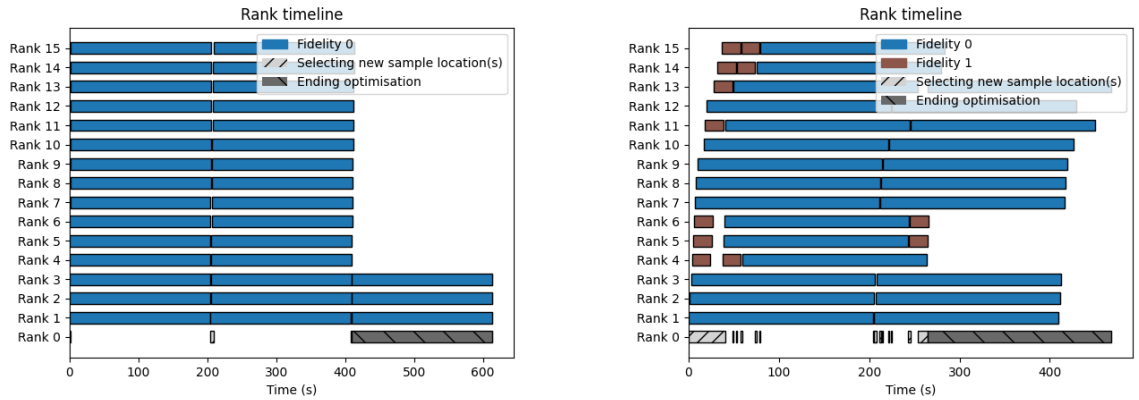
5.3.2. Asynchronous Efficient Global Optimisation

The last main element verified was the asynchronous EGO algorithm based on the method from Przysowa et al. [38]. To verify the working of this algorithm, the same steps were undertaken as for the sequential EGO. Thus, the optimiser was run for several seeds on the function included in MF2 [21], which are discussed in more detail in chapter 7. The results are then compared with the known optima of the functions, and finally, it is ensured that all found optima do not have a function value better than the actual optimum. The results are tabulated in Table 5.6, all problems are minimisation problems, except for the Currin function which is a maximisation problem.

Table 5.6: Verification of Asynchronous EGO.

Function	Known Optimum	Found Optimum	Difference
Forrester (1D)	-6.0207	-6.0207	1.190×10^{-11}
Bohachevsky (2D)	0	4.68×10^{-4}	4.68×10^{-4}
Booth (2D)	0	6.31×10^{-29}	6.31×10^{-29}
Branin (2D)	-333.916	-333.916	7.78×10^{-9}
Currin (2D), maximisation	13.799	13.799	-1.837×10^{-5}
Himmelblau (2D)	0	4.073×10^{-5}	4.073×10^{-5}
Six Hump Camelback (2D)	-1.032	-1.032	4.19×10^{-5}
Park91A (4D)	2.718×10^{-8}	2.718×10^{-8}	0
Park91B (4D)	0.667	0.667	0
Hartmann6 (6D)	-3.042	-3.042	4.4×10^{-4}
Borehole (8D)	7.820	7.822	2.49×10^{-3}

From this table, it is clear that the found optima are all close to the actual optima, and no value is more optimal than the actual optimum. Therefore, the operation of the optimisation algorithm is verified, except for the correct operation of the inter-rank communication. To this extent, the task of a rank at a time is plotted in the timeline in Figure 5.8. In Figure 5.8a the rank timeline for an SF problem is displayed. Since this verification case was performed on an SF problem with 16 ranks, ranks 1 through 15 are expected to perform calculations using the HF function. These blocks of calculation on the timeline are expected to start staggered, since the first rank that receives a point immediately starts the calculation, followed by the second rank receiving its sample location later. These ranks should also be busy for most of the optimisation duration.
Rank 0, on the other hand, should show the selection of the new sample locations, followed by a large downtime, while the ranks compute the resulting sample points. Since an SF problem was used, the selection of new samples should be in blocks for every rank. This would not be the case if a MF problem were used, since the computation block of the worker ranks would differ in length across fidelities.
Such a MF case is displayed in Figure 5.8b. As expected, it is visible in this case that the different fidelity levels have different computation times, reflected by the width of the evaluation blocks. In addition, where in the SF case all ranks received a new sample location at approximately the same time, in the MF case this differs. This difference is caused by the different evaluation times per fidelity.



(a) SF optimisation.

(b) MF optimisation with two fidelities.

Figure 5.8: Rank timeline of an SF and MF optimisation run for the Bohachevsky function with $n_{\text{rank}} = 16$.

In Figure 5.8, the expected behaviour is visible for both the SF and MF case. Thus, in combination with the identified optima, it demonstrates the correct operation of the asynchronous EGO algorithm. However, this method has several limitations, as is evident from the timeline plot.

The first limitation is the extensive downtime of rank 0 between the calculation of new sample locations. This could be resolved by computing a new sample point on rank 0 in addition to the coordination of the optimisation process. This, however, is not possible in the current implementation, which is based on a single thread per rank. This results in the calculation of a point blocking rank 0, which can result in downtime for a part or all of the other ranks, once they are finished calculating their assigned sample point.
Adding multiple threads to at least rank 0 could circumvent the blocking problem, since the rank can switch tasks between computing the sample point and coordinating the optimisation problem. This would, however, require preemptive threading to switch between calculating with a black-box function and coordinating.
This poses another problem, since the optimisation framework is implemented in Python 3.10 using the CPython implementation, which does not support the preemptive threading required for the switching of tasks. This is caused by the Global Interpreter Lock (GIL). The GIL ensures only one thread can access the Python interpreter. Since the GIL cannot be released from the black-box function during the calculation, this effectively locks rank 0. Python 3.13 introduced an experimental free-threaded CPython implementation that could resolve this issue.
The combination of single-thread ranks and the GIL also introduced the second limitation. As shown in Figure 5.8, ranks 1 and 2 compute new sample points when rank 0 finishes the optimisation. This is due to the asynchronous nature, as the convergence criteria are met only after rank 3 returns its new sample point. This means that rank 0 did provide a new sample location to ranks 1 and 2, as the convergence criteria are not met at that point. Since the black-box function cannot be stopped, ranks 1 and 2 will terminate the optimisation only after the new sample points are returned. This could increase wall-clock time, while providing no useful new information to the optimisation process. However, since the end optimisation task for rank 0 consists of finding the optimum of the surrogate and computing the HF function at this optimum, there will likely be no significant increase in wall-clock time, if any.

6

Experimental Setup

This chapter introduces the experimental setup used for all tests performed, starting with an introduction of the infill strategies tested in section 6.1, followed by an introduction of the tested functions in section 6.2. Next, an issue with using computationally cheap functions in combination with the asynchronous Multi-Point (MP) infill strategy and its solutions is highlighted in section 6.3. Next, a discussion of the convergence criteria used and the performance metric used is presented in section 6.4 and section 6.5, respectively. Lastly, the determination of the required number of runs featuring unique random seeds is presented in section 6.6

6.1. Infill Strategies Tested

The proposed asynchronous MP Multi-Fidelity (MF) infill strategy (discussed in section 4.6) is benchmarked against several other infill strategies. The full list of tested infill strategies is:

- ***SF SP KRG Random***: A combination of the random infill strategy, with a Kriging (KRG) surrogate and using the sequential Efficient Global Optimisation (EGO). This serves as a worst-case performance comparison.
- ***MF SP Hierarchical Kriging (HK) Random***: This is expected to be a slight improvement over the Single-Fidelity (SF) Single-Point (SP) KRG Random, as this algorithm uses a MF random infill strategy and the MF HK model.
- ***SF SP KRG EI***: The standard sequential EGO using a KRG surrogate and Expected Improvement (EI) infill strategy.
- ***SF SP KRG GEI***: The sequential EGO using the Generalised Expected Improvement (GEI) infill strategy with $g = 2$.
- ***SF Async KRG EI***: The asynchronous MP version of the standard EGO using EI. This will be run with two different numbers of ranks: 8 and 16.
- ***SF Async KRG GEI***: The version of SF Async KRG EI, utilising the GEI infill strategy. Again, this is run for 8 and 16 ranks and using $g - 2$.
- ***MF SP HK Two-Step***: This is the first MF algorithm, using the Two-Step infill strategy with a Jensen-Shannon distance (JSd)-threshold of 0.7 and EI as the SF infill strategy. This version uses the SP sequential EGO optimiser.
- ***MF Async HK Two-Step***: The last combination is the MP MF combination. Consisting of the Two-Step algorithm using an JSd-threshold of 0.7, EI as the SF infill strategy, and the asynchronous EGO. Again, a test is performed using 8 and 16 ranks. As discussed in section 4.2, this algorithm is expected to outperform the other algorithms.

This list will provide insight into the performance of the proposed infill strategy against a commonly used baseline (SF SP EI or GEI) and will also display the performance difference relative to the combined elements on their own.

6.2. Tested Functions

These infill strategies are benchmarked on a suite of eleven numerical test functions, all of which have a MF implementation in MF2 [21] and range from one to eight-dimensional problems. All the included functions feature low- and high-fidelity functions; the functions and their optima are tabulated in Table 6.1 with a detailed introduction for every function included in Appendix C. All of the included functions are to be minimised, except for the Currin function, which is to be maximised. This problem is converted to a minimisation problem during the optimisation process, by returning $-f$.

Table 6.1: Numerical test functions and their optima.

Function	Dimension	x^*	f^*_{HF}
Forrester	1	0.7572	-6.0207
Bohachevsky	2	$(0, 0)$	0
Booth	2	$(1, 3)$	0
Branin	2	$(-3.7861, 15)$	-333.916
Currin (maximisation)	2	$(0.2167, 0)$	13.7987
Himmelblau	2	$(3, 2)$	0
Six Hump Camelback	2	$(0.0898, -0.7126)$	-1.0316
Park91A	4	$\left(1 \times 10^{-8}, 0, 0, 0\right)$	2.718×10^{-8}
Park91B	4	$(0, 0, 0, 0)$	0.6667
Hartmann6	6	$(0.207, 0.15, 0.4769, 0.2753, 0.3117, 0.6573)$	-3.0425
Borehole	8	$\left(0.05, 5 \times 10^4, 6.307 \times 10^4, 990, 63.1, 820, 1680, 9855\right)$	7.820

6.3. Artificial Delay

The numerical functions are computationally very cheap to evaluate; this will break the logic of the Asynchronous EGO. Since the worker ranks can return the new sample points practically instantly, there is no need for an asynchronous optimisation algorithm. As the optimisation process proceeds, it will at most utilise 2 or 3 worker ranks. To resolve this, a function delay is added to the sample-point calculation. During this time, the evaluation of the new sample point is paused until the function delay is passed. The added delays were determined from the time required to select 16 new sample locations using the Kriging Believer (KB) and Constant Liar (CL) heuristics. This time was used to simulate the time required to select a full set of new samples to address this issue. This time was multiplied by a factor depending on the High Fidelity (HF) or Low Fidelity (LF) function. For the LF function, this was set to 1.2 and for the HF function to 12. The difference in multiplication factor between the LF and HF is used to mimic the time difference for point calculations for real-world LF and HF functions. In real-world engineering functions, this difference could be significantly larger. However, setting a difference in order of magnitude for the delay was not possible for the tests due to the computational limitations and time constraints imposed by the testing environment. The resulting function delays for the numerical test functions are tabulated in Table 6.2.

Table 6.2: Added function delay for numerical test functions.

Function	HF Delay	LF Delay
Forrester	120 s	12 s
Bohachevsky	204 s	20.4 s
Booth	192 s	19.2 s
Branin	228 s	22.8 s
Currin	288 s	28.8 s
Himmelblau	252 s	25.2 s
Six Hump Camelback	444 s	44.4 s
Park91A	600 s	60.0 s
Park91B	1512 s	151.2 s
Hartmann6	2280 s	228.0 s
Borehole	6600 s	660.0 s

6.4. Convergence Criteria

The optimisation runs are performed until one of the several convergence criteria is met. The settings used for the convergence criteria are kept constant for all combinations of infill strategy and function. The complete set of criteria is listed below. Unless otherwise stated, the default settings were used for all tests performed.

- ***Absolute function value convergence***: The absolute difference between the optimum value of the surrogate for the current iteration and n iterations ago. This difference is calculated using $\left|\hat{f}^*_{k-n} - \hat{f}^*_k\right|$, if this is below the set threshold (default is 1×10^{-6}) the convergence criterion is met. n can be set by changing the 'convergence_window', which is set to 5 by default. In the tests using the asynchronous EGO optimisation algorithm, this has been changed to be based on the number of ranks used in the test; $n_{\text{rank}} + 1$. This has been done to ensure the stopping criterion is not met prematurely.
- ***Relative function value convergence***: The relative difference between the optimum value of the surrogate for the current iteration and n iterations ago, with default value 1×10^{-6}. This is calculated using $\frac{\left|\hat{f}^*_{k-n} - \hat{f}^*_k\right|}{\left|\hat{f}^*_{k-n}\right|}$. n is equal to the 'convergence window' set for the absolute function value convergence.
- ***Minimum infill value***: The last infill criterion is the minimum value for the infill value. The implemented infill strategies select a new sample location by maximising the infill value. The corresponding infill value is then returned, which should be above a set threshold, by default 1×10^{-6}. If the returned infill value is below this threshold, the optimisation is deemed converged. In cases where the infill strategy does not return an infill value, such as the random infill strategy used for comparison, this convergence criterion is not applied.

In addition, several stopping criteria based on the available computational budget are included:

- ***Total function calls***: This is the sum of the number of times the objective function for all fidelities is called. If this exceeds the maximum number set, the optimisation stops. This is set to 1000 by default.
- ***Number of HF function calls***: The number of times the HF objective function is called. The default is 150.
- ***Number of iterations***: The number of iterations the optimisation process can take. If this number is exceeded, the optimisation is stopped. By default, this is set to 300.

6.5. Comparison of the Performance

For each optimisation run, several parameters were measured, including the resulting optimum and the elapsed wall-clock time (per iteration and time till solution). A detailed description of all of the saved data can be found in section A.5. The objective function indicates whether the infill method finds the same (local) optimum as the existing SF SP, SF MP, or MF SP infill strategies. If the returned optimum is a worse point, the infill strategy is deemed worse, even if the optimisation is faster in wall-clock time, as it yields a worse optimum.
The wall-clock time indicates the potential speed-up achieved by the combined infill strategy. This parameter is chosen over other timing metrics, such as Central Processing Unit (CPU) time, because fully utilising the CPU across an increasing number of CPUs is difficult, potentially skewing the results to appear more positive than the real-world performance. In addition, wall-clock time is the main performance metric used in High Performance Computing (HPC) benchmarking [56], [57].
These metrics will be combined in an adapted version of the Expected Runtime (ERT) as introduced by Hansen et al. [58] (or ENES by Price [59]). This ERT will be calculated using:

$$\text{ERT} = \frac{\text{Total wall clock time}}{\text{Number of successful runs}}. \tag{6.1}$$

This measure thus gives an insight into the expected performance of the optimisation algorithm. Suppose an algorithm A is twice as fast as algorithm B, but algorithm B reaches the optimum in 4 out of 4 runs, whereas algorithm A does this in 1 out of 4, algorithm B will outperform algorithm A. For this measure,

first, a definition of when a run is successful must be established, which is set to a threshold of the function value of the returned optimum relative to the known optimum. This threshold was set to:

$$\left| f^*_{\text{opt}} - f^*_{\text{known}} \right| \leq \text{atol} + \text{rtol} \left| f^*_{known} \right|. \tag{6.2}$$

With the absolute tolerance (atol) set to 0.01 and the relative tolerance (rtol) set to 0.01. If no runs succeed, the ERT is undefined, and the algorithm will receive the lowest rank when comparing performance. In case of a tie a fractional rank is assigned, as outlined in the comparison method introduced by Demšar [60].
Next, in accordance with this method, first, a Friedman test is performed. This test will show if a significant difference in performance exists between the set of optimisers tested. If the resulting $p < 0.05$, the null hypothesis, that the algorithms perform equally, is rejected. This proves that there is a significant difference in performance between the algorithms.
If the null hypothesis is rejected, a post-hoc Nemenyi test [60] is performed. This test compares the assigned ranks of all algorithms with the ranks of all other algorithms, from which a critical difference diagram can be constructed. This diagram shows the average ranks of all the algorithms over the set of test functions. A horizontal bar connecting different optimisers indicates there is no significant difference in performance for the connected set, based on the assigned ranks.
This critical difference diagram, however, is based only on the assigned ranks, not on the actual magnitude difference in the ERT. This analysis is followed by the creation of a data profile [61]. This diagram shows the fraction of optimisation problems solved by an optimiser over time. Thus, complementing the critical difference diagram by showing performance differences in terms of wall-clock time.

6.6. Random Seeds

The benchmark runs start from a pseudo-random state, based on the random seed used. This random seed is used in the selection of the initial sample points using Latin Hypercube Sampling (LHS). These initial starting conditions can have a profound effect on the performance metrics, as a certain starting condition can be easier to solve than another. What a favourable starting set of samples is, can differ per infill strategy.

In addition, the random seeds control the internal processes in the optimisation, which include a random process, namely the optimisation of the log-likelihood in the training of the HK or KRG surrogate or in finding the maximum or minimum of the infill strategy. A more detailed explanation of which part of the implementation depends on the random seed is included in section A.6 For these two reasons, each combination of function-infill strategy is benchmarked using a set of random seeds. The set of random seeds is constant across combinations, which results in the infill strategies being paired for comparison.

This raised the question of how many random seeds should be used for every combination. The main performance metric used is ERT, which divides the total elapsed time by the number of successful runs, both of which depend on the random seeds run. Thus, the number of random seeds used needs to ensure the ERT is a reliable metric. This was calculated by utilising a pilot experiment. First, a set of 20 random seeds was benchmarked. Next, the standard error of the ratio of the proposed infill strategy's ERT compared to the ERT of the comparison infill strategies is retrieved from this pilot experiment using paired bootstrapping [62].

Since the proposed infill strategy is expected to significantly outperform the SF SP baseline infill strategy as well as the SP MF and MP SF infill strategies, the sizing of the number of random seeds was done for a relatively large difference to be detected. This difference was set to 40%. Tightening the difference that can be detected will result in an increase in the required number of random seeds, which is unrealistic within the available computational budget and time frame. The difference to be detectable was used in a power analysis in combination with a significance level of $\frac{0.05}{3}$ (the 0.05 divided by the number of comparisons made) and power of 0.8 [63]. From this power analysis, the minimum number of random seeds required is calculated.

A second source of variation in the run time is the hardware timing noise induced by the HPC environment. This, however, was deemed insignificant in relation to the evaluation time of the benchmark functions with the added artificial delay as discussed in section 6.3.

The pilot experiment showed that several benchmark functions are expensive to evaluate due to the added artificial delay. This makes it impossible to run the required number of random seeds for these functions within the available computational budget and time frame. These functions will only be used to get a qualitative insight into the behaviour of the optimisation algorithms. These functions are also excluded from the overall significance analysis as discussed in section 6.5. The determined functions that are too computationally expensive are:

- Park91B
- Hartmann6
- Borehole

The calculation of the required number of random seeds resulted in Table 6.3 as the most driving comparisons.

Table 6.3: Limiting comparisons for the required number of random seeds

Function	Comparison	Required number of seeds
Bohachevsky	SP MF	209
Bohachevsky	SP SF	129
Bohachevsky	MP SF	72
Branin	MP SF	29
Branin	SP MF	28

While running the analysis, statistically comparing the GEI based algorithms proved to require large numbers of random seeds, in the order of hundreds. This was deemed not possible given the constraints on the computational power and timeframe; for this reason, these comparisons will not be used in the statistical evaluation of the results. For the same reason, no direct comparison using the Bohachevsky function can be made. These elements will all be used to give a qualitative insight into the performance. Taking these omissions into account, the required number of random seeds for the functions was found to be 29.

7

Test Problems

The benchmark tests on the numerical test problems were performed as described in chapter 6. This chapter introduces the results of these tests, starting with the overall results in section 7.1. Following the overall results, a more detailed discussion of the problems by number of dimensions is presented in section 7.2, section 7.3, section 7.4, section 7.5 and section 7.6, for the one-, two-, four-, six- and eight-dimensional problems respectively. Next, the performance difference between Expected Improvement (EI) and Generalised Expected Improvement (GEI) based strategies is examined in section 7.7, concluding with the main findings in section 7.8.

7.1. Overall Results

The first results presented are the combined set of numerical test problems. The full set of results for the EI based infill strategies is presented in Table 7.1 and for the GEI based infill strategies in Table 7.2. The algorithms and functions considered in the statistical analysis are ranked based on their Expected Runtime (ERT). As discussed in section 6.6, this excludes the GEI based infill strategies and the Bohachevsky, Park91B, Hartmann6, and Borehole functions. The remaining infill strategies are ranked by their ERT for the remaining functions, according to the procedure described in section 6.5. Next, the Friedman test is performed, resulting in $p < 0.001$, rejecting the null hypothesis. This signifies a significant difference in performance. Following the Friedman test, the critical-difference diagram was created using the post-hoc Nemenyi test. The resulting Critical-Difference (CD) diagram is figured in

Table 7.1: The ERT (HH:MM:SS) of the EI based infill strategies, with the corresponding ranks for functions used in the statistical analysis.

Function	SF SP KRG Random	MF SP HK Random	SF SP KRG EI	SF Async KRG EI n=8	SF Async KRG EI n=16	MF SP HK Two-Step EI	MF Async HK Two-Step EI n=8	MF Async HK Two-Step EI n=16
Forrester	00:53:44 (7)	00:56:49 (8)	00:13:49 (5)	00:05:58 (3)	00:05:33 (1)	00:14:40 (6)	00:06:05 (4)	00:05:53 (2)
Bohachevsky	10:13:48	23:09:15	01:52:18	00:24:37	00:15:46	03:35:05	00:42:08	00:24:53
Booth	07:43:14 (7)	12:32:15 (8)	00:25:53 (5)	00:08:59 (2)	00:08:59 (1)	00:38:08 (6)	00:12:44 (4)	00:10:33 (3)
Branin	05:44:32 (8)	04:16:27 (7)	00:33:05 (5)	00:13:38 (2)	00:10:54 (1)	00:56:37 (6)	00:16:06 (4)	00:14:40 (3)
Currin	11:21:53 (7)	14:27:10 (8)	01:28:43 (6)	00:35:10 (4)	00:27:50 (2)	01:18:01 (5)	00:28:08 (3)	00:22:12 (1)
Himmelblau	10:25:49 (7)	14:36:19 (8)	02:51:56 (5)	00:37:04 (2)	00:25:41 (1)	03:24:23 (6)	00:49:55 (4)	00:46:29 (3)
Six hump camelback	18:07:56 (7)	23:27:08 (8)	04:01:08 (5)	00:54:42 (3)	00:38:22 (1)	05:34:08 (6)	01:00:24 (4)	00:43:21 (2)
Park91a	25:11:07 (7)	33:11:56 (8)	01:08:51 (6)	00:26:16 (3)	00:22:24 (1)	01:08:22 (5)	00:29:03 (4)	00:23:59 (2)
Park91b	60:50:42	65:12:00	03:47:33	01:19:01	01:05:45	04:09:38	01:54:52	01:15:59
Hartmann6	No successful run	No successful run	52:52:29	13:05:46	04:59:59	66:26:03	08:14:33	11:02:51
Borehole	661:26:31	604:51:29	25:26:35	08:04:10	05:13:17	433:47:55	No data	No data
Avg Rank	7.14	7.86	5.29	2.71	1.14	5.71	3.86	2.29

Table 7.2: The ERT (HH:MM:SS) for the GEI based infill strategies.

Function	SF SP KRG Random	MF SP HK Random	SF SP KRG GEI	SF Async KRG GEI n=8	SF Async KRG GEI n=16	MF Async HK Two-Step GEI n=8	MF Async HK Two-Step GEI n=16
Forrester	00:53:44	00:56:49	00:12:01	00:04:02	00:04:02	00:06:00	00:05:46
Bohachevsky	10:13:48	23:09:15	06:10:43	00:46:47	00:25:58	01:50:41	04:13:16
Booth	07:43:14	12:32:15	00:37:39	00:07:44	00:06:56	00:22:11	00:17:33
Branin	05:44:32	04:16:27	00:40:52	00:18:07	00:09:25	00:24:13	00:13:45
Currin	11:21:53	14:27:10	01:25:54	00:28:02	00:19:56	00:41:59	00:22:30
Himmelblau	10:25:49	14:36:19	02:51:48	00:47:27	00:34:51	01:42:21	01:00:15
Six hump camelback	18:07:56	23:27:08	04:35:08	01:28:26	00:38:43	01:07:52	00:59:10
Park91a	25:11:07	33:11:56	02:06:36	00:24:33	00:21:03	00:34:28	00:30:29
Park91b	60:50:42	65:12:00	05:29:41	01:31:42	01:07:45	01:29:08	02:23:58
Hartmann6	No successful run	No successful run	53:30:43	No successful runs	43:06:18	45:14:37	No successful run
Borehole	661:26:31	604:51:29	32:56:49	06:33:03	06:10:20	11:49:29	06:47:03

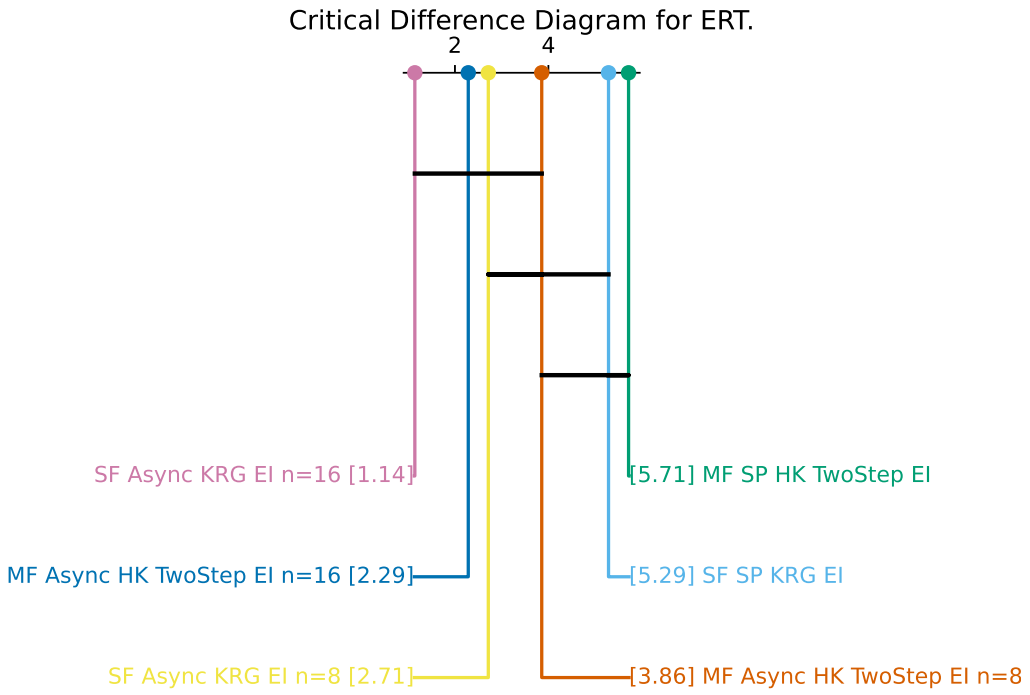


Figure 7.1: The critical difference diagram for the non-random infill strategies on the set of numerical test functions.

Table 7.1 shows that consistently the random infill strategies perform worst. A drastic increase in performance can be observed moving to the Single-Point (SP) infill strategies. Between the SP infill strategies the Single-Fidelity (SF) SP EI outperforms the Multi-Fidelity (MF) infill strategy. This is against the expectations from section 4.2. The Currin maximisation problem is a notable exception here, where the MF infill strategy outperforms the SF infill strategy. This behaviour is again shown for the Multi-Point (MP) infill strategies, where the SF infill strategy consistently outperforms their equivalent MF variant. Again, with the exception of the Currin maximisation problem. The MP do outperform the SP infill strategies, with the relative difference between the $n = 16$ and $n = 8$ variants differing per problem. The variant with more computational power available $n = 16$, outperforms the $n = 8$ variant on every function, but the difference is small for the two computationally easiest optimisation problems: Forrester and Booth.

This ranking of the infill strategies is also apparent from the CD diagram in Figure 7.1. This plot shows the average rank of the infill strategies over the functions used in the analysis. This shows the SF MP strategies outperforming their SF counterparts. The MF strategies are followed by the SF SP and lastly the MF SP infill strategies. The CD diagram also shows that based on the ERT ranks there is no significant difference in performance between all MP infill strategies, as denoted by the black horizontal bar in the diagram. This is also the case for the SF MP $n = 8$, MF MP $n = 8$ and SF SP strategies group and the MF MP $n = 8$, SF SP and MF SP group. This relatively large number of infill strategies that perform non-significantly differently is due to the small set of seven functions used in the analysis. Expanding this set of test functions will allow the critical-difference (the difference in ERT for which two infill strategies are not significantly different) to be reduced.

The CD diagram, however, only displays the performance based on the ranking of the infill strategies, and does not take into account the difference in magnitude of the ERT. To further investigate the performance difference, a data profile has been created and is displayed in Figure 7.2. This data profile has been created using the set of functions, with a sufficient number of random seed runs as discussed earlier and in section 6.6. This data-profile shows the fraction of problems from the set of seven test functions a infill strategy can solve after a given elapsed time. This shows a large jump in performance from the SP to the MP infill strategies. The difference between the SF and MF is less pronounced, but

does show that the SF outperforms the MF.

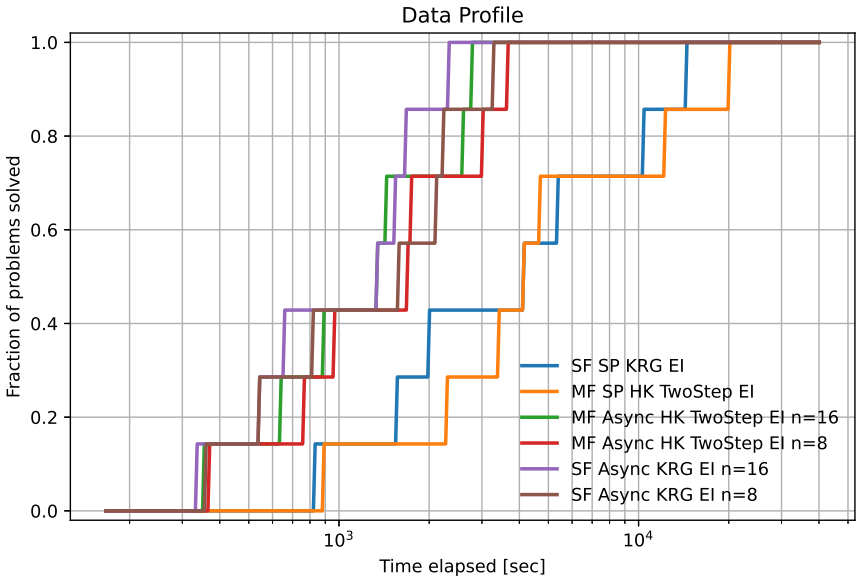


Figure 7.2: The data profile for the optimisers on the numerical test functions.

Lastly, the performance of the infill strategies has been compared with the baseline SP, SF EI infill strategy. The geometric mean of the difference in ERT of the seven functions is visualised in Figure 7.3. From this diagram it is evident the MP strategies outperform the baseline infill strategy. The difference is above the 40% detectable threshold as discussed in section 6.6, thus the difference in performance compared to the baseline is significant. It can also be seen that overall the SF infill strategies outperform their equivalent MF versions, these differences are however not above the 40% threshold and can thus not be significantly detected given the used number of random seeds.

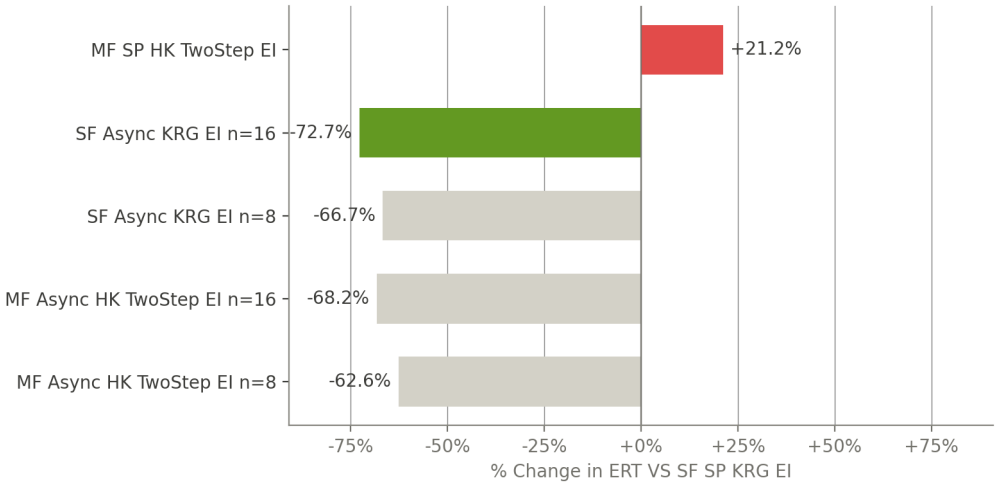


Figure 7.3: The geometric mean performance difference for the seven numerical test functions (lower is better).

Next, some notable results per function are discussed, starting with the 1-dimensional Forrester function. A complete set of the convergence, resulting optimum, function calls, and elapsed time plots for every function is included in Appendix D. In addition to these figures the success rate, average elapsed time and average elapsed time per successful run tables are included in this appendix.

7.2. 1-Dimensional Problem

A single 1-dimensional problem was investigated, the Forrester function. This function was solved by the majority of infill strategies in 100% of the runs. The MF infill strategies achieved a slightly lower success rate, while staying above 85%, with the MF MP GEI being an exception at 56%. This means that the difference in ERT is thus mostly dictated by the difference in elapsed time. The elapsed time for all infill strategies is illustrated in the raincloud plot [64] in Figure 7.4. This raincloud plot displays the box-plot of the different runs, with the scatter plot of the runs and the probability density plot of the runs included on the left and right side, respectively.

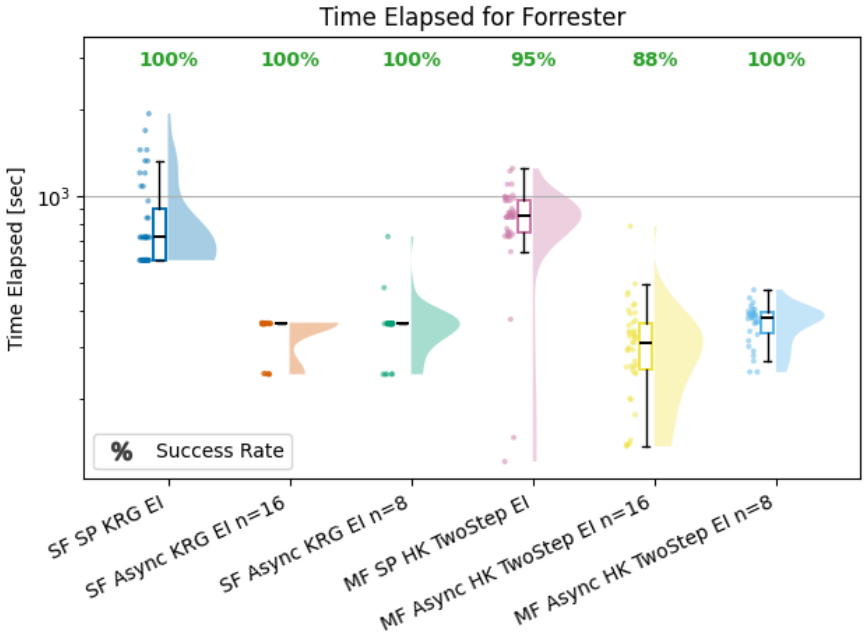
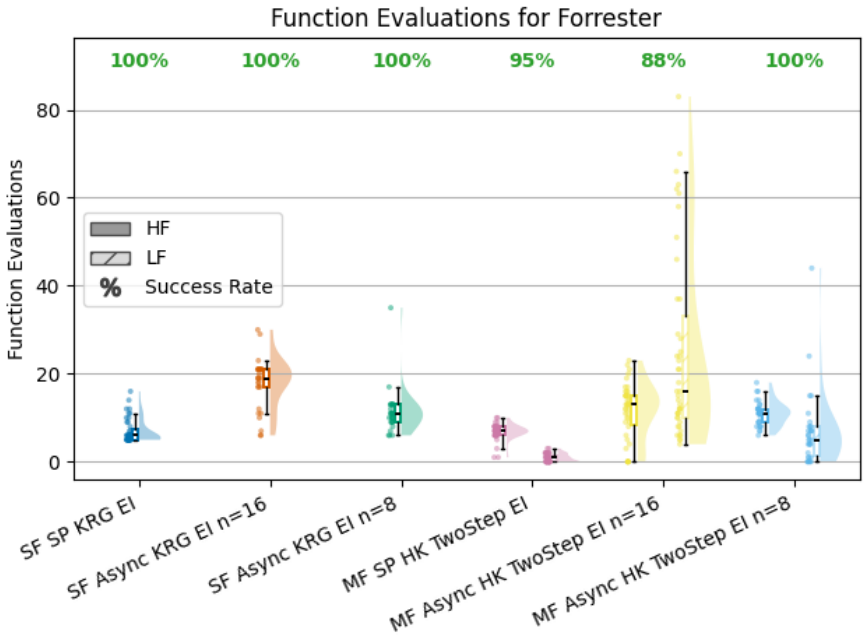
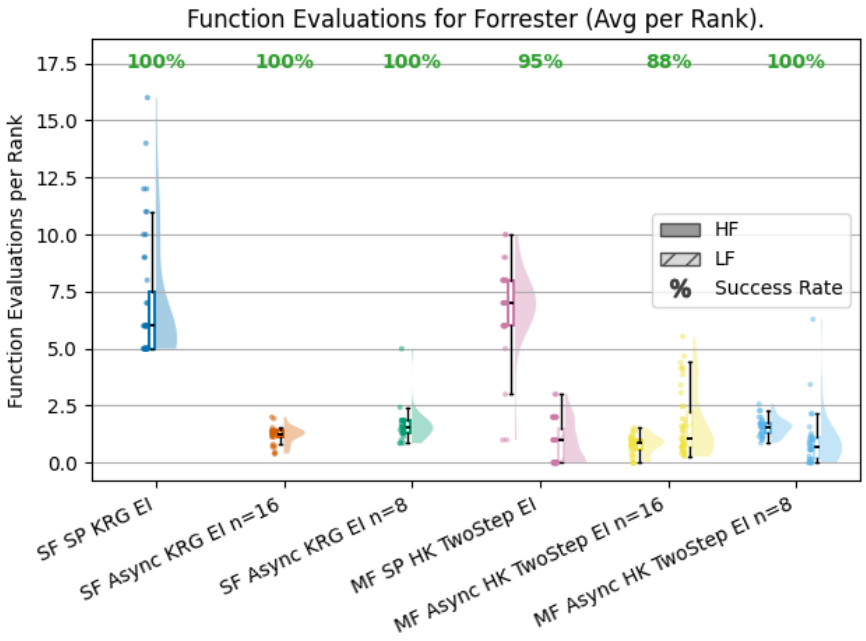


Figure 7.4: Elapsed time Forrester function.

This problem also showed that the number of function calls (and thus the number of sample points) differs substantially between the methods. In Figure 7.5a, it can be seen that significantly more sample points are added for the async and random method compared to the SP methods. This is the cause of the increased elapsed time for the random optimiser. For the async methods, this, however, is not the case. This is caused by async methods computing points in parallel, thereby reducing the overall wall-clock time. For wall-clock time, the number of function calls per rank is more meaningful. This plot is included in Figure 7.5b. This figure also illustrates why the MF MP strategies perform worse compared to the SF MP infill strategies. This is caused by the MF MP infill strategy sampling approximately the same number of High Fidelity (HF) sample points, not leading to a reduction in computation time, while Low Fidelity (LF) samples are added, increasing the computation time and thus ERT.



(a) The total function evaluations.



(b) The function evaluations per worker rank (per infill strategy left is the HF samples with LF on the right).

Figure 7.5: Function evaluations for the Forrester function.

7.3. 2-Dimensional Problems

The set of 2D problems consists of 6 functions: Bohachevsky, Booth, Branin, Currin (maximisation), Himmelblau and Six Hump Camelback. On five of these functions, the SF MP strategy outperforms the MF MP infill strategy; however, on the Currin problem, the MF achieves a lower ERT. This performance is caused by the reduction in elapsed time as a result of a reduction in HF function calls per rank, as illustrated in Figure 7.6. This difference in ERT is reduced by the small difference in success rate between the infill strategies. This difference is visible in the returned optima as illustrated in Figure 7.7.

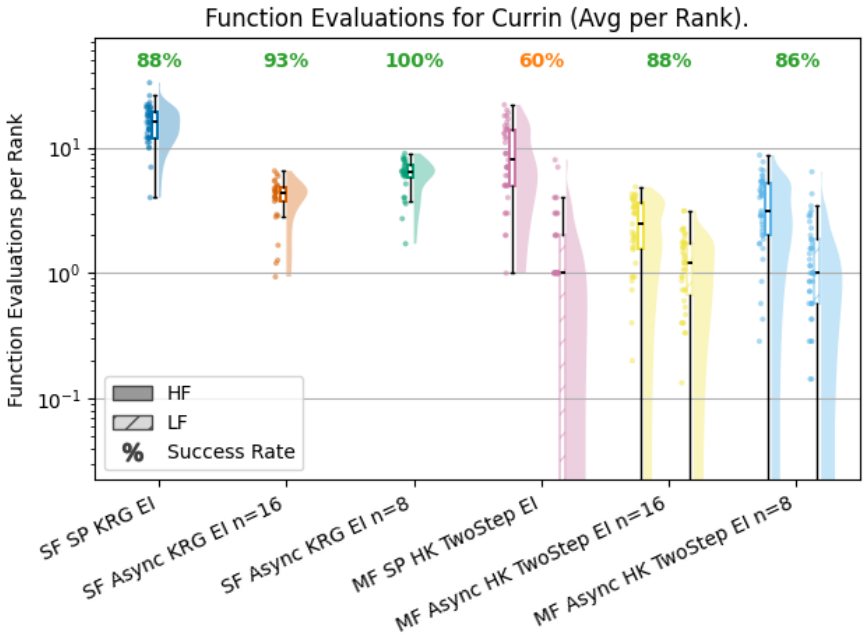


Figure 7.6: Function evaluations per rank for the Currin function.

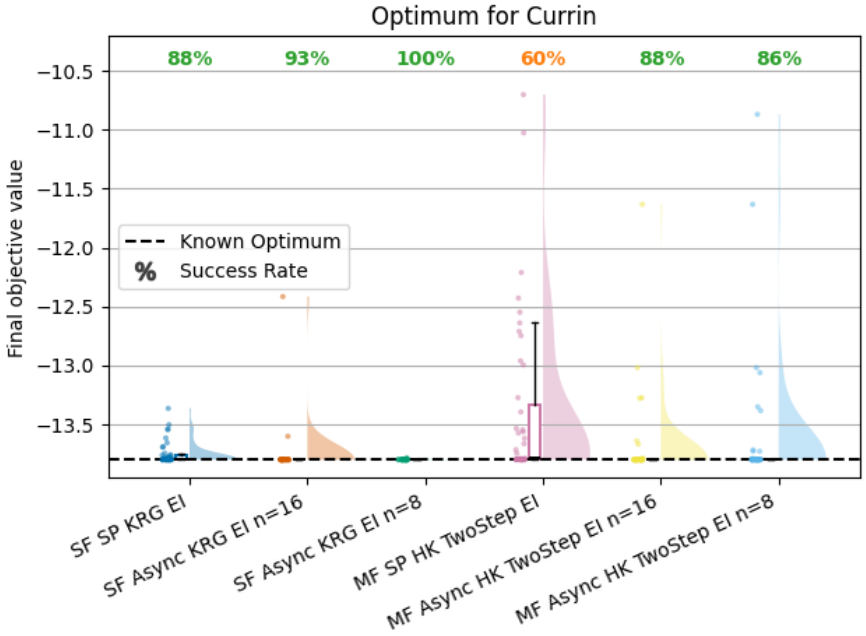


Figure 7.7: The returned optimum per rank for the Currin function (the function value is scaled by -1, to convert the maximisation problem to a minimisation problem).

The trend from the Forrester function, as discussed in section 7.2, is continued for the other 2-dimensional problems. The behaviour again is that the MP infill strategies outperform the SP infill strategies by the reduction in function evaluations per rank. The MF infill strategies do not show a reduction in HF function evaluations, leading to an increase in the elapsed time per optimisation caused by the introduction of the LF samples.

7.4. 4-Dimensional Problems

The 4-dimensional Park91A and Park91B functions again showed the same difference in ERT as for the earlier discussed functions. The MF showed the same behaviour as the number of HF samples did not reduce, while LF samples were added. However, for both the Park91A and Park91B, there was a large difference in success rate between the SF and MF infill strategies. This can be visualised by the returned optimum for both functions, Figure 7.8 for Park91A and Figure 7.9 for Park91B. In both cases the returned optima do not show two or more groupings of runs, suggesting the unsuccessful runs hit stopping criteria prematurely, instead of reaching a local minimum.

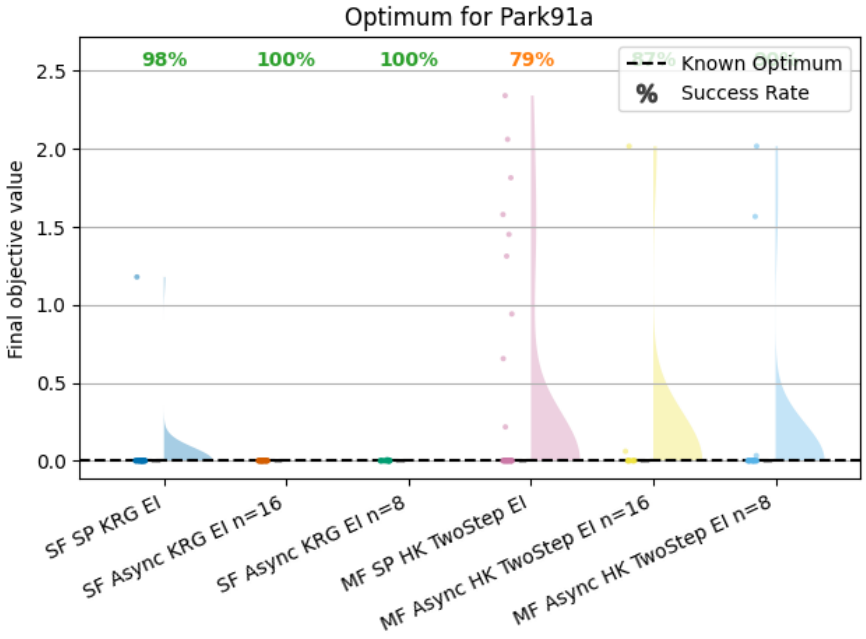


Figure 7.8: The returned optimum for the Park91A function.

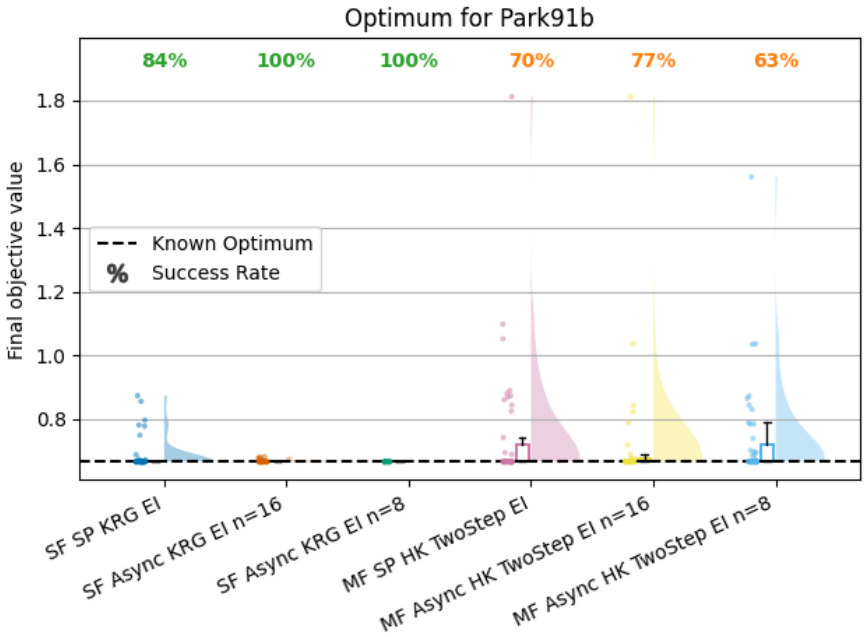


Figure 7.9: The returned optimum for the park91B function.

7.5. 6-Dimensional Problems

The 6-dimensional Hartmann6 posed a challenging problem for most optimisers, with low success rates for all infill strategies. This function was, however, not taken into account in the significance testing due to the large computational time required for completing the required number of random seeds. This is driven by the apparent hard nature of the problem to be optimised. For the MP the additional limitation of the computational budget available in combination with the large artificial pause time required further reduced the possible random seeds to be completed. This could skew the results and no significant conclusion can be drawn from them. The low success rate is visualised by the returned optima in Figure 7.10.

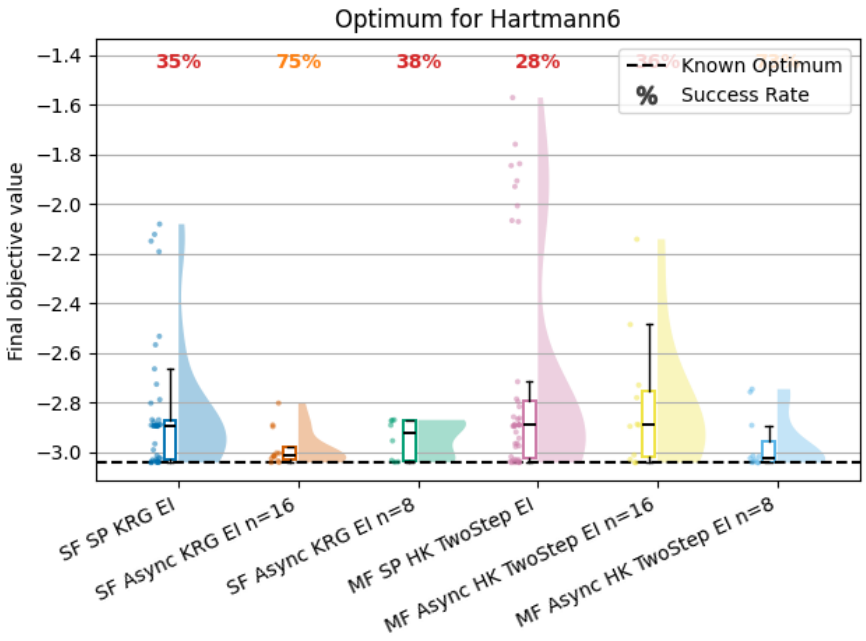


Figure 7.10: The returned optimum for the Hartmann6 function.

7.6. 8-Dimensional Problems

The test problem with the highest dimension is the Borehole problem. This problem suffers from the same limitations as the Hartmann6 function as discussed in section 7.5. This function shows the largest reduction in performance for the MF SP infill strategy compared to the SF SP infill strategy. In addition to the difference in success rate changing the ERT, the difference is driven by the elapsed time as illustrated in Figure 7.11. This in turn is caused by the large increase in HF samples added by the MF infill strategy; Figure 7.12. The MF SP infill strategy, however, completed a significantly smaller number of runs, skewing the results. The runs with random seeds completed are however completed by both infill strategies. The SF SP does not have outliers in function evaluations and elapsed time as severe as the MF SP runs. This suggests that the MF SP's performance will be similar when adding more random seeds. As mentioned earlier, however, not enough runs have been completed to form conclusions.

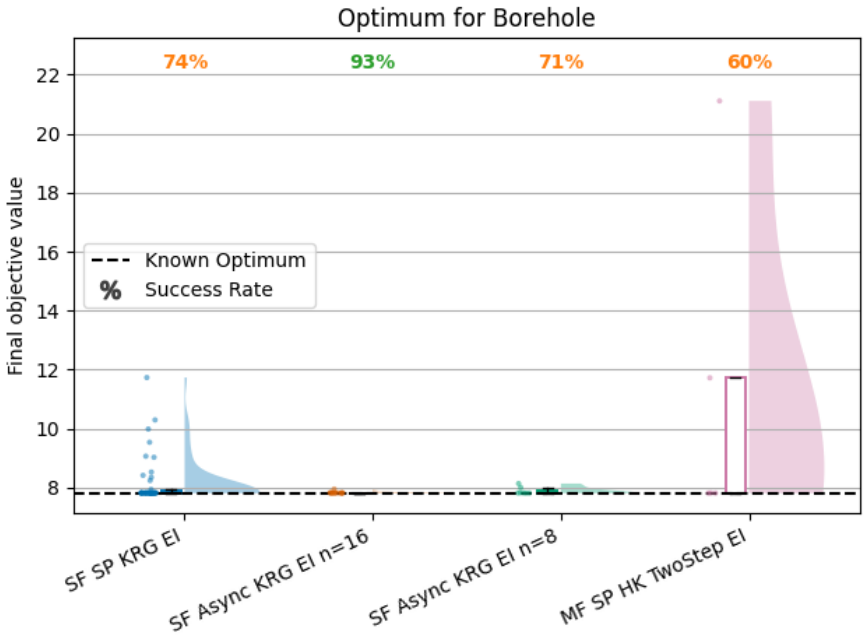


Figure 7.11: The time elapsed for the Borehole function.

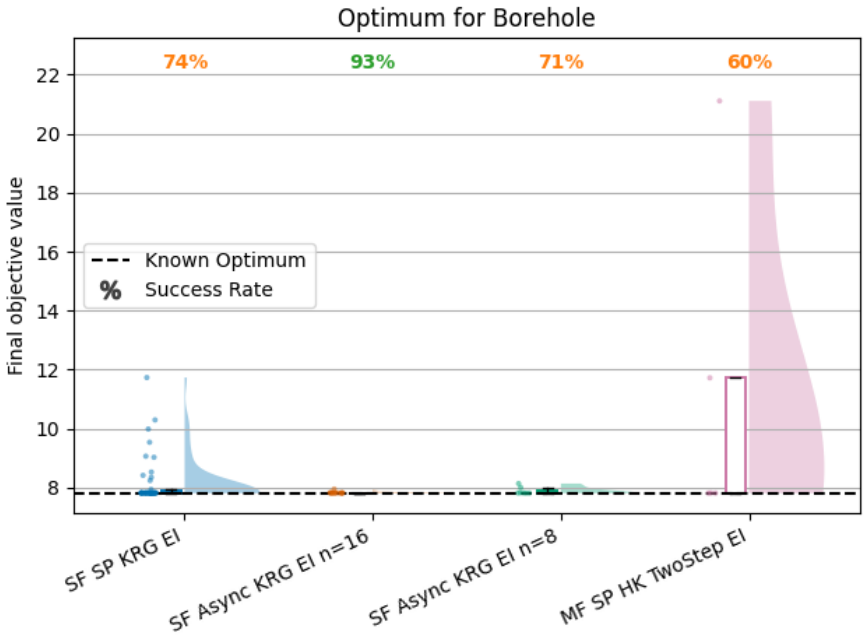


Figure 7.12: The function evaluations per rank for the Borehole function.

7.7. Comparing EI and GEI

The test functions were also examined using GEI based infill strategies. As discussed in section 6.6 these infill strategies required too large a number of random seed runs for the available computational budget and time frame to result in significant relevant results. Therefore the results displayed here are illustrative, but no significant conclusion can be drawn. The GEI based infill strategies show the same behaviour in performance difference as the EI based strategies. The MP infill strategies increase the performance over the baseline, where adding the MF aspect reduces the performance. In addition as visualised in Figure 7.13, the GEI based infill strategies are outperformed by the equivalent EI based infill strategies. This is largely caused by the reduced success rate for the GEI based infill strategies.

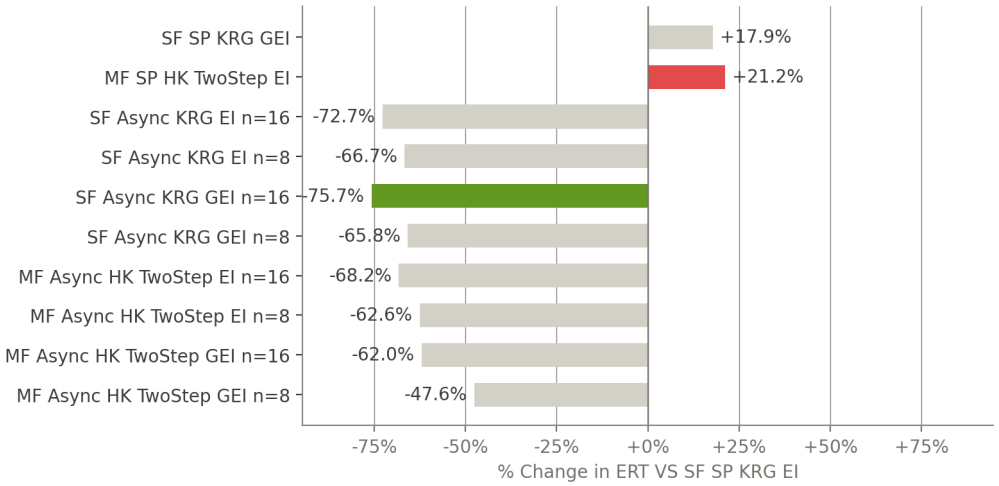


Figure 7.13: The geometric mean performance difference for the seven numerical test functions for the EI and GEI based infill strategies (lower is better).

7.8. Conclusion

As illustrated in the previously discussed results the MP infill strategies significantly outperform the SP infill strategies. The SF based infill strategies display better performance than the MF based infill strategies for most functions. This difference is, however, smaller than the threshold of the significantly detectable difference of 40%. This reduction in performance has been shown to be caused by the addition of LF samples, while the number of HF samples required for convergence did not reduce. To determine whether the LF function influences this behaviour the Forrester function was tested using the SP MF EI infill strategy for a set of different LF functions. The results of which are discussed in chapter 8.

8

Different Low-Fidelity Forrester

From the results for the numeric test functions discussed in chapter 7, it was concluded that the Multi-Point (MP) strategies outperform their equivalent Single-Point (SP) counterparts. The Multi-Fidelity (MF) aspect, however, reduces the performance compared to the Single-Fidelity (SF) version. This behaviour was caused by the addition of Low Fidelity (LF) sample points to be evaluated, while the number of High Fidelity (HF) sample points did not reduce. To investigate the role the LF function plays, additional tests were performed with a set of different LF Forrester functions. These functions are introduced in section 8.1 followed by a discussion of the results in section 8.2.

8.1. Low-Fidelity Forrester Function Definitions

To define several new LF Forrester functions, first, a base Forrester function was defined:

$$f_b(x, A, B, C) = (6(x + A) - 2)^2 (1 + B) \sin(12(x + A) - 4) + C. \tag{8.1}$$

By setting A, B, C to 0, the HF Forrester function as defined in Equation (8.2) is retrieved:

$$f_{HF}(x, 0, 0, 0) = (6x - 2)^2 \sin(12x - 4). \tag{8.2}$$

The LF variations examined are tabulated in Table 8.1, the resulting equations and visualisation are presented in Appendix E.

Table 8.1: LF Forrester Definitions.

Identifier	A	B	C
Horizontal offset 0.5	0.5	0	0
Horizontal offset -0.5	−0.5	0	0
Amplitude change 5	0	5	0
Amplitude change -5	0	−5	0
Vertical offset 5	0	0	5
Vertical offset -5	0	0	−5
Combined change 5	0.5	5	5
Combined change -5	−0.5	−5	−5

In addition to these altered LF formulations, three linearly interpolated LF functions were examined. These functions were constructed by dividing the bounds into n equal-width sections, between whose edges a linearly interpolated function was generated. This was done for $n = 2$, $n = 5$ and $n = 10$. The $n = 5$ case is illustrated in Figure 8.1; the other visualisations can be found in Appendix E.

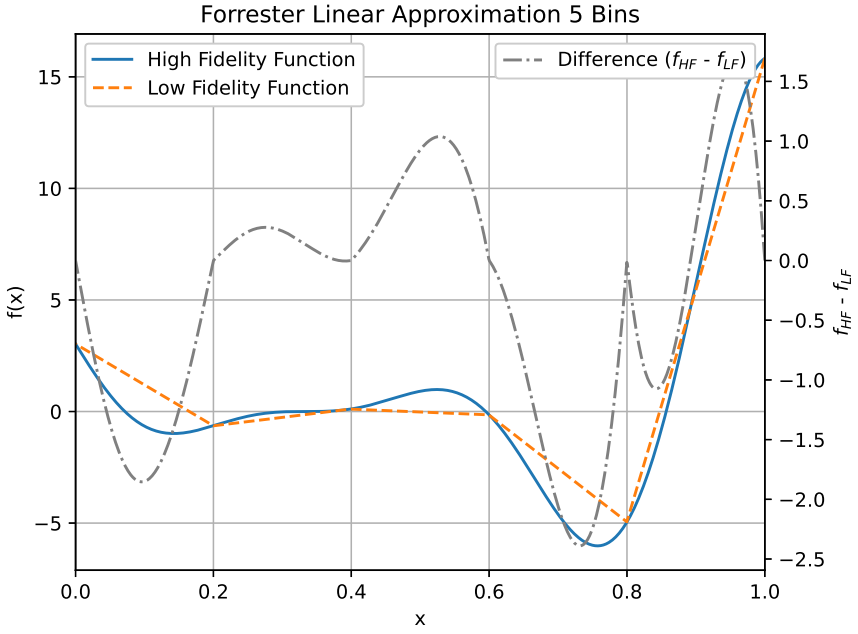


Figure 8.1: Linearly interpolated Forrester function with $n = 5$ sections.

8.2. Results

The set of different LF functions was tested using the MF SP HK Two-Step EI infill strategy. As a reference strategy the baseline SF SP KRG EI was used. Since this reference strategy is an SF approach, the results do not differ for the changing LF function definitions. The resulting Expected Runtime (ERT) for every combination is tabulated in Table 8.2. From this table it is evident that for every LF function definition the MF infill strategy reduces performance. Table 8.3 shows the tabulated mean function calls per optimisation run. This table shows that the number of HF samples is reduced slightly for the MF infill strategy for most LF functions. However, the success rate of the MF is slightly lower resulting in increased ERT. This difference in success rate is caused by some MF runs hitting the minimum infill value convergence criterion, as some combination of infill samples resulted in low variance for the Hierarchical Kriging (HK) surrogate model.

Table 8.2: The ERT (HH:MM:SS) and corresponding rank for the differing Forrester LF functions. Note that the results for the SF SP infill strategy are equal, since this is solely using the HF function.

Function	SF SP KRG EI	MF SP HK Two-Step EI
Forrester	00:13:49 (1)	00:14:40 (2)
Forrester horizontal offset 0.5	00:13:49 (1)	00:14:57 (2)
Forrester horizontal offset -0.5	00:13:49 (1)	00:14:37 (2)
Forrester amplitude change 5	00:13:49 (1)	00:15:38 (2)
Forrester amplitude change -5	00:13:49 (1)	00:15:00 (2)
Forrester vertical offset 5	00:13:49 (1)	00:15:07 (2)
Forrester vertical offset -5	00:13:49 (1)	00:15:37 (2)
Forrester combined change 5	00:13:49 (1)	00:15:12 (2)
Forrester combined change -5	00:13:49 (1)	00:15:07 (2)

Continued on next page

Table 8.2 – continued from previous page

Function	SF SP KRG EI	MF SP HK Two-Step EI
Forrester linear approximation 2 bins	00:13:49 (1)	00:15:32 (2)
Forrester linear approximation 5 bins	00:13:49 (1)	00:14:22 (2)
Forrester linear approximation 10 bins	00:13:49 (1)	00:14:41 (2)
Avg Rank	1.00	2.00

Table 8.3: Mean function evaluations $\pm$ STD.

Function	SF SP KRG EI	MF SP HK Two-Step EI
Forrester	HF: 6.91 ± 2.75, LF: -	HF: 6.79 ± 1.76, LF: 0.81 ± 0.92
Forrester horizontal offset 0.5	HF: 6.91 ± 2.75, LF: -	HF: 6.16 ± 2.05, LF: 1.37 ± 1.63
Forrester horizontal offset -0.5	HF: 6.91 ± 2.75, LF: -	HF: 6.74 ± 1.67, LF: 1.21 ± 1.11
Forrester amplitude change 5	HF: 6.91 ± 2.75, LF: -	HF: 5.63 ± 2.23, LF: 0.88 ± 1.22
Forrester amplitude change -5	HF: 6.91 ± 2.75, LF: -	HF: 5.56 ± 2.19, LF: 1.12 ± 1.32
Forrester vertical offset 5	HF: 6.91 ± 2.75, LF: -	HF: 5.63 ± 2.26, LF: 0.63 ± 0.84
Forrester vertical offset -5	HF: 6.91 ± 2.75, LF: -	HF: 5.84 ± 2.26, LF: 0.65 ± 1.08
Forrester combined change 5	HF: 6.91 ± 2.75, LF: -	HF: 5.86 ± 2.26, LF: 0.53 ± 1.00
Forrester combined change -5	HF: 6.91 ± 2.75, LF: -	HF: 6.84 ± 1.68, LF: 0.91 ± 1.41
Forrester linear approximation 2 bins	HF: 6.91 ± 2.75, LF: -	HF: 7.02 ± 1.53, LF: 0.98 ± 1.09
Forrester linear approximation 5 bins	HF: 6.91 ± 2.75, LF: -	HF: 6.26 ± 2.15, LF: 1.23 ± 1.34
Forrester linear approximation 10 bins	HF: 6.91 ± 2.75, LF: -	HF: 5.58 ± 2.25, LF: 1.21 ± 1.21

However, tabulating the mean function calls for the successful runs (Table 8.4), shows a similar behaviour as seen in chapter 7. This table shows a small change in HF samples (both an increase and a decrease, depending on the LF function), while LF can be added. This in turn leads to longer elapsed times. Given the lower success rate and the longer elapsed time for the successful runs, this leads to an increase in ERT for the MF infill strategies

Table 8.4: Mean function evaluations $\pm$ STD for the successful optimisation runs.

Function	SF SP KRG EI	MF SP HK Two-Step EI
Forrester	HF: 6.91 ± 2.75, LF: -	HF: 7.02 ± 1.42, LF: 0.83 ± 0.93
Forrester horizontal offset 0.5	HF: 6.91 ± 2.75, LF: -	HF: 6.84 ± 1.24, LF: 1.57 ± 1.67
Forrester horizontal offset -0.5	HF: 6.91 ± 2.75, LF: -	HF: 7.00 ± 1.23, LF: 1.27 ± 1.10
Forrester amplitude change 5	HF: 6.91 ± 2.75, LF: -	HF: 6.75 ± 1.25, LF: 1.19 ± 1.29
Forrester amplitude change -5	HF: 6.91 ± 2.75, LF: -	HF: 6.58 ± 1.30, LF: 1.45 ± 1.33
Forrester vertical offset 5	HF: 6.91 ± 2.75, LF: -	HF: 6.67 ± 1.36, LF: 0.73 ± 0.86
Forrester vertical offset -5	HF: 6.91 ± 2.75, LF: -	HF: 6.88 ± 1.37, LF: 0.85 ± 1.16
Forrester combined change 5	HF: 6.91 ± 2.75, LF: -	HF: 6.85 ± 1.29, LF: 0.65 ± 1.08
Forrester combined change -5	HF: 6.91 ± 2.75, LF: -	HF: 7.20 ± 1.05, LF: 0.97 ± 1.44
Forrester linear approximation 2 bins	HF: 6.91 ± 2.75, LF: -	HF: 7.28 ± 1.26, LF: 1.05 ± 1.09
Forrester linear approximation 5 bins	HF: 6.91 ± 2.75, LF: -	HF: 6.74 ± 1.58, LF: 1.31 ± 1.36
Forrester linear approximation 10 bins	HF: 6.91 ± 2.75, LF: -	HF: 6.50 ± 1.48, LF: 1.50 ± 1.19

As discussed in section 4.2, the MF strategies were expected to outperform their SF counterparts, since the number of HF samples required was expected to be reduced, caused by the LF aiding in determining the trend of the function, which could be corrected by the addition of some HF samples. When plotting the optimisation process for a MF SP Two-Step Expected Improvement (EI) infill strategy on the Forrester function (included in Figure 8.2), it is visible that the infill strategy add multiple points around the optimum. This optimisation run first adds an LF around the optimum. Training the HK on this new set of sample points will reduce the variance at this location, but the variance will not be zero. As the sample point added is LF the HK does not treat it as a true point, as it would do for an HF point. The

surrogate predictor in turn does not pass directly through the LF sample point(s). This non-zero variance near the optimum, results in the region being exploited by the infill strategy. This could result in the infill strategy reselecting the previous infill location at the lower fidelity level, since the LF function is deemed sufficiently accurate by the Two-Step infill strategy. The implementation, as discussed in Appendix A, recognises that this location has already been sampled at the LF fidelity level and will sample the point one fidelity level higher (in this case HF). This is visible in iterations 5, 6 and 7 in Figure 8.2. This new HF sample point does reduce the variance at the point to 0, reducing the infill value. This process thus essentially adds a redundant LF point near the optimum.

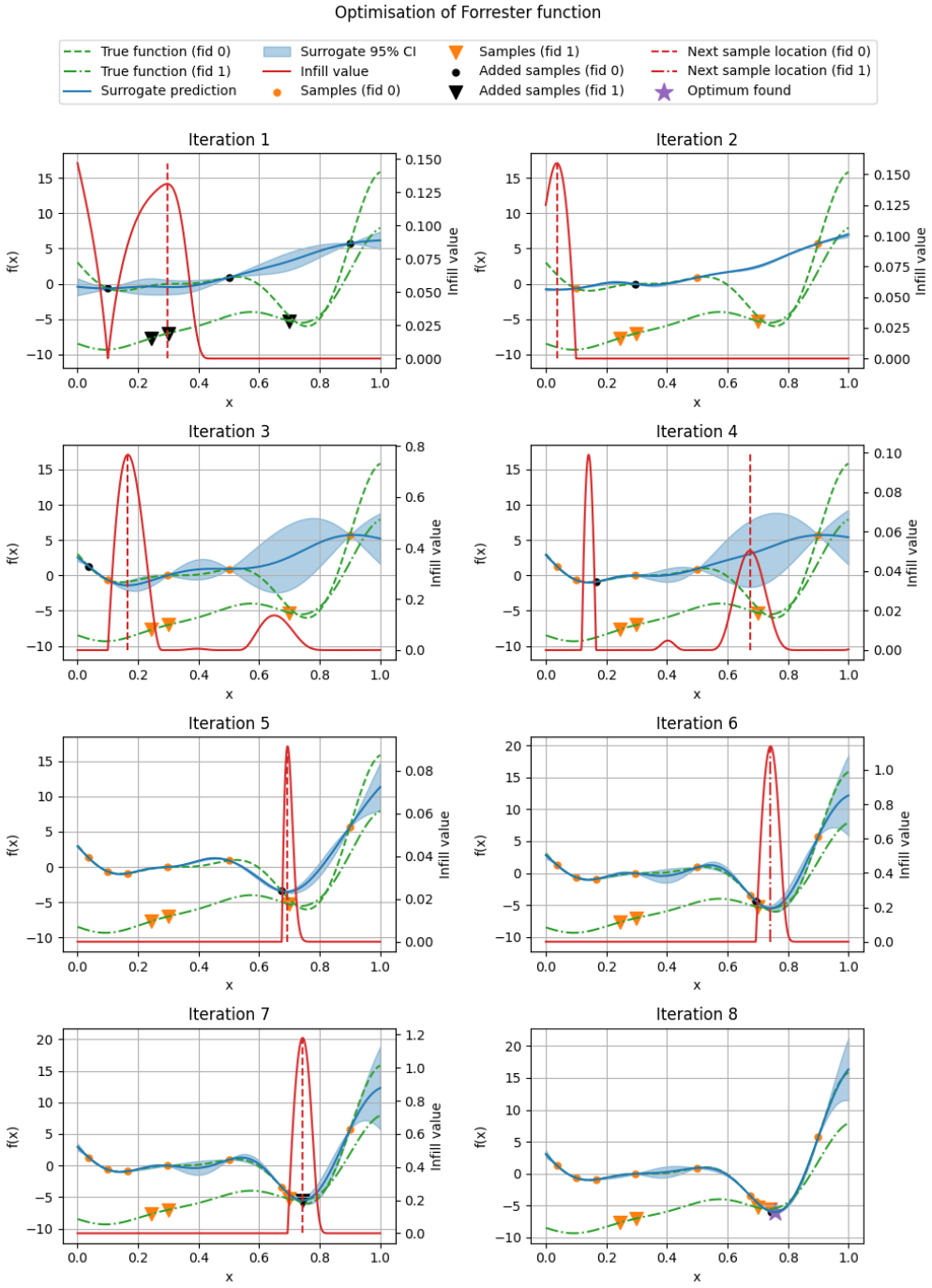


Figure 8.2: The optimisation process for the Forrester function with the reference LF function using the MF SP Two-Step EI infill strategy.

9

Discussion

This chapter discusses the results as presented in chapter 7 and chapter 8. First, the findings for numerical problems are discussed in section 9.1, followed by the discussion of the results for the varying Low Fidelity (LF) functions in section 9.2. Next, the limitations of the research and implemented method are discussed in section 9.3. Lastly, this chapter highlights the practical application and recommendations for future work in section 9.4 and section 9.5, respectively.

9.1. Numerical Test Problems

The results for the numerical test functions as introduced in chapter 7 show that the asynchronous outperforms the non-asynchronous infill strategies. This is consistent with the reference asynchronous infill strategy from Przysowa et al. [38], which demonstrated improved performance. This is the result of the optimisers adding more information over time by sampling multiple new points concurrently. This increase in information provided to the surrogate at time t allows the surrogate to be more accurate, leading to earlier convergence.
This resulted in a reduction of 72.7% in Expected Runtime (ERT) compared to the Single-Point (SP) Single-Fidelity (SF) Expected Improvement (EI) baseline. This is above the threshold of 40% above which a significant difference in performance could be detected.

The Multi-Fidelity (MF) infill strategies were expected to outperform the SF infill strategies; however, the results showed the opposite. The SF infill strategies consistently outperformed the MF strategies. This was caused by the addition of the LF samples, while the number of High Fidelity (HF) infill samples selected per run did not change significantly. This led to the elapsed time increasing. To investigate the role of the LF function in this behaviour, several LF Forrester functions were defined on which the performance was compared, the results of which are discussed in section 9.2.

Next, a comparison was made between the EI and Generalised Expected Improvement (GEI) based infill strategies. This comparison showed that the EI based infill strategies slightly outperformed the GEI based strategies. This was mostly due to the lower success rate of the GEI based infill strategies. Note that the GEI based infill strategies have not been run for the required number of random seeds, so no significant conclusion can be made from the results.

9.2. Varying Low Fidelity Functions

As discussed in chapter 7, chapter 8 and section 9.1, the MF infill strategies underperformed compared to the hypothesis as discussed in section 4.2. The benchmarks on the numerical test functions showed that the elapsed time increased as LF samples were added by the MF infill strategy, whilst the number of HF required did not change. The test performed in chapter 8, showed the same behaviour for the successful optimisation runs, leading to longer optimisation run times. After visualising the optimisation process, it was apparent that the MF infill strategy included more sample points around the optimum of the function. This was caused by the LF function deemed to be sufficiently accurate in the area around the optimum by the Two-Step infill strategy. An LF sample would thus be selected near the optimum.

This resulted in the variance of the Hierarchical Kriging (HK) surrogate reducing near the new sample point, but since it is an LF sample point not reducing to 0. The used EI SF infill strategy in the Two-Step strategy would reselect the sample location, based on the reduced variance and the surrogate prediction at that point. This reselected sample point would be sampled using the HF function, as otherwise an identical sample would be added. This thus results in an additional sample point added at the same location, increasing the computational cost without adding any meaningful information.

9.3. Limitations

The results presented previously are constrained by several limitations of the research. The first limitation is the selection of the hyperparameters for the optimisation algorithms. The tests performed were all run using a consistent set of default hyperparameters, since, due to time constraints, it was not possible to optimise these on a function-infill strategy basis. The choice of these parameters could affect an algorithm's performance. For example, changing the exponent for the Gaussian correlation function in the HK surrogate from the used 2.0 to 1.9 yielded different ERTs for each numerical test function. The results of this change for SP HK Two-Step are tabulated in Table 9.1. In this table, it is visible that the $p = 2.0$ achieves a better ERT for most problems. However, it does not outperform $p = 1.9$ for every function. These results show that the optimal settings for these parameters are highly problem-dependent. This is in line with the no free lunch theorem, which states that no single optimisation algorithm is universally better across all problems [65]. Optimising this full set of hyperparameters and settings for each optimisation algorithm and problem could thus improve performance across subsets of the function-optimiser combinations.

Table 9.1: ERT (HH:MM:SS) for SP HK Two-Step for changing exponent of the Gaussian correlation function with 30 random seeds.

Algorithm	SP HK Two-Step $p = 2.0$	SP HK Two-Step $p = 1.9$
Forrester	00:14:40	00:32:27
Bohachevsky	03:35:05	02:50:14
Booth	00:38:09	04:12:03
Branin	00:56:38	00:56:48
Currin	01:18:01	02:21:11
Himmelblau	03:24:24	07:01:18
Six Hump Camelback	05:34:08	09:50:38
Park91A	01:08:23	02:03:23
Park91B	04:09:38	04:51:22
Hartmann6	18:26:04	16:15:20

Another set of parameters that can be tuned to achieve different performance are the stopping criteria. The most commonly hit stopping criterion was the value returned by the infill strategy. The threshold was set to 1×10^{-6} for all combinations of optimiser and function. However, Schonlau [18] advised to use $E\left[I^g\right]^{\frac{1}{g}} <$ threshold, where currently it was assessed the GEI $<$ threshold, or $E\left[I^2\right] <$ threshold for the value of g used ($g = 2$).
Changing this stopping criterion for the combinations using the SF GEI infill strategy will result in a stricter criterion, possibly leading the optimiser to continue for longer, adding more sample points. This could result in an increase in successful runs, at the cost of longer elapsed time. Since the GEI-based combinations already display a relatively high success rate, this change is not expected to result in a better ranking based on ERT compared to the EI-based combinations.

A limitation of the implementation is the use of Python 3.10 without preemptive threading or the Global Interpreter Lock (GIL). Implementing the framework in a newer version of Python without the GIL and supporting preemptive threading, or using a different language altogether, could improve the performance of the Multi-Point (MP) strategies. The change to the rank 0, which currently solely orchestrates the optimisation process, could aid the optimisation process by evaluating a sample point itself. This will likely improve the performance of these MP infill strategies over the SP strategies, but not alter the ranking within this group, given the same number of Central Processing Units (CPUs). This difference

should be more pronounced for runs with fewer ranks available, since there is a relatively larger increase in computational power available as adding 1 rank to a 2-rank optimisation is a relatively larger increase over an optimisation utilising 16 ranks. In addition, this difference is exacerbated by the coordinator rank in a 16-rank optimisation, which requires selecting new sample points more often, resulting in less computational time available for sample point evaluation on rank 0.

9.4. Practical Application

The purpose of this MSc thesis was to examine whether an optimisation problem can be solved more efficiently in terms of wall-clock time by utilising a MF MP infill strategy for a Surrogate-Based Optimisation (SBO). This resulted in a proposed infill strategy using the Two-Step infill strategy for an asynchronous optimisation algorithm. This implementation can be used as-is for any unconstrained single-value optimisation problem.

This, however, does not yield a performance gain for typical engineering optimisation problems, since these problems are commonly constrained optimisation problems. By utilising the proposed optimiser on these kinds of problems, the expensive constraints will be evaluated directly when optimising the SF infill strategy and while finding the optimum on the surrogate. As a result, this will turn the SBO process into an expensive optimisation with many evaluations of (one of) the expensive functions, negating the benefit of an SBO algorithm.

The proposed method can easily be adapted to handle expensive constraints by utilising an SF infill strategy that selects a sample point without evaluating the expensive constraint. An example of such an infill strategy is Constrained Expected Improvement (CEI). This infill strategy uses the Probability of Feasibility (PoF), in addition to EI, to select the new sample point, with the PoF evaluated on a surrogate constructed for each expensive constraint. The resulting new infill point is then sampled, with the corresponding constraint values added to the constraint surrogate(s).

The current implementation of the optimisation framework would require a change. First, a constraint-handling infill strategy must be included, such as CEI. In addition, in the currently implemented optimisation processes, only one surrogate is constructed and updated; this should be extended to include the surrogates required for all expensive constraints. Lastly, the configuration class must be extended to allow the user to pass expensive constraints to the problem definition.

9.5. Future Work

The proposed MF MP infill strategy showed an increase in performance over the baseline, but disappointing results compared to the SF MP infill strategy. Also for the SP the addition of the MF aspect reduced performance, suggesting that the selected MF infill strategy hampers performance on the set of benchmark functions. The combination of the MP and MF aspects is theoretically a promising direction for further improving the performance of SBO. The first recommendation for future work is thus to adapt the Two-Step infill strategy or combine the asynchronous MP strategy with a different MF infill strategy, which could aid in increasing the combined performance.

One possible avenue for adjusting the Two-Step infill strategy is by adjusting the Jensen-Shannon distance (JSd) threshold. This threshold determines whether a fidelity is deemed sufficiently accurate at a sample location. The currently identified problem with the MF infill strategy is that additional samples are added around the optimum. These points are generally sampled near the end of the optimisation run, when the area of the Forrester function with the optimum has already been identified as promising. Tweaking the JSd throughout the optimisation might resolve this problem. Starting the optimisation with a lenient JSd, will result in more LF samples to be added at the start of the optimisation, possibly giving the optimiser early on in the optimisation a better overview of the general trend of the design space. When the optimisation run progresses, this threshold can be made stricter, resulting in more HF sample points to be added. This will give a more detailed overview of the at that point identified promising area(s) in the design space. This would require the threshold to be set based on a variable, which can be linked to the state of the optimisation run, such as the infill value at the last selected infill point. This value will reduce when the optimisation run progresses.

Changing the Two-Step infill strategy for another MF infill strategy is another approach that could result in increased performance. Other infill strategies can be readily incorporated into the framework, enabling

benchmarking among these methods. In section 4.5, the choice was made to favour a MF infill strategy, which does not take into account the cost-ratio between fidelity levels, as this could make the selection of infill points challenging in a MP setting. However, utilising the asynchronous MP strategy, this concern is dropped as the fidelity level and fidelity levels of the other sampled points do not influence the iteration time as they would for a synchronous MP strategy.
This thus opens the possibility to investigate the performance of other MF infill strategies in combination with an asynchronous MP optimiser. This would, however, require the (estimation of) cost-ratio(s), the choice of which could influence the selection of sample fidelity. These ratios can be easily updated throughout the optimisation process, since the duration of a sample point evaluation is logged.

The last recommendation focuses on improving the framework's practical applications. The current implementation, as discussed in section 9.3 and section 9.4, has several limitations that limit its practical applications. The most limiting limitation for the practical applicability is the lack of constraint handling; the last recommendation is thus to extend the framework to be able to handle constrained optimisation problems.

10

Conclusion

This MSc thesis aimed to investigate the performance of a combined Multi-Point (MP) Multi-Fidelity (MF) infill strategy in Surrogate-Based Optimisation (SBO), formalised into the following research question:

> How does a parallel multi-point multi-fidelity infill strategy affect the performance of a surrogate-based optimisation run?

To this end, an existing MP Single-Fidelity (SF) and a Single-Point (SP) MF infill strategy were combined into the proposed MF MP infill strategy. The MP aspect of this proposed strategy is adapted from the strategy proposed by Przysowa et al. [38]. The proposed strategy selects one new sample location at a time, using an SP infill strategy. This new sample point is sent to an available rank for evaluation. While this point is being evaluated, another new sample point is selected (possibly using a heuristic like Kriging Believer (KB) or Constant Liar (CL) if the surrogate has not been updated) and sent to an available rank; this is repeated until no free rank is available. After the new sample point is evaluated, it is added to the surrogate.
The used SP MF infill strategy in this process is the Two-Step strategy by Garbo et al. [31]. This strategy works by first selecting the infill location on the High Fidelity (HF) surrogate of a Hierarchical Kriging (HK) model, using an SP SF infill strategy. In the tests performed, both an Expected Improvement (EI) and a Generalised Expected Improvement (GEI) infill strategy were used for this. Next, the sample fidelity is selected by first generating a list of fidelities that are deemed sufficiently accurate at the selected infill location. A fidelity is considered sufficiently accurate if the Jensen-Shannon distance (JSd) is below a set threshold. This JSd measures the similarity between two distributions. The two distributions tested are generated from the prediction and variance returned by the highest-fidelity HK model, and the prediction and variance of the HK at the fidelity to be tested. From this list of sufficiently accurate fidelities, the computationally cheapest model is selected, finalising the selection of the new sample point.
The formulation of this proposed MF MP infill strategy answered the sub-questions. The first sub-question;

> How to decide the sample location and fidelity level for a new sample point?

was answered by the selection of the Two-Step infill strategy.

The second and third sub-questions are answered by selecting the asynchronous MP strategy.

> How to decide on the number of new high-fidelity and low(er)-fidelity samples per iteration?
> How can the available computational resources be maximally used?

The number of new HF and Low Fidelity (LF) samples per iteration is not relevant for the asynchronous approach, as asynchronous evaluation of a sample point minimises idle time on the worker ranks. This is achieved by selecting a new sample location immediately once a sample point is returned; a rank evaluating an LF sample point thus does not have to wait for a rank evaluating an HF sample point to finish.

The proposed method was tested using eleven two-fidelity benchmark problems. Of these eleven, four were used only as an indication, as the number of runs required for significant results could not be reached within the computational budget. For the seven remaining problems the MP SF infill strategy with 16 ranks achieved a 72.7% reduction in Expected Runtime (ERT) compared to the SF SP EI baseline. This difference was above the threshold of 40% above which a significant difference could be detected. Contrary to the hypothesis, the addition of the MF aspect reduced the performance for both the MP and SP implementations compared to their SF counterparts. This reduction in performance was caused by the sampling of multiple points around the optimum, starting with an LF sample point followed in the next iteration by an HF sample point at the same location. This was the result of the variance of the HK not being driven to 0 at the LF sample location, where the residual variance would prompt the infill criterion to include a sample point in the same location. The addition of this LF point thus did not add any meaningful information to the optimisation process, while taking up computational resources. To investigate whether the formulation of the LF function has an effect on the performance, the proposed infill strategy was tested on the Forrester function with a set of different LF functions. These tests confirmed the behaviour as discussed above was structural and not an artefact of the formulation of the used LF functions.

The four functions from the test suite not used in the statistical testing were run for a set of random seeds, the results of which displayed the same behaviour: the MP addition improved performance, whereas adding MF decreased it.

The tests described above all used EI as the SF SP infill strategy in the Two-Step infill strategy. An indicative test was performed using GEI as the SP SF infill strategy. This showed the performance of EI was slightly superior to GEI, owing to the higher success rate of EI.

In answer to the main question, the proposed asynchronous parallel MP MF infill strategy improves SBO performance, through the asynchronous MP component, which delivers a statistically significant reduction in ERT over the baseline. The selected MF, however, degrades the performance. The hypothesis is therefore partially confirmed: the expected performance increase is realised by the asynchronous MP aspect, but hampered by the inclusion of MF.

The proposed strategy can be directly applied to unconstrained single-objective optimisation problems. Implementing an SF SP infill strategy capable of handling constraints, like Constrained Expected Improvement (CEI), ensures the proposed method is applicable to constrained problems. To benefit from the combined MP MF aspect, the MF has to be adapted. A promising direction is to alter the JSd threshold during the optimisation process. This could begin with a lenient threshold so as to select several LF sample points to obtain a fast and computationally cheap overview of the general trend in the design space. As the optimisation process progresses, the threshold would be tightened, so that the promising areas of the design space are explored using HF samples, which give a better view of the areas of interest. The possible performance increase from MP MF could be realised by exchanging the MF infill strategy used.

This thesis thus showed that an asynchronous MP infill strategy yields significant reductions in wall-clock time, while the Two-Step MF infill strategy partially negated this performance increase.

References

[1] P. Saves et al., “SMT 2.0: A Surrogate Modeling Toolbox with a focus on hierarchical and mixed variables Gaussian processes,” *Advances in Engineering Software*, vol. 188, p. 103 571, Feb. 2024, ISSN: 0965-9978. DOI: 10.1016/J.ADVENGSOFT.2023.103571. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096599782300162X.

[2] B. Do and R. Zhang, “Multi-fidelity Bayesian Optimization in Engineering Design,” Nov. 2024. [Online]. Available: https://arxiv.org/abs/2311.13050.

[3] A. I. J. Forrester, A. J. Keane, J. M. Parr, and C. M. E. Holden, “Review of Efficient Surrogate Infill Sampling Criteria with Constraint Handling,” Tech. Rep., 2010. [Online]. Available: https://www.researchgate.net/publication/265227879.

[4] J. X. Leng, Z. G. Wang, W. Huang, Y. Shen, and K. An, “Multidisciplinary Design Optimization Processes for Efficiency Improvement of Aircraft: State-of-the-Art Review,” *International Journal of Aeronautical and Space Sciences*, pp. 1–23, Aug. 2024, ISSN: 20932480. DOI: 10.1007/s42405-024-00811-8. [Online]. Available: https://link.springer.com/article/10.1007/s42405-024-00811-8.

[5] P. Jiang, Q. Zhou, and X. Shao, *Surrogate Model-Based Engineering Design and Optimization*, 1st ed., S.-B. Choi, H. Duan, Y. Fu, C. Guardiola, J.-Q. Sun, and Y. W. Kwon, Eds. Signapore: Springer Singapore, Nov. 2019, ISBN: 978-981-15-0731-1. DOI: https://doi.org/10.1007/978-981-15-0731-1. [Online]. Available: https://link.springer.com/book/10.1007/978-981-15-0731-1.

[6] J. R. Martins and A. Ning, *Engineering Design Optimization*. Cambridge, UK: Cambridge University Press, Jan. 2022, ISBN: 9781108833417. DOI: 10.1017/9781108980647. [Online]. Available: https://mdobook.github.io/.

[7] R. T. Haftka, D. Villanueva, and A. Chaudhuri, “Parallel surrogate-assisted global optimization with expensive functions – a survey,” *Structural and Multidisciplinary Optimization*, vol. 54, no. 1, pp. 3–13, Jul. 2016, ISSN: 16151488. DOI: 10.1007/S00158-016-1432-3/FIGURES/2. [Online]. Available: https://link.springer.com/article/10.1007/s00158-016-1432-3.

[8] M. Schouler, A. Belme, and P. Cinnella, *Bayesian and non-Bayesian multi-fidelity surrogate models for multi-objective aerodynamic optimization under extreme cost imbalance*, May 2025. [Online]. Available: https://arxiv.org/abs/2505.17279.

[9] J. Sobieszczanski-Sobieski, A. Morris, M. J. van Tooren, G. La Rocca, and W. Yao, *Multidisciplinary Design Optimization Supported by Knowledge Based Engineering*. Chichester, UK: John Wiley & Sons, 2015, pp. 1–378, ISBN: 9781118492123. DOI: 10.1002/9781118897072.

[10] A. I. J. Forrester, A. Sóbester, and A. J. Keane, *Engineering Design via Surrogate Modelling A Practical Guide*. Chichester, UK: John Wiley & Sons, Ltd., Jul. 2008, ISBN: 978-0-470-06068-1. DOI: 10.1002/9780470770801. [Online]. Available: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470770801.

[11] M. D. Buhmann, “Radial basis functions,” *Acta Numerica*, vol. 9, pp. 1–38, Mar. 2000, ISSN: 1474-0508. DOI: 10.1017/S0962492900000015. [Online]. Available: https://www.cambridge.org/core/journals/acta-numerica/article/abs/radial-basis-functions/3FD3A8BBC9B020FA349305142D0EB367.

[12] S. De Marchi and E. Perracchione, *Lectures on Radial Basis Functions*, Padua, It, Jan. 2018. [Online]. Available: https://www.math.unipd.it/~demarchi/RBF/LectureNotes_new.pdf.

[13] A. Iske, “Scattered data approximation by positive definite kernel functions,” *Rendiconti del Seminario Matematico*, vol. 69, no. 3, pp. 217–246, 2011. [Online]. Available: https://seminariomatematico.polito.it/rendiconti/69-3/217.pdf.

[14] B. Peherstorfer, K. Willcox, and M. Gunzburger, “Survey of Multifidelity Methods in Uncertainty Propagation, Inference, and Optimization,” *https://doi.org/10.1137/16M1082469*, vol. 60, no. 3, pp. 550–591, Aug. 2018, ISSN: 00361445. DOI: 10.1137/16M1082469. [Online]. Available: https://epubs.siam.org/doi/10.1137/16M1082469.

[15] J. Bussemaker, N. Bartoli, T. Lefebvre, P. D. Ciampa, and B. Nagel, “Effectiveness of Surrogate-Based Optimization Algorithms for System Architecture Optimization,” in *AIAA Aviation and Aeronautics Forum and Exposition, AIAA AVIATION Forum 2021*, American Institute of Aeronautics and Astronautics Inc, AIAA, Aug. 2021, ISBN: 9781624106101. DOI: 10.2514/6.2021-3095.

[16] G. I. Hawe and J. K. Sykulski, “An Enhanced Probability of Improvement Utility Function for Locating Pareto Optimal Solutions,” in *16th Conference on the Computation of Electromagnetic Fields COMPUMAG*, Aachen, Germany, 2007, pp. 965–966. [Online]. Available: https://eprints.soton.ac.uk/264368/.

[17] D. R. Jones, “A Taxonomy of Global Optimization Methods Based on Response Surfaces,” *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, Dec. 2001. DOI: 10.1023/A:1012771025575. [Online]. Available: https://doi.org/10.1023/A:1012771025575.

[18] M. Schonlau, “Computer Experiments and Global Optimization,” Ph.D. dissertation, University of Waterloo, Waterloo, Ont, Can, 1997, ISBN: 0612222349.

[19] M. Giselle Fernández-Godino, “Review of multi-fidelity models,” *Advances in Computational Science and Engineering*, vol. 1, no. 4, pp. 351–400, 2023, ISSN: 2837-1739. DOI: 10.3934/acse.2023015. [Online]. Available: https://www.aimsciences.org//article/doi/10.3934/acse.2023015.

[20] M. Kennedy and A. O’Hagan, “Predicting the Output from a Complex Computer Code When Fast Approximations Are Available,” *Biometrika*, vol. 87, no. 1, pp. 1–13, Mar. 2000, ISSN: 00063444, 14643510. [Online]. Available: https://www.jstor.org/stable/2673557?seq=1.

[21] S. v. Rijn and S. Schmitt, “MF2: A Collection of Multi-Fidelity Benchmark Functions in Python,” *Journal of Open Source Software*, vol. 5, no. 52, p. 2049, Aug. 2020, ISSN: 2475-9066. DOI: 10.21105/JOSS.02049. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.02049.

[22] Z. H. Han and S. Görtz, “Hierarchical kriging model for variable-fidelity surrogate modeling,” *AIAA Journal*, vol. 50, no. 9, pp. 1885–1896, Sep. 2012, ISSN: 00011452. DOI: 10.2514/1.J051354. [Online]. Available: https://doi.org/10.2514/1.J051354.

[23] H. Dong, S. Sun, B. Song, and P. Wang, “Multi-surrogate-based global optimization using a score-based infill criterion,” *Structural and Multidisciplinary Optimization*, vol. 59, no. 2, pp. 485–506, Feb. 2019, ISSN: 16151488. DOI: 10.1007/s00158-018-2079-z. [Online]. Available: https://doi.org/10.1007/s00158-018-2079-z.

[24] Y. Zhang, Z. h. Han, and W. p. Song, “Multi-fidelity expected improvement based on multi-level hierarchical kriging model for efficient aerodynamic design optimization,” *Engineering Optimization*, vol. 56, no. 12, pp. 2408–2430, Dec. 2024, ISSN: 10290273. DOI: 10.1080/0305215X.2024.2310182. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/0305215X.2024.2310182.

[25] M. Sacher, O. Le Maitre, R. Duvigneau, F. Hauville, M. Durand, and C. Lothode, “A Non-Nested Infilling Strategy for Multi-Fidelity based Efficient Global Optimization,” *International Journal for Uncertainty Quantification*, vol. 11, no. 1, pp. 1–30, Jan. 2021, ISSN: 2152-5080. DOI: 10.1615/Int.J.UncertaintyQuantification.2020032982. [Online]. Available: https://inria.hal.science/hal-02901774v1.

[26] L. Shu, P. Jiang, and Y. Wang, “A multi-fidelity Bayesian optimization approach based on the expected further improvement,” *Structural and Multidisciplinary Optimization*, vol. 63, no. 4, pp. 1709–1719, Apr. 2021, ISSN: 16151488. DOI: 10.1007/S00158-020-02772-4. [Online]. Available: https://link.springer.com/article/10.1007/s00158-020-02772-4.

[27] Y. Zhang, Z. H. Han, and K. S. Zhang, “Variable-fidelity expected improvement method for efficient global optimization of expensive functions,” *Structural and Multidisciplinary Optimization*, vol. 58, no. 4, pp. 1431–1451, Oct. 2018, ISSN: 16151488. DOI: 10.1007/S00158-018-1971-X. [Online]. Available: https://link.springer.com/article/10.1007/s00158-018-1971-x.

[28] P. Hao, S. Feng, Y. Li, B. Wang, and H. Chen, “Adaptive infill sampling criterion for multi-fidelity gradient-enhanced kriging model,” *Structural and Multidisciplinary Optimization*, vol. 62, no. 1, pp. 353–373, Jul. 2020, ISSN: 16151488. DOI: 10.1007/S00158-020-02493-8. [Online]. Available: https://link.springer.com/article/10.1007/s00158-020-02493-8.

[29] H. Yang, S. H. Hong, and Y. Wang, “A sequential multi-fidelity surrogate-based optimization methodology based on expected improvement reduction,” *Structural and Multidisciplinary Optimization*, vol. 65, no. 5, pp. 1–17, May 2022, ISSN: 16151488. DOI: 10.1007/S00158-022-03240-X. [Online]. Available: https://link.springer.com/article/10.1007/s00158-022-03240-x.

[30] Z. Guo, Q. Wang, L. Song, and J. Li, “Parallel multi-fidelity expected improvement method for efficient global optimization,” *Structural and Multidisciplinary Optimization*, vol. 64, no. 3, pp. 1457–1468, Sep. 2021, ISSN: 16151488. DOI: 10.1007/S00158-021-02931-1. [Online]. Available: https://link.springer.com/article/10.1007/s00158-021-02931-1.

[31] A. Garbo, J. Parekh, T. Rischmann, and P. Bekemeyer, “Multi-Fidelity Adaptive Sampling for Surrogate-Based Optimization and Uncertainty Quantification,” *Aerospace*, vol. 11, no. 6, p. 448, 2024, ISSN: 2226-4310. DOI: 10.3390/aerospace11060448. [Online]. Available: https://www.mdpi.com/2226-4310/11/6/448.

[32] R. Pellegrini, J. Wackers, R. Broglia, A. Serani, M. Visonneau, and M. Diez, “A multi-fidelity active learning method for global design optimization problems with noisy evaluations,” *Engineering with Computers*, vol. 39, no. 5, pp. 3183–3206, Oct. 2023, ISSN: 14355663. DOI: 10.1007/S00366-022-01728-0. [Online]. Available: https://link.springer.com/article/10.1007/s00366-022-01728-0.

[33] J. Wackers, M. Visonneau, A. Serani, R. Pellegrini, R. Broglia, and M. Diez, “Multi-Fidelity Machine Learning from Adaptive-and Multi-Grid RANS Simulations,” in *33rd Symposium on Naval Hydrodynamics*, Ousaka, Japan, Oct. 2020, pp. 18–23. [Online]. Available: https://hal.science/hal-03124652v1.

[34] J. Lin, “Divergence Measures Based on the Shannon Entropy,” *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 145–151, 1991, ISSN: 15579654. DOI: 10.1109/18.61115. [Online]. Available: https://ieeexplore.ieee.org/document/61115.

[35] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, Ltd, Apr. 2005, p. 748, ISBN: 9780471241959. DOI: 10.1002/047174882X. [Online]. Available: https://onlinelibrary.wiley.com/doi/book/10.1002/047174882X.

[36] D. M. Endres and J. E. Schindelin, “A new metric for probability distributions,” *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1858–1860, 2003, ISSN: 00189448. DOI: 10.1109/TIT.2003.813506. [Online]. Available: https://ieeexplore.ieee.org/document/1207388.

[37] A. Serani et al., “Adaptive multi-fidelity sampling for CFD-based optimisation via radial basis function metamodels,” *International Journal of Computational Fluid Dynamics*, vol. 33, no. 6-7, pp. 237–255, Aug. 2019, ISSN: 10290257. DOI: 10.1080/10618562.2019.1683164. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/10618562.2019.1683164.

[38] K. Przysowa, Ł. Łaniewski-Wołłk, and J. Rokicki, “Shape optimisation method based on the surrogate models in the parallel asynchronous environment,” *Applied Soft Computing*, vol. 71, pp. 1189–1203, Oct. 2018, ISSN: 1568-4946. DOI: 10.1016/J.ASOC.2018.04.028. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494618302199.

[39] W. Wang et al., “Surrogate-based optimization with adaptive parallel infill strategy enhanced by inaccurate multi-objective search,” *Engineering Optimization*, vol. 54, no. 8, pp. 1356–1373, Aug. 2022, ISSN: 10290273. DOI: 10.1080/0305215X.2021.1928109. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/0305215X.2021.1928109.

[40] A. Chaudhuri and R. T. Haftka, “Efficient global optimization with adaptive target setting,” *AIAA Journal*, vol. 52, no. 7, pp. 1573–1577, Jun. 2014, ISSN: 00011452. DOI: 10.2514/1.J052930. [Online]. Available: https://doi.org/10.2514/1.J052930.

[41] C. Chevalier and D. Ginsbourger, “Fast Computation of the Multi-points Expected Improvement with Applications in Batch Selection,” Oct. 2012. [Online]. Available: https://hal.science/hal-00732512v2.

[42] Y. Wang, Z. H. Han, Y. Zhang, and W. P. Song, “Efficient global optimization using multiple infill sampling criteria and surrogate models,” *AIAA Aerospace Sciences Meeting, 2018*, 2018. DOI: 10.2514/6.2018-0555. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2018-0555.

[43] P. Beaucaire, C. Beauthier, and C. Sainvitu, “Multi-point infill sampling strategies exploiting multiple surrogate models,” in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 1559–1567, ISBN: 9781450367486. DOI: 10.1145/3319619.3328527. [Online]. Available: https://doi.org/10.1145/3319619.3328527.

[44] D. Ginsbourger, R. L. Riche, L. Carraro, R. Le Riche, Y. Tenne, and C.-K. Goh, “Kriging Is Well-Suited to Parallelize Optimization,” in *Computational Intelligence in Expensive Optimization Problems*, 2, Berlin, Heidelberg: Springer, Berlin, Heidelberg, 2010, pp. 131–162, ISBN: 978-3-642-10701-6. DOI: 10.1007/978-3-642-10701-6_6. [Online]. Available: https://doi.org/10.1007/978-3-642-10701-6_6.

[45] J. Liu, W. P. Song, Z. H. Han, and Y. Zhang, “Efficient aerodynamic shape optimization of transonic wings using a parallel infilling strategy and surrogate models,” *Structural and Multidisciplinary Optimization*, vol. 55, no. 3, pp. 925–943, Mar. 2017, ISSN: 16151488. DOI: 10.1007/S00158-016-1546-7. [Online]. Available: https://link.springer.com/article/10.1007/s00158-016-1546-7.

[46] J. Liu, Z.-H. Han, and W. Song, “Comparison of infill sampling criteria in Kriging-based aerodynamic optimization,” *28th Congress of the International Council of the Aeronautical Sciences 2012, ICAS 2012*, vol. 2, pp. 1625–1634, Jan. 2012. [Online]. Available: https://www.researchgate.net/publication/287020982_Comparison_of_infill_sampling_criteria_in_Kriging-based_aerodynamic_optimization.

[47] H. Liu, Y. S. Ong, and J. Cai, “A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design,” *Structural and Multidisciplinary Optimization 2017 57:1*, vol. 57, no. 1, pp. 393–416, Jun. 2017, ISSN: 1615-1488. DOI: 10.1007/S00158-017-1739-8. [Online]. Available: https://link.springer.com/article/10.1007/s00158-017-1739-8.

[48] C. Sainvitu, V. Iliopoulou, and I. Lepot, “Global Optimization with Expensive Functions - Sample Turbomachinery Design Application,” *Recent Advances in Optimization and its Applications in Engineering*, pp. 499–509, 2010. DOI: 10.1007/978-3-642-12598-0_44. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-12598-0_44.

[49] K. Deb, “An efficient constraint handling method for genetic algorithms,” *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 311–338, Jun. 2000, ISSN: 0045-7825. DOI: 10.1016/S0045-7825(99)00389-8. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0045782599003898.

[50] F. A. C. Viana, R. T. Haftka, and L. T. Watson, “Why Not Run the Efficient Global Optimization Algorithm with Multiple Surrogates?” In *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Orlando, Florida, USA: American Institute of Aeronautics and Astronautics, Inc., Apr. 2010. DOI: 10.2514/6.2010-3090.

[51] J. Mueller, “MATSuMoTo: The MATLAB Surrogate Model Toolbox For Computationally Expensive Black-Box Global Optimization Problems,” Apr. 2014. [Online]. Available: http://arxiv.org/abs/1404.4261.

[52] R. G. Regis and C. A. Shoemaker, “A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions,” *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 497–509, Jul. 2007, ISSN: 15265528. DOI: 10.1287/ijoc.1060.0182. [Online]. Available: https://doi.org/10.1287/ijoc.1060.0182.

[53] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard, *DACE - A Matlab Kriging Toolbox, Version 2.0*. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2002, vol. IMM-TR-2002-12. [Online]. Available: https://orbit.dtu.dk/en/publications/dace-a-matlab-kriging-toolbox-version-20/.

[54] P. Virtanen et al., “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[55] R. Storn and K. Price, “Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997, ISSN: 09255001. DOI: 10.1023/A:1008202821328. [Online]. Available: https://link.springer.com/article/10.1023/A:1008202821328.

[56] *Benchmarking*, Hamburg, Jun. 2020. [Online]. Available: https://www.hhcc.uni-hamburg.de/learning-hpc/getting-started-with-hpc-clusters-b/getting-started-with-hpc-clusters-b-y-benchmarking-b.html.

[57] D. Bindel, *Performance analysis basics*, Aug. 2015. [Online]. Available: https://cornell-cs5220-f15.github.io/2015/08/10/performance.html.

[58] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, “COCO: a platform for comparing continuous optimizers in a black-box setting,” *Optimization Methods and Software*, vol. 36, no. 1, pp. 114–144, 2021, ISSN: 10294937. DOI: 10.1080/10556788.2020.1808977.

[59] K. Price, “Differential evolution vs. the functions of the 2nd ICEO,” in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC ’97)*, 1997, pp. 153–157. DOI: 10.1109/ICEC.1997.592287.

[60] J. Demšar, “Statistical Comparisons of Classifiers over Multiple Data Sets,” *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, Dec. 2006, ISSN: 1532-4435. DOI: 10.5555/1248547.1248548. [Online]. Available: https://dl.acm.org/doi/10.5555/1248547.1248548.

[61] J. J. Moré and S. M. Wild, “Benchmarking Derivative-Free Optimization Algorithms,” *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 172–191, Mar. 2009, ISSN: 10526234. DOI: 10.1137/080724083. [Online]. Available: https://epubs.siam.org/doi/10.1137/080724083.

[62] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, 1st ed. New York: Chapman and Hall/CRC, May 1994, ISBN: 9780429246593. DOI: 10.1201/9780429246593. [Online]. Available: https://www.taylorfrancis.com/books/mono/10.1201/9780429246593/introduction-bootstrap-bradley-efron-tibshirani.

[63] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. New York: Routledge, Apr. 2013, ISBN: 9780203771587. DOI: 10.4324/9780203771587. [Online]. Available: https://www.taylorfrancis.com/books/mono/10.4324/9780203771587/statistical-power-analysis-behavioral-sciences-jacob-cohen.

[64] M. Allen, D. Poggiali, K. Whitaker, T. R. Marshall, and R. A. Kievit, “Raincloud plots: a multi-platform tool for robust data visualization,” *Wellcome Open Research*, vol. 4, p. 63, Apr. 2019, ISSN: 2398-502X. DOI: 10.12688/WELLCOMEOPENRES.15191.2.

[65] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997, ISSN: 1089778X. DOI: 10.1109/4235.585893. [Online]. Available: https://ieeexplore.ieee.org/document/585893.

[66] C. R. Harris et al., “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10.1038/s41586-020-2649-2. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[67] D. H. P. C. C. (DHPC), *DelftBlue Supercomputer (Phase 2)*, https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024.

[68] G. Shainer, P. Lui, T. Liu, T. Wilde, and J. Layton, “The Impact of Inter-node Latency versus Intra-node Latency on HPC Applications,” in *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Systems*, Dec. 2011. [Online]. Available: http://www.hpcadvisorycouncil.com.

[69] G. Hu, Y. Zhang, and W. Chen, “Exploring the performance of singularity for high performance computing scenarios,” *Proceedings - 21st IEEE International Conference on High Performance Computing and Communications, 17th IEEE International Conference on Smart City and 5th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2019*, pp. 2587–2593, Aug. 2019. DOI: 10.1109/HPCC/SMARTCITY/DSS.2019.00362. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8855563.

[70] E. Le and D. Paz, “Performance analysis of applications using singularity container on SDSC Comet,” in *ACM International Conference Proceeding Series*, vol. Part F128771, Association for Computing Machinery, Jul. 2017, ISBN: 9781450352727. DOI: 10.1145/3093338.3106737.

[71] A. Torrez, T. Randles, and R. Priedhorsky, “HPC Container Runtimes have Minimal or No Performance Impact,” *Proceedings of CANOPIE-HPC 2019: 1st International Workshop on Containers and New Orchestration Paradigms for Isolated Environments in HPC - Held in conjunction with SC 2019: The International Conference for High Performance Computing, Networking, Storag…*, pp. 37–42, Nov. 2019. DOI: 10.1109/CANOPIE-HPC49598.2019.00010. [Online]. Available: https://ieeexplore.ieee.org/document/8950978.

[72] R. Xu, F. Han, and N. Dandapanthula, “Containerizing HPC Applications with Singularity,” Dell EMC HPC Inovation Lab, Tech. Rep., Oct. 2017. [Online]. Available: https://dl.dell.com/manuals/all-products/esuprt_solutions_int/esuprt_solutions_int_solutions_resources/high-computing-solution-resources_white-papers10_en-us.pdf.

[73] A. I. Forrester, A. Sóbester, and A. J. Keane, “Multi-fidelity optimization via surrogate modelling,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2088, pp. 3251–3269, Dec. 2007, ISSN: 1364-5021. DOI: 10.1098/RSPA.2007.1900. [Online]. Available: https://dx.doi.org/10.1098/rspa.2007.1900.

[74] S. Surjanovic and D. Bingham, *Forrester et al. (2008) Function*, 2013. [Online]. Available: https://www.sfu.ca/~ssurjano/forretal08.html.

[75] H. Dong, B. Song, P. Wang, and S. Huang, “Multi-fidelity information fusion based on prediction of kriging,” *Structural and Multidisciplinary Optimization 2014 51:6*, vol. 51, no. 6, pp. 1267–1280, Dec. 2014, ISSN: 16151488. DOI: 10.1007/s00158-014-1213-9. [Online]. Available: https://link.springer.com/article/10.1007/s00158-014-1213-9.

[76] J. Zheng, X. Shao, L. Gao, P. Jiang, and Z. Li, “A hybrid variable-fidelity global approximation modelling method combining tuned radial basis function base and kriging correction,” *Journal of Engineering Design*, vol. 24, no. 8, pp. 604–622, Aug. 2013, ISSN: 09544828. DOI: 10.1080/09544828.2013.788135. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/09544828.2013.788135.

[77] S. Xiong, P. Z. Qian, and C. F. Wu, “Sequential Design and Analysis of High-Accuracy and Low-Accuracy Computer Codes,” *Technometrics*, vol. 55, no. 1, pp. 37–46, 2013, ISSN: 00401706. DOI: 10.1080/00401706.2012.723572. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00401706.2012.723572.

[78] C. Park, R. T. Haftka, and N. H. Kim, “Remarks on multi-fidelity surrogates,” *Structural and Multidisciplinary Optimization*, vol. 55, no. 3, pp. 1029–1050, Mar. 2017, ISSN: 16151488. DOI: 10.1007/s00158-016-1550-y. [Online]. Available: https://doi.org/10.1007/s00158-016-1550-y.

[79] M. D. Morris, T. J. Mitchell, and D. Ylvisaker, “Bayesian Design and Analysis of Computer Experiments: Use of Derivatives in Surface Prediction,” *Technometrics*, vol. 35, no. 3, pp. 243–255, 1993, ISSN: 15372723. DOI: 10.1080/00401706.1993.10485320. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00401706.1993.10485320.

A

Implementation

This appendix focuses on the implementation of the proposed infill strategy, the measures taken to ensure consistency and repeatability and a description of the hardware on which the tests were performed, starting with a description of the overall framework in section A.1, followed by the surrogate models in section A.2. Next, the implementation of the infill strategy and the optimisation algorithms are introduced in section A.3 and section A.4 respectively. The verification of the implemented method is discussed in chapter 5. Next, a short description is included of the data that is saved throughout the optimisation process in section A.5. Lastly, the measures taken to ensure consistency and repeatability across tests are discussed in section A.6, including a description of the hardware used.

A.1. Overall Framework

The implementation begins with an overarching view of how each component interacts with the others in the Surrogate-Based Optimisation (SBO) framework. It was chosen to create an object-oriented SBO framework that allows the selected infill strategies, surrogates, and optimisation algorithms to be implemented and used in every compatible combination interchangeably. The flow of the framework is illustrated in Figure A.1, with the corresponding pseudocode in Algorithm A.1. To configure the optimisation problem, a user would interact with the classes that serve as the problem configuration and the optimiser entry point: the singleton Config class and the SBO class, respectively. The components initialised in the SBO instance are the surrogate model, infill strategy and optimisation algorithm. All these components are classes for each selected algorithm, sharing a common interface within each category. The common interface ensures, for example, that different infill strategies can be used interchangeably (provided they are compatible with the problem definition) and that the framework can be easily extended to include additional infill strategies, optimisation algorithms, or surrogate models. The framework is implemented in Python 3.10 using the CPython interpreter and relies on several third-party libraries, including Numpy [66], SciPy [54], and the Surrogate Modelling Toolbox 2.0 (SMT) [1]. In addition, OpenMPI is used via mpi4py to enable parallel computation if available on the host and required by the problem configuration. The basic logic of the SBO class is outlined in Algorithm A.1, and the flow of the framework is visualised in Figure A.1.

Algorithm A.1 SBO framework.

Require: Problem definition: *config*, initial samples: S_0
Ensure: Best point x^* with objective value f^*

procedure SBO(*config*, S_0)
  surr ←InitSurrogate() ▷ Initialise surrogate model based on *config*
  infill ←InitInfill() ▷ Initialise infill strategy based on *config*
  opt ←InitOpt() ▷ Initialise optimisation algorithm based on *config*
  if $S_0 = \varnothing$ **then**
    X_0←LHS() ▷ Generate initial samples using LHS
    for $x \in X_0$ **do**
      x,value ← $f_{\text{fid}}(x)$
      S_0 ← $S_0 \cup \{(x, \text{value})\}$
    end for
  end if
  S_0←EnsureMinSamplesPerFidelity() ▷ Ensure S_0 is nested if required and contains enough samples
  Train *surr* on S_0
  x^*, f^*_{HF} ←arg min$_x$ *Opt*(*surr*, *infill*) ▷ Perform optimisation
  return x^*, f^*_{HF}
end procedure

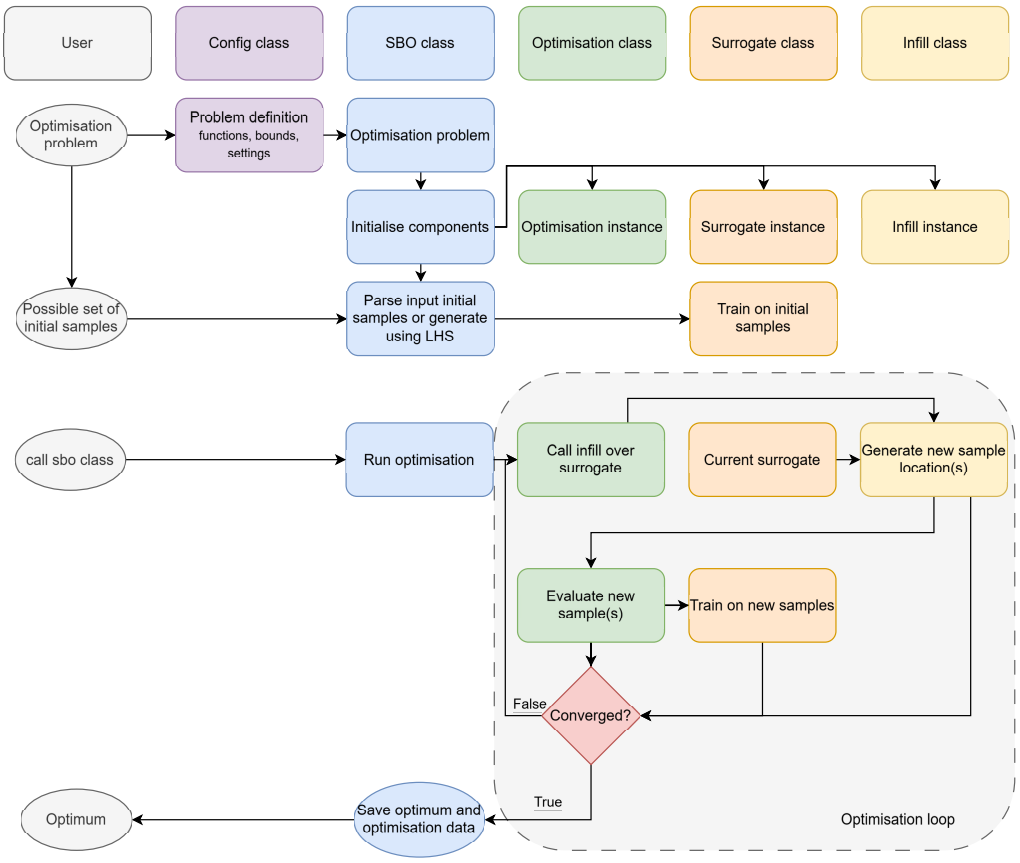


Figure A.1: SBO framework flow.

A.2. Surrogate Models

The first components used in the optimisation, which were implemented, are the surrogate models. The selected surrogate models are the Single-Fidelity (SF) Kriging (KRG) surrogate and the Multi-Fidelity (MF) Hierarchical Kriging (HK) model. The KRG surrogate is used in the SF reference test and as a base model for the MF HK model.

A.2.1. Kriging

The first of these surrogate models is the KRG surrogate model. Since the surrogate-based optimisation framework uses SMT [1] as a base, the implemented KRG model is the SMT KRG model. To ensure a consistent interface with the framework, a wrapper was written for the SMT KRG implementation.

A.2.2. Hierarchical Kriging

The next surrogate model implemented is the MF HK model introduced by Han and Görtz [22] and discussed in subsection 3.3.2. This implementation includes both the Gaussian exponential and the cubic-spline correlation introduced by Lophaven et al. [53]. The implementation uses the previously discussed SMT KRG model as the base surrogate model for the lowest fidelity. This base surrogate is then used to build up from the lowest-fidelity model to the High Fidelity (HF) model, completing the training of the full HK model. This training process is discussed in detail in subsection 3.3.2.

As discussed in the description of the training process, the hyperparameter θ has to be found by means of optimising the log-likelihood. Since this log-likelihood can be highly multi-modal, this was done using SciPy's differential evolution [54]. Han and Görtz [22] use a genetic algorithm to overcome this issue.

Differential evolution is a stochastic population-based optimisation algorithm, which does not use gradient information to find a minimum. It can search large regions of the design space, but often requires many function evaluations. The optimisation starts with a population of candidates, or vectors, with $\bar{v} \in \mathbb{R}^n$. In each iteration, these vectors are mixed according to a selected strategy, resulting in the mutated vector. In the next step, crossover is applied between the mutated vector and the reference vector to increase population diversity. This is achieved by combining the two vectors and selecting the corresponding element from each. This is done by taking n numbers from a binomial distribution on $[0, 1)$. For each element of the vector, the value from the mutated vector is taken if the corresponding value from the binomial distribution is less than the set recombination constant. Otherwise, the element is taken from the reference vector. This results in the candidate vector. Next, the fitness of the trial vector is determined. Namely, if the trial is better than the original vector, the original vector is replaced by the trial vector. In the original algorithm from Storn and Price [55], at the end of each full iteration, the best solution is updated from the new population, so it is updated once per iteration. The implementation in SciPy (1.15.3), however, continually updates the best solution throughout an iteration. This can lead to faster convergence, since the trial vectors can immediately benefit from the improved solution.

A.3. Infill Strategies

The implementation of the selected infill strategies, discussed in chapter 4, is discussed next. This discussion of the implementation starts with the random-point infill strategy, followed by the Expected Improvement (EI) and lastly the Generalised Expected Improvement (GEI) infill strategies.

A.3.1. Random Point

As a baseline reference, a random infill strategy was implemented. This infill strategy illustrates a worst-case scenario in which a random point is added to the set of sample points until convergence is achieved.

This is achieved by first selecting a sample coordinate on the closed interval $\bar{x}_{\text{sample}} \in \left[\bar{x}_{LB}, \bar{x}_{UB}\right]$, using Numpy's [66] `numpy.random.uniform()`. This function draws samples from a uniform distribution on the half-open interval $\bar{x} \in \left[\bar{x}_{LB}, \bar{x}_{UB}\right)$. To ensure that the upper-bound of the design space can be sampled, a small nugget was added to the upper bound used in the sampling method. Next, the fidelity level was selected using `numpy.random.randint()`, which returns an integer in the half-open interval $x \in \left[x_{LB}, x_{UB}\right)$. The lower and upper bounds were set to 0 and the total number of fidelities n_{fid} used in the optimisation. Since the numbering of the fidelities starts at 0, all fidelities can be sampled as the maximum number returned by `numpy.random.randint()` is $n_{fid} - 1$, which corresponds to the lowest

fidelity level. Combining the two would yield a new sample point. To ensure consistent behaviour across runs, the method uses the same random seed throughout the optimisation process. If the selected sample was already included in the set of sample points, this process would be repeated until a unique sample point was selected. This results in the pseudo-code implementation in Algorithm A.2:

Algorithm A.2 Random Infill Strategy.

Require: Sampled points: S, Lower bound: $\bar{x}_{LB}$, Upper bound: $\bar{x}_{UB}$, Number of fidelities: n_{fid}
Ensure: New sample point(s): S_{new}

```
unique ← False
while not unique do
    x̄_new ← randunif(x̄_LB, x̄_UB + 1e−8)
    fid_new ← randint(0, n_fid)
    S_new ← (x̄_new, fid_new)
    if S_new ∉ S then
        unique ← True
    end if
end while
return S_new
```

A.3.2. Expected Improvement

The next implemented infill strategy is the EI infill strategy (discussed in more detail in subsection 3.2.3). This infill strategy selects a point at the location $\bar{x}$ that maximises expected improvement over the design space, thereby balancing exploration and exploitation. Since the EI-value, calculated using (3.28), can be highly multi-modal, the optimum was found using SciPy's [54] `scipy.optimize.differential_evolution()`. This method implements the differential evolution algorithm introduced by Storn and Price in [55] and discussed in subsection A.2.2.

A.3.3. Generalised Expected Improvement

The GEI, discussed in Equation (3.32), was implemented similarly to the previously discussed EI. As this infill value is to be maximised and the landscape can be highly multi-modal, `scipy.optimize.differential_evolution()` is used for the optimisation of the GEI value. The resulting x^* is the location of the new infill point.

A.3.4. Two-Step

The Two-Step infill strategy was implemented using the same function used in the reference by Garbo et al. [31], namely `scipy.spatial.distance.jensenshannon`. Note that this function returns the Jensen-Shannon distance (JSd), which is the square root of the Jensen-Shannon Divergence (JSD). This metric is calculated using the probability vectors created using the probability density function defined on:

$$\begin{aligned} \text{LB} &= \min\left(\mu_{\text{HF}} - 3\sigma, \mu_{fid} - 3\sigma\right), \\ \text{UB} &= \max\left(\mu_{\text{HF}} + 3\sigma, \mu_{fid} + 3\sigma\right). \end{aligned} \tag{A.1}$$

Using the HF probability vector and the probability vector for fidelity fid the JSd is calculated. If the value is below the set threshold, the fidelity is deemed accurate enough. After performing this test for each fidelity level, the lowest-fidelity option is selected as the sample point. The mean prediction and standard deviation used in this calculation are the mean and standard deviation returned by the surrogate at the selected sample location. This sample location is first selected using the SF infill strategy.

A.4. Optimisation Algorithms

Next, the implementation of the optimisation algorithms is discussed. Two optimisation algorithms are implemented, namely a sequential Efficient Global Optimisation (EGO) as a reference and the selected Multi-Point (MP) algorithm, as introduced in section 4.4, resulting in an asynchronous EGO. To verify both cases, the previously verified KRG surrogate and EI infill strategy were used. The first optimisation algorithm discussed is the sequential EGO.

A.4.1. Sequential Efficient Global Optimisation

The implemented sequential EGO algorithm is a Single-Point (SP) optimisation algorithm as discussed in subsection 3.2.3. This algorithm selects a new sample point according to a specified infill strategy. After which, a check is performed to determine if the selected point is in the set of sampled points. This was shown to occur for the Two-Step infill strategy, since the variance of the HF HK model at the location of a Low Fidelity (LF) sample is not 0. This can cause the point to be reselected by the infill criterion, which is purely based on the HF HK model. If such a point were selected, the infill strategy would deem it a promising area for placing a sample. The sample would be evaluated at one fidelity level higher, adding information without resampling an already sampled point.
The selected sample point is then computed and added to the surrogate model, after which a new sample point is selected based on the infill strategy. This process is repeated until the convergence criteria discussed in section 6.4 are met. After which, the location of the optimum of the surrogate is determined, and the corresponding HF function value is calculated and returned as the optimum. Thus, the total number of function evaluations, excluding the initial samples, is $n_{\text{iter}} + 1$. The whole process is outlined in Algorithm A.3.

Algorithm A.3 Sequential EGO.

Require: Surrogate model: *surr*, infill criterion: *infill*, initial sample set: S_0, with coordinates X_0
Ensure: Best point x^* with objective value f^*

Train *surr* on S_0
$f^* \leftarrow \min_{x \in X_0} f(x), \quad k \leftarrow 0$
converged $\leftarrow$ `False`
while $\neg$ converged **do**
  $\{x_{\text{new}}\}, \mathit{infill}^* \leftarrow \arg\max_x \mathit{infill}(x \mid \mathit{surr})$ ▷ Optimise infill criterion
  $x_{\text{new}} \leftarrow \text{RemoveDuplicates}(x_{\text{new}}, X_k)$ ▷ Remove duplicate points
  for each candidate $x_i \in \{X_{\text{new}}\}$ **do**
    $f_i \leftarrow f_{\text{fid}}(x_i)$ ▷ Evaluate objective at prescribed fidelity
    $S_{k+1} \leftarrow S_k \cup \{(x_i, f_i)\}$
  end for
  Retrain *surr* on S_{k+1}
  $f_k^* \leftarrow \min_{x \in X_{k+1}} f(x)$
  if Convergence **then**
    converged $\leftarrow$ `True`
    break ▷ Convergence criterion met
  end if
  $k \leftarrow k + 1$
end while
$x^*_{\mathit{surr}} \leftarrow \arg\min_x \hat{f}(x)$ ▷ Surrogate optimum
Evaluate $f_{\text{HF}}(x^*_{\mathit{surr}})$ update f^* if improved
return $x^* = \arg\min f^*$

A.4.2. Asynchronous Efficient Global Optimisation

The last main element implemented was the asynchronous EGO algorithm based on the method from Przysowa et al. [38]. This method uses the same basic principle to find the optimum as the sequential EGO discussed previously, while aiming to reduce the clock time required by utilising parallel computing. The implementation requires at least two ranks to function. The coordinator rank (rank 0) tracks the surrogate and distributes new points to be sampled to the available ranks. The other ranks only compute the corresponding function value at the desired new sample location and return the new information to rank 0. This is repeated until convergence is achieved. In more detail, rank 0 keeps track of the worker ranks (rank 1 through $n_{\text{ranks}} - 1$), which are available to calculate a new point. If such a rank is available, rank 0 will determine the next sample point using the specified infill criterion on the current surrogate and perform the same duplicates check as used in EGO Sequential. The resulting point will be sent to the worker rank for calculation using the Message Passing Interface (MPI) with OpenMPI, implemented in Python via mpi4py. This process is then repeated until either no worker ranks are available or a convergence criterion is met.

During this process, rank 0 monitors whether a worker rank has sent a new sample point back to rank 0. If such a new point is available, it is added to the surrogate's training data, and the surrogate is retrained on the updated training data. This ensures the next sample locations are always calculated based on the most current surrogate. However, the surrogate may not yet be updated when selecting a new sample location. This is, for example, the case at the start of the optimisation, when all worker ranks are available, but no update has been made to the surrogate. In these cases, the new sample location will be generated using a temporarily updated surrogate, obtained via the Kriging Believer (KB) or Constant Liar (CL) method (as specified by the user), as discussed in section 3.5. This temporary surrogate will be updated with all pending points, whose values are determined by the KB or CL algorithm. This ensures that the new sample location is selected on a surrogate with additional information, since otherwise the previously selected sample location would be reselected.
This yields the algorithm for the coordinator outlined in Algorithm A.4 and Algorithm A.5 for the worker ranks, with the flow of information between ranks visualised in Figure A.2. This implementation requires at least two ranks for the optimisation process. Note that when two ranks are available, the algorithm essentially operates as the previously discussed sequential EGO. However, there is likely a slight increase in wall-clock time due to the computational overhead of coordinating the sample points and inter-rank communication.

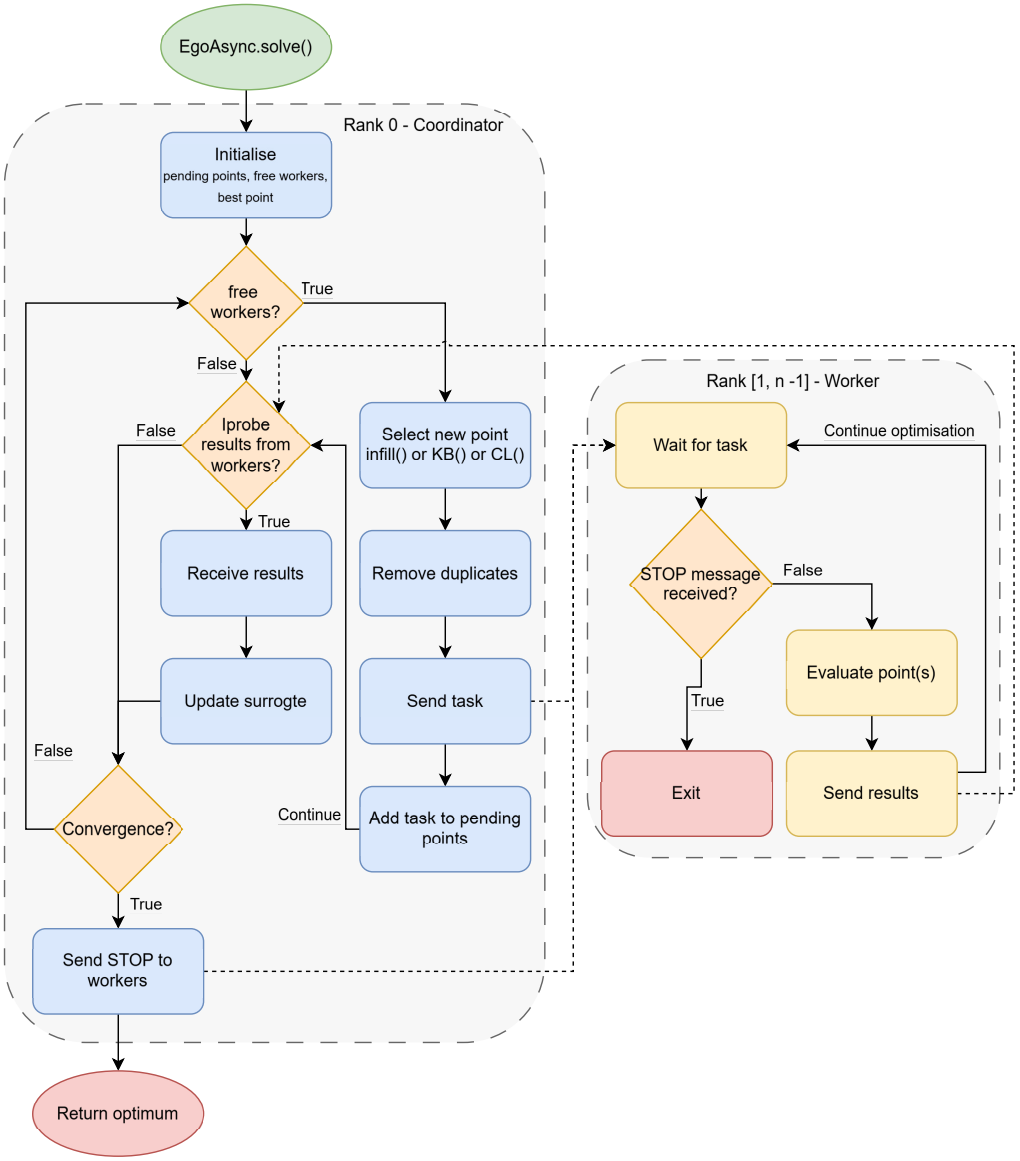


Figure A.2: Flow of Asynchronous EGO.

Algorithm A.4 Asynchronous EGO — Coordinator (rank 0).

Require: Surrogate *surr*, infill criterion *infill*, initial samples S_0, with coordinates X_0, worker set $W = \{1, \dots, n-1\}$
Ensure: Best point x^* with objective value f^*

Train *surr* on S_0
$f^* \leftarrow \min_{x \in X_0} f(x)$, pending $\leftarrow \varnothing$, free $\leftarrow W$
surr_updated $\leftarrow$ `True`, converged $\leftarrow$ `False`
while $\neg$ converged **do**
  if free $\neq \varnothing$ **then**
    if surr_updated **then**
      $x_{\text{new}}, \mathit{infill}^* \leftarrow \arg\max_x \mathit{infill}(x \mid \mathit{surr})$ ▷ Optimise infill criterion
    else
      if use KB **then**
        $x_{\text{new}}, \mathit{infill}^* \leftarrow \text{KB}(\mathit{surr}, \text{pending})$ ▷ KB multipoint strategy
      else
        $x_{\text{new}}, \mathit{infill}^* \leftarrow \text{CL}(\mathit{surr}, \text{pending})$ ▷ CL multipoint strategy
      end if
    end if
    $x_{\text{new}} \leftarrow \text{RemoveDuplicates}(x_{\text{new}}, X_k, \text{pending})$
    $w \leftarrow$ free.pop()
    Send$(x_{\text{new}}, \text{dest} = w, \text{tag} = 11)$ ▷ Dispatch task to worker w
    pending $\leftarrow$ pending $\cup \{x_{\text{new}}\}$
    surr_updated $\leftarrow$ `False`
  end if
  if MessageAvailable(tag $= 22$) **then** ▷ Non-blocking probe for results
    w, results $\leftarrow$ Recv(tag $= 22$)
    free $\leftarrow$ free $\cup \{w\}$
    pending $\leftarrow$ pending $\setminus$ results
    $S_{k+1} \leftarrow S_k \cup$ results; retrain *surr* on S_{k+1}
    surr_updated $\leftarrow$ `True`
    $f^* \leftarrow \min\big(f^*, \min_{\mathbf{r} \in \text{results}} f(r)\big)$
    if Convergence criterion met **then**
      converged $\leftarrow$ `True`
      $x^*_{\text{surr}} \leftarrow \arg\min_x \hat{f}(x)$ ▷ Surrogate optimum
      Evaluate $f_{\text{HF}}(x^*_{\text{surr}})$; update f^* if improved
      break
    end if
  end if
  sleep(0.1) ▷ Avoid busy-waiting
end while

▷ Shutdown phase
for $w \in W$ **do**
  Send("STOP", dest $= w$, tag $= 11$) ▷ Signal each worker to stop
  Await Recv("STOPPED", source $= w$, tag $= 22$)
  Collect any final results and add to S
end for

return $x^* = \arg\min f^*$

Algorithm A.5 Asynchronous EGO — Worker (rank $w \geq 1$).

Require: Rank w, objective function f
Ensure: `None` (results returned via MPI)

while `True` **do**
  task ← Recv(source $= 0$, tag $= 11$) ▷ Block until task arrives from coordinator
  if task = "STOP" **then**
    Send("STOPPED", dest $= 0$, tag $= 22$) ▷ Confirm shutdown to coordinator
    break
  else
    for each $x_i \in$ task **do**
      $f_i \leftarrow f_{\text{fid}}(x_i)$ ▷ Evaluate objective at prescribed fidelity
    end for
    Send$\big(\{(x_i, f_i)\}$, dest $= 0$, tag $= 22\big)$ ▷ Return results to coordinator
  end if
end while
return `None`

A.5. Performance Measurement

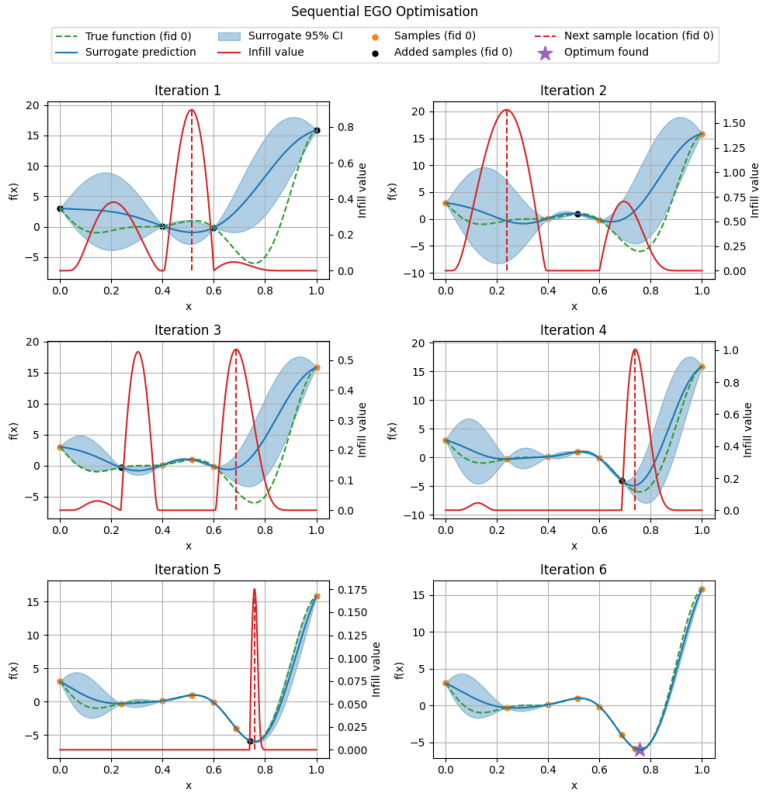
As discussed in section 6.5, the performance metrics measured are the wall-clock time of the optimisation process and the returned optimum. For each run, with a unique random seed, these metrics are recorded and saved. In addition, for each iteration of the sequential EGO, or for each new point added in the async EGO, the current set of training data, convergence criteria, surrogate optimum and time elapsed are recorded and saved. All points in the training data include the start and end times of their computation, as well as the rank at which they were computed. The resulting data saved includes the problem definition and settings used by the instance of the config class and is stored as a serialised pickle file. The timing included is taken from the Python module `time`. The start time is taken before the optimisation loop begins, i.e., after initialisation and the acquisition of the initial samples (if not provided as input by the user). The end time is set to the end of the optimisation loop, after the final optimum point is evaluated using the HF function. Between these points, all operations are relevant to the optimisation problem; no Input/Output (I/O) actions are performed, as the optimisation data is saved only after the optimisation process has finished.

A.6. Repeatability and Consistency

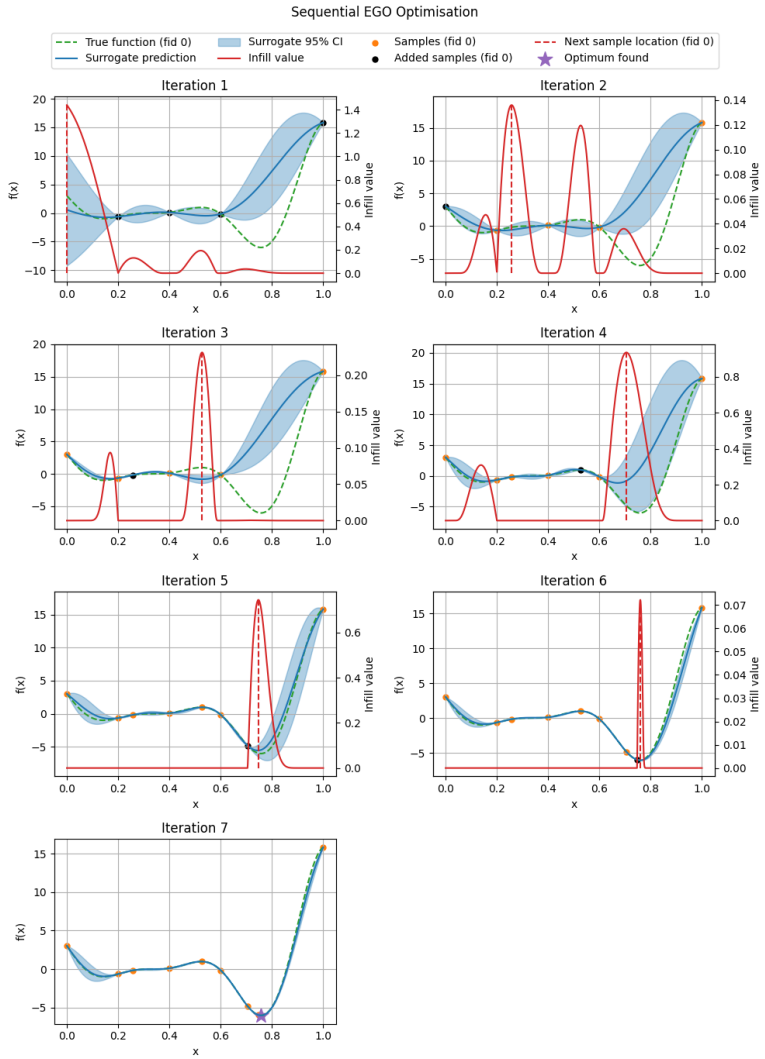
To ensure repeatable and consistent results, two measures have been taken. First, the tests were run for a set of random seeds. These seeds fix the pseudorandom aspects of the optimisation, ensuring consistent results across runs with the same seed. Secondly, the testing environment can affect the measured performance of the tests; to mitigate this, the tests have been run on the High Performance Computing (HPC) environment of TU Delft; DelftBlue [67].

A.6.1. Random Seeds

The runs have been run for 30 random seeds. The random seeds affect several parts of the optimisation process. Firstly, if no initial samples are passed by the user to the SBO class, an initial set of samples is generated using Latin Hypercube Sampling (LHS). This LHS method uses the random seed to control the pseudo-random pairings for the coordinates. The set of initial points can affect the optimisation process and possibly result in a different number of iterations, wall-clock time or a different returned optimum. A visualisation of the difference is included for the Forrester function in Figure A.3. Here, Figure A.3a starts with initial samples $\mathbf{X}_0 = \{0, 0.4, 0.6, 1\}$ and Figure A.3b with samples $\mathbf{X}_0 = \{0.2, 0.4, 0.6, 1\}$. In this example, there is no difference in the resulting optimum; however, the starting conditions result in a different starting surrogate. This difference results in different infill samples to be added, ultimately resulting in the second set of initial samples requiring an additional iteration.



(a) Using initial samples $\mathbf{X}_0 = \{0, 0.4, 0.6, 1\}$.



(b) Using initial samples $\mathbf{X}_0 = \{0.2, 0.4, 0.6, 1\}$.

Figure A.3: Difference Forrester function optimisation for different sets of initial samples.

The second aspect of the optimisation process influenced by the different random seed is the internal optimisations in the overall optimisation process. These internal optimisations, the maximisation of the infill strategy and training of the surrogate(s), depend on SciPy's differential evolution [54]. As discussed in subsection A.2.2, differential evolution starts with a random population; thus, this population is altered by changing the random state, potentially resulting in a different optimum or a different computational cost for the internal optimisation. These differences could lead to one combination of optimiser, surrogate, and infill strategy outperforming another for a given seed, whereas another seed could yield the reverse. To mitigate this result, the test optimisations were run for 30 different random seeds. The determination of this number of random seeds is discussed in section 6.6.

A.6.2. Consistent Environment

The second measure taken to ensure repeatability and consistency is to perform the tests in a consistent environment. Since hardware and library versions can affect the implementation's performance, to this end, all tests were performed on the second compute partition of DelftBlue [67]. This second compute partition is the second phase of the DelftBlue HPC environment and comprises 90 Intel Xeon compute nodes, equipped with Intel Xeon 6448Y Central Processing Units (CPUs), with each node having 64 CPU cores and 250 GB of Random-Access Memory (RAM). The tests were performed with varying numbers of CPU cores (1, 8, or 16), with 1 GB of RAM allocated per CPU core. Since the test is scheduled by the Simple Linux Utility for Resource Management (SLURM) scheduler, a task may be assigned to cores across several nodes. This can again affect the performance, since this would require inter-node MPI, whereas other tests can use solely intra-node MPI.

Inter-node MPI is the communication between HPC server nodes and uses the InfiniBand interconnect (HDR/HDR100) to pass messages, whereas intra-node MPI is MPI on one specific node, utilising the shared memory of the node instead of the interconnect connection. Although the InfiniBand connection will likely have a negligible impact on wall-clock time [68], this form of communication was deemed undesirable because the environment would be inconsistent between tests. Therefore, the tests were run exclusively on a single node.

In addition to ensuring consistency and repeatability, the required libraries and environment were defined and packaged into an Apptainer (previously Singularity) container. This removes the requirement for the modules that can be loaded from the HPC environment. Relying on these modules would result in an inconsistent environment if they were removed or updated. In addition, the Apptainer resolves the issue that, if the implementation were used in another HPC environment that lacks all required modules, those modules would need to be added. Since the Apptainer includes them in the environment, this is not necessary, improving the portability.

The Apptainer container was designed for use in HPC environments, providing native support for HPC schedulers and MPI. In addition, because an Apptainer container is run with the user's credentials, it does not require root access, unlike some other containers, such as Docker. The container also exhibits no or negligible performance loss relative to a bare-metal implementation, as measured by wall-clock time, with some studies reporting a minimal increase in memory overhead [69], [70], [71], [72].

B

Manual Verification Case for Hierarchical Kriging

This appendix includes the full calculations used to verify the Hierarchical Kriging (HK) implementation. As discussed in subsection 5.1.2, there is a discrepancy in the found β_0 of the implementation and the reference by Han and Görtz [22]. The methodology and reasoning behind the calculations are discussed in detail in subsection 5.1.2. The numerical results and calculations are introduced here.
The manual calculations presented here are done on a simple set of functions (High Fidelity (HF): Equation (B.1), Low Fidelity (LF): Equation (B.2)) and a small sample set. As in the numerical verification cases presented by Han and Görtz [22], the selected sample sets for the low- and high-fidelity data are nested, that is, $\mathbf{X}_{\mathsf{HF}} \subseteq \mathbf{X}_{\mathsf{LF}}$.

$$f_{\mathsf{HF}}(x) = 2x + 5. \tag{B.1}$$

$$f_{\mathsf{LF}}(x) = 0.5 f_{\mathsf{HF}}(x) + 2x - 10. \tag{B.2}$$

With the sample locations and corresponding sample vector:

$$\mathbf{X}_{\mathsf{HF}} = [0.5, 2.5]^{\mathsf{T}}, \tag{B.3}$$

$$\mathbf{X}_{\mathsf{LF}} = [0.5, 2.5]^{\mathsf{T}}, \tag{B.4}$$

$$\mathbf{y}_{\mathbf{s}} = \left[f_{\mathsf{HF}}\left(\mathbf{x}^1\right), f_{\mathsf{HF}}\left(\mathbf{x}^2\right)\right]^{\mathsf{T}} = [6, 10]^{\mathsf{T}}. \tag{B.5}$$

Since the base model is an ordinary Kriging model, which is an interpolation model, and the HF sample set is a subset of the LF sample set, the regression matrix for the Kriging predictor ($\mathbf{F}$) becomes;

$$\mathbf{F} \coloneqq \left[\hat{y}_{\mathsf{LF}}\left(\mathbf{x}^1\right), \hat{y}_{\mathsf{LF}}\left(\mathbf{x}^2\right)\right]^{\mathsf{T}} = \left[f_{\mathsf{LF}}\left(\mathbf{x}^1\right), f_{\mathsf{LF}}\left(\mathbf{x}^2\right)\right]^{\mathsf{T}} = [-6, 0]^{\mathsf{T}}. \tag{B.6}$$

After defining the input for training the HK model, the next step taken is to calculate the correlation matrix (B.7). This was done using the cubic spline correlation function used by Han and Görtz [22] and defined by Lophaven et al. [53]; (B.8).

$$R\left(\mathbf{x}, \mathbf{x}'\right) = \prod_{k=1}^{m} R_k\left(\theta_k, x_k - x'_k\right) \quad \text{with } x \in \mathbb{R}^m. \tag{B.7}$$

$$R_k(\theta_k, x_k - x'_k) = \begin{cases} 1 - 15\xi_k^2 + 30\xi_k^3 & \text{for } 0 \le \xi_k \le 0.2 \\ 1.25(1 - \xi_k)^3 & \text{for } 0.2 < \xi_k < 1 \\ 0 & \text{for } \xi_k \ge 1 \end{cases} \quad \text{where } \xi_k = \theta_k |x_k - x'_k|. \tag{B.8}$$

Note that this correlation function depends on the hyperparameter vector θ. In this example, it is assumed to be equal to $\theta = [0.4]^{\mathsf{T}}$. In the training process of the HK model, this hyperparameter is chosen by maximising the log-likelihood function (B.9), resulting in (B.10). In the implementation, this optimisation was performed using `scipy.optimize.differential_evolution` from the Python library SciPy

[54]. This optimisation was chosen because the log-likelihood can be highly multi-modal. Using a population-based optimisation algorithm may yield a better solution than a gradient-based method. The reference [22] used a genetic algorithm for this same reason.

$$\ln|L(\theta)| = -n \ln \sigma^2(\theta) - \ln|\mathbf{R}(\theta)|. \tag{B.9}$$

$$\theta = \arg\max_{\theta} \{\ln[L(\theta)]\}. \tag{B.10}$$

To verify that the log-likelihood function implementation is correct, a comparison was made between a manual calculation and the implementation's output. Since the log-likelihood depends on σ^2 and the correlation matrix, first the correlation matrix was calculated;

$$R = \begin{bmatrix} 1 & 0.01 \\ 0.01 & 1 \end{bmatrix}. \tag{B.11}$$

After which σ^2 could be calculated, which in turn is depends on β_0 and θ via (B.12) with β_0 defined as (B.13).

$$\sigma^2\left(\theta, \beta_0\right) = \frac{1}{n}\left(\mathbf{y}_s - \beta_0 \mathbf{F}\right)^{\intercal} \mathbf{R}^{-1}\left(\mathbf{y}_s - \beta_0 \mathbf{F}\right). \tag{B.12}$$

.

$$\beta_0(\theta) = \left(\mathbf{F}^{\intercal}\mathbf{R}^{-1}\mathbf{F}\right)^{-1} \mathbf{F}^{\intercal}\mathbf{R}^{-1}\mathbf{y}_s. \tag{B.13}$$

This resulted in the Table B.1 tabulated values for the log-likelihood, σ^2 and β_0. Since the manual calculations and the values returned by the implementation are equal, it is concluded that the difference in the β_0 value for the reference cases [22] and the custom implementation as discussed in subsection 5.1.2, is caused by the found value of θ since this value is obtained via optimisation, which is the only non-deterministic factor in the HK model's training. The reference shortly mentions the optimisation process of (B.10) with: 'we solve this optimization problem [(B.10)] approximately by using a genetic algorithm (...)'. To verify the effect of the chosen optimisation method for the hyperparameter, the optimisation has been run for a set of 50 different random seeds as well as by using a different optimisation algorithm; namely `scipy.optimize` [54]. These different cases did not result in a meaningful difference in the found and thus not in the resulting β_0.

Table B.1: Comparison of computed values.

Parameter	Manually computed value	Implementation computed value
log-likelihood	-7.824	-7.824
β_0	-0.9833	-0.9833
σ^2	50.0	50.0

C

Numerical Test Functions

This appendix introduces the test functions included in the Python library MF2 [21], which are used in chapter 7, starting with the 1-dimensional Forrester function in section C.1, followed by the 2-dimensional Bohachevsky, Booth, Branin, Currin, Himmelblau and Six Hump Camelback functions in section C.2, section C.3, section C.4, section C.5, section C.6 and section C.7 respectively. Next, the 4-dimensional functions, Park91A and Park91B, are discussed in section C.8 and section C.9. Lastly, the Hartmann6 6-dimensional function is discussed in section C.10 followed by the 8-dimensional Borehole problem in section C.11.

C.1. Forrester

The High Fidelity (HF) and Low Fidelity (LF) Forrester functions ((C.1) and (C.2) respectively) were introduced by Forrester et al. [73].

$$f_{\mathsf{HF}}(x) = (6x-2)^2 \sin(12x-4). \tag{C.1}$$

$$f_{\mathsf{LF}}(x) = 0.5 f_{\mathsf{HF}}(x) + 10(x-0.5) - 5. \tag{C.2}$$

The HF function is a 1-dimensional, multi-modal test function and features one global and one local minimum and a zero-gradient inflection point [74]. The function is defined on $x \in [0,1]$ and is visualised in Figure C.1.

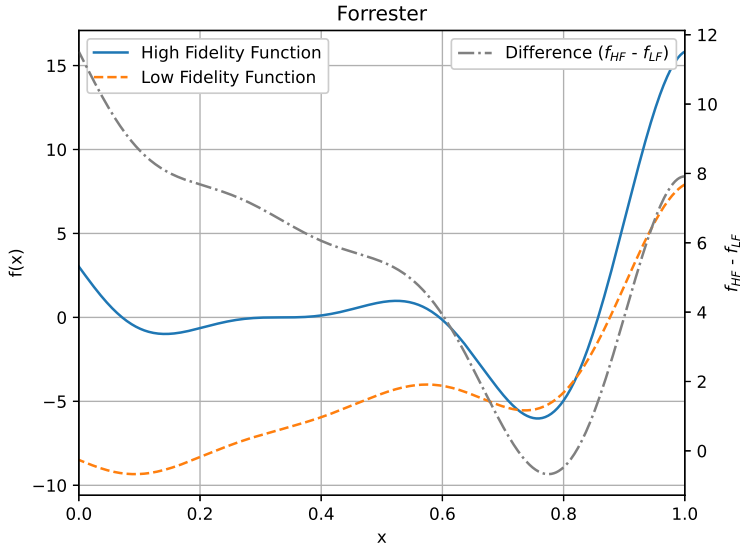


Figure C.1: Forrester high and low fidelity functions.

C.2. Bohachevsky

The Bohachevsky is a 2-dimensional bowl-shaped test function defined on $x \in [-5, 5] \times [-5, 5]$ and illustrated in Figure C.2, with HF and LF functions (C.3) and (C.4) defined by Dong et al. [75] and Zheng et al. [76].

$$f_{\text{HF}}\left(x_1, x_2\right) = x_1^2 + 2x_2^2 - 0.3\cos\left(3\pi x_1\right) - 0.4\cos\left(4\pi x_2\right) + 0.7. \tag{C.3}$$

$$f_{\text{LF}}\left(x_1, x_2\right) = f_{\text{HF}}\left(0.7x_1, x_2\right) + x_1 x_2 - 12. \tag{C.4}$$

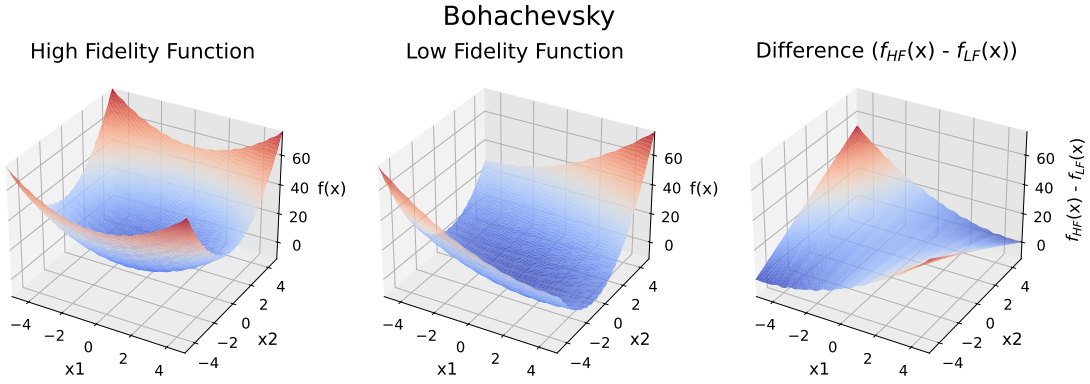


Figure C.2: Bohachevsky high and low fidelity functions.

C.3. Booth

The Booth function is a plate-shaped function and is defined in MF2 by the HF and LF functions, (C.5) and (C.6) respectively, as introduced by Dong et al. [75] and Zheng et al. [76]. These functions are evaluated on $x \in [-10, 10] \times [-10, 10]$ and are figured in Figure C.3. The known optimum of the HF function is $f\left(1, 3\right) = 0$.

$$f_{\text{HF}}\left(x_1, x_2\right) = \left(x_1 + 2x_2 - 7\right)^2 + \left(2x_1 + x_2 - 5\right)^2. \tag{C.5}$$

$$f_{\text{LF}}\left(x_1, x_2\right) = f_{\text{HF}}\left(0.4x_1, x_2\right) + 1.7x_1x_2 - x_1 + 2x_2. \tag{C.6}$$

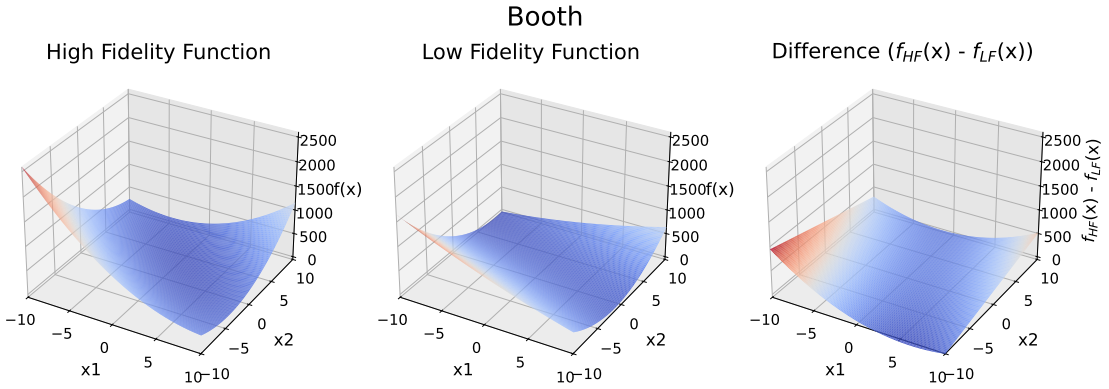


Figure C.3: Booth high and low fidelity functions.

C.4. Branin

The Branin is a 2-dimensional function with three global optima. The HF and LF functions are described by (C.8) and (C.29) with base function (C.7) evaluated on $x \in [-5, 10] \times [0, 15]$ [75], [76]. The three global optima are $f\left(x^*\right) = 0.397887$ at $x^* = (-\pi, 12.275)$, $(\pi, 2.275)$ and $(9.42478, 2.475)$.

$$f_b\left(x_1, x_2\right) = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos\left(x_1\right) + 10. \tag{C.7}$$

$$f_{\text{HF}}\left(x_1, x_2\right) = f_b\left(x_1, x_2\right) - 22.5x_2. \tag{C.8}$$

$$f_{\text{LF}} = f_b\left(0.7x_1, 0.7x_2\right) - 15.75x_2 + 20\left(0.9 + x_1\right)^2 - 50. \tag{C.9}$$

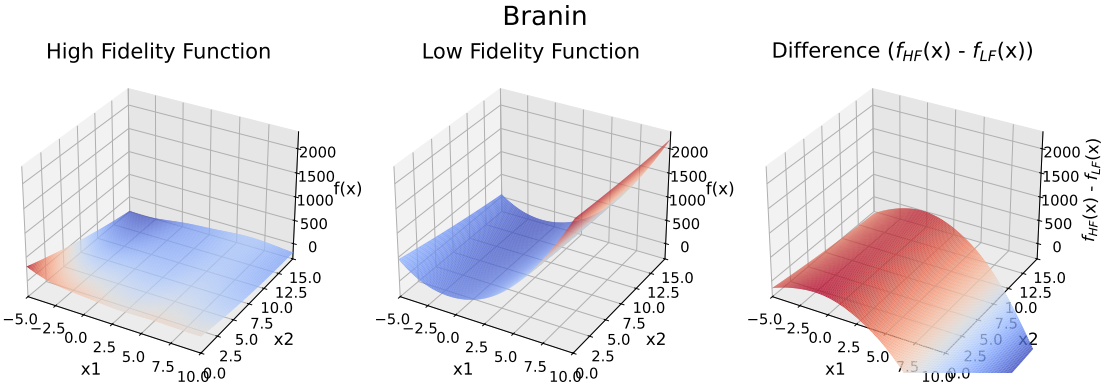


Figure C.4: Branin high and low fidelity functions.

C.5. Currin

The MF2 implementation of the Currin function was introduced by Xiong et al. [77] and is defined by (C.10) and (C.11) for the HF and LF functions, respectively. The input domain is on $x \in [0, 1] \times [0, 1]$.

$$f_{\text{HF}}\left(x_1, x_2\right) = \left(1 - \exp\left(-\frac{1}{2x_2}\right)\right)\frac{2300x_1^3 + 1900x_1^2 + 2092x_1 + 60}{100x_1^3 + 500x_2^2 + 4x_1 + 20}. \tag{C.10}$$

$$f_{\text{LF}}\left(x_1, x_2\right) = \frac{\begin{array}{c} f_{\text{HF}}\left(x_1 + 0.05, x_2 + 0.05\right) + f_{\text{HF}}\left(x_1 + 0.05, x_2 - 0.05\right) + \\ f_{\text{HF}}\left(x_1 - 0.05, x_2 + 0.05\right) + f_{\text{HF}}\left(x_1 - 0.05, x_2 - 0.05\right) \end{array}}{4}. \tag{C.11}$$

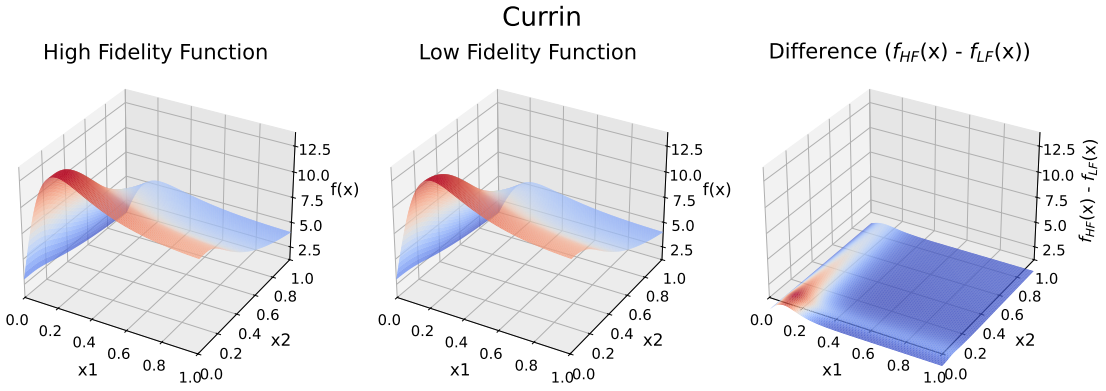


Figure C.5: Currin high and low fidelity functions.

C.6. Himmelblau

The Himmelblau function features four equal global minima in the domain $x \in [-4, 4] \times [-4, 4]$; $f\left(x^*\right) = 0$ for $x^* = (3, 2)$, $(-2.805118, 3.131312)$, $(-3.7791, -3.283186)$ and $(3.584428, -1.848126)$. The function is defined by HF (C.12) and LF (C.12) [75] and is illustrated in Figure C.6.

$$f_{\text{HF}}\left(x_1, x_2\right) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_2^2 + x_1 - 7\right)^2. \tag{C.12}$$

$$f_{\text{LF}}\left(x_1, x_2\right) = f_{\text{HF}}\left(0.5x_1, 0.8x_2\right) + x_2^3 - \left(x_1 + 1\right)^2. \tag{C.13}$$

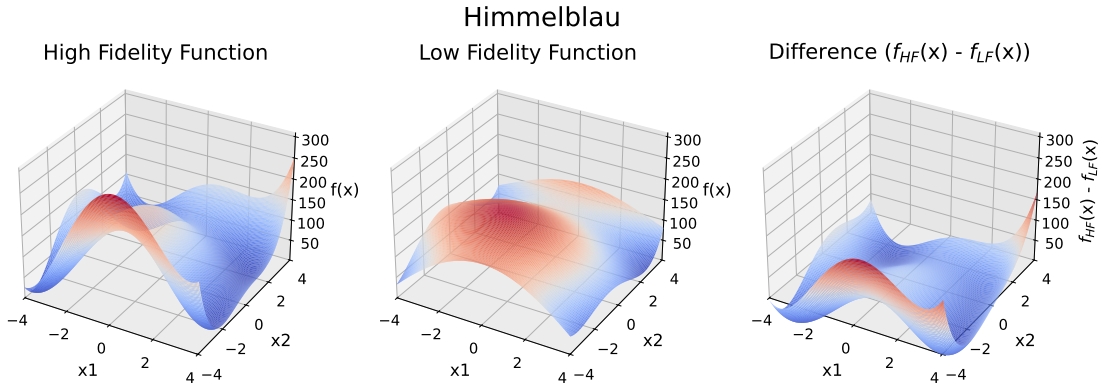


Figure C.6: Himmelblau high and low fidelity functions.

C.7. Six Hump Camelback

The last 2-dimensional test function is the six-hump camelback function. This is a valley shaped function on the domain $x \in [-2, 2] \times [-2, 2]$, with two global minima; $f\left(x^*\right) = -1.0316$ at $x^* = (0.0898, -0.7126)$ and $(-0.0898, 0.7126)$. The function is defined by (C.14) and (C.15) for the HF and LF functions, respectively [75]. The function is figured in Figure C.7.

$$f_{\text{HF}}\left(x_1, x_2\right) = 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1 x_2 - 4x_2^2 + 4x_2^4. \tag{C.14}$$

$$f_{\text{LF}}\left(x_1, x_2\right) = f_{\text{HF}}\left(0.7x_1, 0.7x_2\right) + x_1x_2 - 15. \tag{C.15}$$

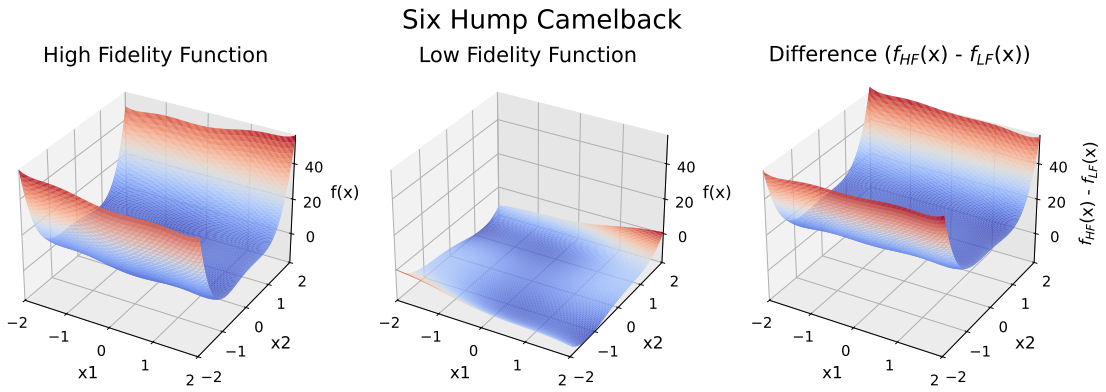


Figure C.7: Six Hump Camelback high and low fidelity functions.

C.8. Park91A

The first of the two 4-dimensional test functions is the Park91A function. This function is defined by (C.16) (HF) and (C.17) (LF) on $x \in [1\text{e}{-8}, 1] \times [0, 1] \times [0, 1] \times [0, 1]$ [77]. With an global optimum at $f\left(1\text{e}{-08}, 0, 0, 0\right) = 2.71828183\text{e}{-08}$.

$$f_{\text{HF}}\left(x_1, x_2, x_3, x_4\right) = \frac{x1}{2}\left(\sqrt{1 + \left(x_2 + x_3^2\right)}\frac{x_4}{x_1^2} - 1\right) + \left(x_1 + 3x_4\right)\exp\left(1 + \sin\left(x_3\right)\right). \tag{C.16}$$

$$f_{\text{LF}}\left(x_1, x_2, x_3, x_4\right) = \frac{1 + \sin\left(x_1\right)}{10} f_{\text{HF}}\left(x_1, x_2, x_3, x_4\right) - 2x_1 + x_2^2 + x_3^2 + 0.5. \tag{C.17}$$

C.9. Park91B

The last four-dimensional function is the Park91B function, with the LF function (C.19) the average of four HF functions (C.1), with slightly perturbed inputs [77] on $x \in [0, 1] \times [0, 1] \times [0, 1] \times [0, 1]$. With an optimum at $f\left(0, 0, 0, 0\right) = 0.6667$.

$$f_{\text{HF}}\left(x_1, x_2, x_3, x_4\right) = \frac{2}{3}\exp\left(x_1 + x_2\right) - x_4\sin\left(x_3\right) + x_3. \tag{C.18}$$

$$f_{\text{LF}}\left(x_1, x_2, x_3, x_4\right) = 1.2 f_{\text{HF}}\left(x_1, x_2, x_3, x_4\right) - 1. \tag{C.19}$$

C.10. Hartmann6

The six-dimensional test function is the Hartmann6 function, as defined by Park et al. [78] on $x \in [0.1, 1] \times [0.1, 1] \times [0.1, 1] \times [0.1, 1] \times [0.1, 1] \times [0.1, 1]$. The HF function is (C.20) and the LF function (C.24). With the A and P matrices (C.21) and (C.22) and α_i (C.23).

$$f_{\text{HF}}\left(x_1, \ldots, x_6\right) = -\frac{1}{1.94}\left[2.58 + \sum_{i=1}^{4}\alpha_i \exp\left(-\sum_{j=1}^{6} A_{ij}\left(x_j - P_{ij}\right)^2\right)\right]. \tag{C.20}$$

$$A = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}. \tag{C.21}$$

$$P = 10^{-4}\begin{pmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{pmatrix}. \tag{C.22}$$

$$\alpha = \begin{Bmatrix} 1.0 & 1.2 & 3.0 & 3.2 \end{Bmatrix}^T. \tag{C.23}$$

In the LF function, the f_{exp} is the approximation of the exponential term of the HF function and is defined in (C.25), with α_i (C.26).

$$f_{\mathsf{LF}}\left(x_1, \ldots, x_6\right) = -\frac{1}{1.94}\left[2.58 + \sum_{i=1}^{4} \alpha_i' f_{exp}\left(-\sum_{j=1}^{6} A_{ij}\left(x_j - P_{ij}\right)^2\right)\right]. \tag{C.24}$$

$$f_{exp}\left(\bar{x}\right) = \left[\exp\left(-\frac{4}{9}\right) + \left(\exp\left(-\frac{4}{9}\right)\frac{x+4}{9}\right)\right]^9. \tag{C.25}$$

$$\alpha' = \begin{Bmatrix} 0.5 & 0.5 & 2.0 & 4.0 \end{Bmatrix}^T. \tag{C.26}$$

The of the function is $f\left(0.2017, 0.15, 0.4769, 0.2753, 0.3117, 0.6573\right) = -3.0425$.

C.11. Borehole

The last test function is an eight-dimensional function which models the water flow through a Borehole drilled from the surface through two aquifers. This problem ((C.27), with the HF function (C.28)) was introduced by Morris et al. [79] with an LF model (C.29) by Xiong et al. [77]. The input domain is tabulated in Table C.1.

$$f_b\left(\bar{x}, A, B\right) = \frac{A * T_u\left(H_u - H_l\right)}{\log\left(\frac{r}{r_w}\right)\left(B + \frac{2L*T_u}{\log\left(\frac{r}{r_w}\right)r_w^2*K_w} + \frac{T_u}{T_l}\right)}, \text{ with } x = \left\{r_w, r, T_u, H_u, T_l, H_l, L, K_w\right\}. \tag{C.27}$$

$$f_{\mathsf{HF}}\left(\bar{x}\right) = f_b\left(\bar{x}, 2\pi, 1\right), \text{ with } x = \left\{r_w, r, T_u, H_u, T_l, H_l, L, K_w\right\}. \tag{C.28}$$

$$f_{\mathsf{LF}}\left(\bar{x}\right) = f_b\left(\bar{x}, 5, 1.5\right), \text{ with } x = \left\{r_w, r, T_u, H_u, T_l, H_l, L, K_w\right\}. \tag{C.29}$$

Table C.1: Input domain Borehole.

Parameter	Domain	Represents
r_w	$[0.05, 0.15]$ m	radius of Borehole
r	$[100, 5\text{e}4]$ m	radius of influence
T_u	$[6.3070\text{e}4, 115.6\text{e}4]$ m^2 yr^{-1}	transmissivity of upper aquifer
H_u	$[990, 1100]$ m	Potentiometric head of upper aquifer
T_l	$[63.1, 116]$ m^2 yr^{-1}	Transmissivity of lower aquifer
H_l	$[700, 820]$ m	Potentiometric head of lower aquifer
L	$[1120, 1680]$ m	Length of Borehole
K_w	$[9.855\text{e}3, 12.045\text{e}3]$ m yr^{-1}	Hydraulic conductivity of Borehole

D

Numerical Test Problems Additional Results

This appendix includes all the results for the set of numerical test functions. Note that as discussed in section 6.6, the Generalised Expected Improvement (GEI) based infill strategies have not been run for the required number of random seeds. This is also the case for the following functions:

- Park91B
- Hartmann6
- Borehole

This means that these results can only be used to gain a qualitative insight into the performance of these infill strategies and/or of these functions. No significant conclusion can be drawn from these specific results. This appendix starts with presenting the result tables in section D.1. This is followed by the visualisation of the convergence, returned optimum, total function evaluations, function evaluations per rank, time elapsed and time elapsed for a successful run ordered per function in section D.2.

D.1. Result Tables

The first table, Table D.1, tabulates the success rate for each function-infill strategy combination. The colour coding in this table is as follows:

- $100 - 80\%$: Green
- $79 - 50\%$: Orange
- $49 - 1\%$: Red
- 0%: Dark red

The next table in Table D.2 presents the mean elapsed time per run $\pm$ the standard deviation. The same colour coding is used to signify the success rate. Lastly in Table D.3 the mean elapsed time per successful run $\pm$ the standard deviation is tabulated.

Table D.1: Success rates per (function, algorithm) for the numerical testfunctions. Note the results for the Bohachevsky, Park91B, Hartmann6 and Borehole functions and the GEI based infill strategies can be skewed due to the lower number runs.

Algorithm	Forrester	Bohachevsky	Booth	Branin	Currin	Himmelblau	Six hump camelback	Park91a	Park91b	Hartmann6	Borehole
SF SP KRG Random	100%	84%	100%	100%	100%	98%	100%	100%	100%	0%	42%
MF SP HK Random	100%	62%	100%	100%	100%	100%	100%	100%	100%	0%	50%
SF SP KRG EI	100%	44%	84%	86%	88%	72%	81%	98%	84%	35%	74%
SF SP KRG GEI	100%	86%	95%	79%	93%	98%	81%	100%	100%	44%	98%
SF Async KRG EI n=8	100%	52%	97%	87%	100%	83%	100%	100%	100%	38%	71%
SF Async KRG EI n=16	100%	65%	97%	87%	93%	77%	97%	100%	100%	75%	93%
SF Async KRG GEI n=8	100%	62%	93%	57%	82%	61%	50%	94%	69%	0%	78%
SF Async KRG GEI n=16	100%	55%	96%	87%	79%	47%	82%	95%	81%	6%	69%
MF SP HK TwoStep EI	95%	19%	72%	49%	60%	56%	60%	79%	70%	28%	60%
MF Async HK TwoStep EI n=8	100%	33%	77%	75%	86%	68%	100%	90%	63%	73%	No data
MF Async HK TwoStep EI n=16	88%	53%	87%	65%	88%	60%	98%	87%	77%	36%	No data
MF Async HK TwoStep GEI n=8	86%	15%	35%	35%	26%	18%	50%	47%	44%	7%	100%
MF Async HK TwoStep GEI n=16	56%	19%	32%	50%	37%	27%	44%	44%	21%	0%	100%

Table D.2: Mean elapsed time ± std, coloured by success rate (values are tabulated in Table D.1) for the numerical test functions. Note the results for the Bohachevsky, Park91B, Hartmann6 and Borehole functions and the GEI based infill strategies can be skewed due to the lower number runs.

Algorithm	Forrester	Bohachevsky	Booth	Branin	Currin	Himmelblau	Six hump camelback	Park91a	Park91b	Hartmann6	Borehole
SF SP KRG Random	00:53:44 ± 00:19:48	08:33:52 ± 00:00:02	07:43:14 ± 00:59:21	05:44:32 ± 01:39:48	11:21:53 ± 01:57:40	10:11:16 ± 01:39:47	18:07:56 ± 01:17:24	25:11:07 ± 00:00:04	60:50:42 ± 06:20:09	95:16:04 ± 02:28:56	276:52:57 ± 00:00:10
MF SP HK Random	00:56:49 ± 00:22:03	14:14:55 ± 00:21:49	12:32:15 ± 02:04:51	04:16:27 ± 01:35:04	14:27:10 ± 02:39:03	14:36:19 ± 02:39:37	23:27:08 ± 02:35:44	33:11:56 ± 00:49:59	65:12:00 ± 10:23:20	106:46:22 ± 02:32:15	302:25:44 ± 06:53:40
SF SP KRG EI	00:13:49 ± 00:05:31	00:49:37 ± 00:34:26	00:21:40 ± 00:04:15	00:28:28 ± 00:09:36	01:18:24 ± 00:24:53	02:03:57 ± 00:39:56	03:16:16 ± 00:57:44	01:07:15 ± 00:18:43	03:10:30 ± 01:20:30	18:26:40 ± 03:25:07	18:56:03 ± 07:10:12
SF SP KRG GEI	00:12:01 ± 00:00:00	05:18:59 ± 03:02:15	00:35:54 ± 00:15:50	00:32:18 ± 00:09:34	01:19:55 ± 00:24:41	02:47:48 ± 00:39:20	03:43:56 ± 01:12:57	02:06:36 ± 00:41:21	05:29:41 ± 02:04:50	23:38:41 ± 06:01:39	32:10:50 ± 10:27:14
SF Async KRG EI n=8	00:05:58 ± 00:01:25	00:12:46 ± 00:04:12	00:08:41 ± 00:01:28	00:11:49 ± 00:03:27	00:35:10 ± 00:08:23	00:30:53 ± 00:04:23	00:54:42 ± 00:08:47	00:26:16 ± 00:04:51	01:19:01 ± 00:22:17	04:54:39 ± 00:36:49	05:45:50 ± 01:31:38
SF Async KRG EI n=16	00:05:33 ± 00:00:51	00:10:16 ± 00:02:56	00:08:41 ± 00:01:40	00:09:29 ± 00:01:54	00:25:55 ± 00:06:57	00:19:53 ± 00:04:33	00:37:05 ± 00:06:20	00:22:24 ± 00:04:12	01:05:45 ± 00:14:01	03:44:59 ± 00:47:42	04:50:55 ± 01:07:04
SF Async KRG GEI n=8	00:04:02 ± 00:00:00	00:29:14 ± 00:27:09	00:07:11 ± 00:01:31	00:10:21 ± 00:02:51	00:23:02 ± 00:09:28	00:29:05 ± 00:05:09	00:44:13 ± 00:14:04	00:22:58 ± 00:04:32	01:03:02 ± 00:15:26	04:03:19 ± 00:38:44	05:05:42 ± 01:53:20
SF Async KRG GEI n=16	00:04:02 ± 00:00:00	00:14:14 ± 00:11:13	00:06:41 ± 00:00:50	00:08:09 ± 00:01:17	00:15:40 ± 00:05:49	00:16:12 ± 00:03:23	00:31:48 ± 00:06:49	00:20:04 ± 00:00:01	00:55:09 ± 00:09:48	02:23:41 ± 00:39:10	04:14:36 ± 01:04:09
MF SP HK TwoStep EI	00:13:59 ± 00:03:34	00:40:00 ± 00:25:15	00:27:30 ± 00:06:35	00:27:39 ± 00:13:42	00:47:10 ± 00:26:52	01:54:04 ± 00:35:54	03:22:02 ± 01:03:55	00:54:03 ± 00:26:51	02:54:10 ± 01:33:36	18:32:23 ± 10:48:35	260:16:45 ± 57:59:34

Continued on next page

Table D.2 – continued from previous page

Algorithm	Forrester	Bohachevsky	Booth	Branin	Currin	Himmelblau	Six hump camelback	Park91a	Park91b	Hartmann6	Borehole
MF Async HK TwoStep EI n=8	00:06:05 ± 00:00:54	00:13:43 ± 00:04:04	00:09:46 ± 00:01:55	00:12:05 ± 00:03:26	00:24:12 ± 00:10:33	00:33:49 ± 00:10:17	01:00:24 ± 00:10:01	00:26:03 ± 00:08:03	01:12:07 ± 00:25:39	05:59:41 ± 02:00:06	No data
MF Async HK TwoStep EI n=16	00:05:12 ± 00:01:56	00:13:19 ± 00:05:31	00:09:11 ± 00:02:31	00:09:33 ± 00:03:00	00:19:37 ± 00:07:23	00:28:06 ± 00:11:11	00:42:20 ± 00:08:37	00:20:47 ± 00:03:43	00:58:50 ± 00:18:07	04:01:02 ± 01:28:10	No data
MF Async HK TwoStep GEI n=8	00:05:08 ± 00:01:43	00:16:23 ± 00:25:03	00:07:44 ± 00:02:22	00:08:35 ± 00:02:44	00:10:50 ± 00:07:36	00:18:16 ± 00:12:42	00:33:56 ± 00:20:53	00:16:02 ± 00:05:02	00:39:23 ± 00:16:42	03:13:54 ± 02:03:28	11:49:29 ± 00:26:06
MF Async HK TwoStep GEI n=16	00:03:13 ± 00:01:11	00:47:07 ± 02:33:29	00:05:38 ± 00:01:25	00:06:52 ± 00:02:11	00:08:20 ± 00:05:40	00:16:04 ± 00:14:29	00:26:08 ± 00:15:45	00:13:28 ± 00:03:36	00:30:51 ± 00:07:00	01:35:06 ± 00:56:18	06:47:03 ± 02:41:27

Table D.3: Mean elapsed time ± for successful runs of the numerical test functions, coloured by success rate (values are tabulated in Table D.1). Note the results for the Bohachevsky, Park91B, Hartmann6 and Borehole functions and the GEI based infill strategies can be skewed due to the lower number runs.

Algorithm	Forrester	Bohachevsky	Booth	Branin	Currin	Himmelblau	Six hump camelback	Park91a	Park91b	Hartmann6	Borehole
SF SP KRG Random	00:53:44 ± 00:19:48	08:33:52 ± 00:00:02	07:43:14 ± 00:59:21	05:44:32 ± 01:39:48	11:21:53 ± 01:57:40	10:25:13 ± 00:42:41	18:07:56 ± 01:17:24	25:11:07 ± 00:00:04	60:50:42 ± 06:20:09	No successful run	276:52:57 ± 00:00:07
MF SP HK Random	00:56:49 ± 00:22:03	14:16:53 ± 00:19:24	12:32:15 ± 02:04:51	04:16:27 ± 01:35:04	14:27:10 ± 02:39:03	14:36:19 ± 02:39:37	23:27:08 ± 02:35:44	33:11:56 ± 00:49:59	65:12:00 ± 10:23:20	No successful run	299:31:27 ± 06:51:02
SF SP KRG EI	00:13:49 ± 00:05:31	00:52:18 ± 00:33:26	00:21:32 ± 00:04:14	00:31:08 ± 00:07:23	01:19:29 ± 00:25:47	02:10:59 ± 00:41:48	03:33:38 ± 00:47:47	01:07:54 ± 00:18:28	03:23:45 ± 01:20:27	20:39:05 ± 02:26:05	19:42:45 ± 07:16:38
SF SP KRG GEI	00:12:01 ± 00:00:00	06:05:54 ± 02:31:03	00:36:34 ± 00:15:55	00:36:15 ± 00:05:47	01:23:15 ± 00:21:58	02:48:22 ± 00:39:38	04:11:24 ± 00:42:41	02:06:36 ± 00:41:21	05:29:41 ± 02:04:50	27:17:09 ± 05:17:42	32:22:40 ± 10:29:54
SF Async KRG EI n=8	00:05:58 ± 00:01:25	00:13:24 ± 00:04:43	00:08:39 ± 00:01:28	00:12:28 ± 00:03:15	00:35:10 ± 00:08:23	00:31:00 ± 00:03:55	00:54:42 ± 00:08:47	00:26:16 ± 00:04:51	01:19:01 ± 00:22:17	04:51:29 ± 00:17:55	05:08:07 ± 00:44:00
SF Async KRG EI n=16	00:05:33 ± 00:00:51	00:10:23 ± 00:03:21	00:08:45 ± 00:01:39	00:09:46 ± 00:01:53	00:26:56 ± 00:06:01	00:20:03 ± 00:03:53	00:37:20 ± 00:06:17	00:22:24 ± 00:04:12	01:05:45 ± 00:14:01	03:35:30 ± 00:43:53	04:56:21 ± 01:06:33
SF Async KRG GEI n=8	00:04:02 ± 00:00:00	00:34:29 ± 00:26:21	00:07:10 ± 00:01:30	00:12:09 ± 00:02:24	00:25:19 ± 00:08:45	00:29:03 ± 00:04:42	00:51:53 ± 00:11:52	00:23:10 ± 00:04:37	01:06:29 ± 00:16:12	No successful run	05:30:08 ± 01:57:35

Continued on next page

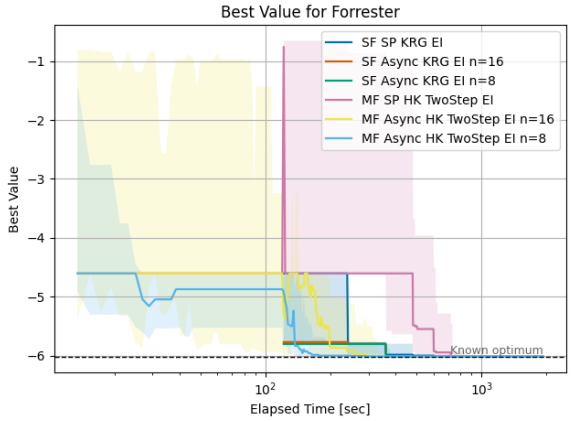
Table D.3 continued from previous page

Algorithm	Forrester	Bohachevsky	Booth	Branin	Currin	Himmelblau	Six hump camelback	Park91a	Park91b	Hartmann6	Borehole
SF Async KRG GEI n=16	00:04:02 ± 00:00:00	00:17:54 ± 00:12:13	00:06:41 ± 00:00:51	00:08:14 ± 00:01:22	00:17:18 ± 00:05:32	00:16:53 ± 00:02:39	00:33:33 ± 00:05:17	00:20:04 ± 00:00:01	00:56:13 ± 00:10:34	03:48:08 ± 00:00:00	04:20:15 ± 01:10:44
MF SP HK TwoStep EI	00:14:28 ± 00:02:52	00:40:07 ± 00:08:11	00:29:34 ± 00:04:28	00:39:15 ± 00:05:35	01:00:28 ± 00:22:28	02:10:19 ± 00:10:30	03:54:10 ± 00:44:13	00:59:58 ± 00:27:01	03:32:01 ± 01:23:55	22:32:56 ± 12:33:29	241:17:35 ± 68:35:11
MF Async HK TwoStep EI n=8	00:06:05 ± 00:00:54	00:15:24 ± 00:04:31	00:10:17 ± 00:01:15	00:13:32 ± 00:02:44	00:26:22 ± 00:09:38	00:36:37 ± 00:05:22	01:00:24 ± 00:10:01	00:27:03 ± 00:07:45	01:24:28 ± 00:22:47	06:24:12 ± 02:09:28	No data
MF Async HK TwoStep EI n=16	00:05:34 ± 00:01:45	00:13:48 ± 00:06:51	00:09:41 ± 00:02:16	00:10:55 ± 00:02:44	00:21:08 ± 00:06:26	00:32:42 ± 00:10:03	00:43:08 ± 00:06:56	00:20:50 ± 00:04:00	01:05:49 ± 00:13:23	04:52:33 ± 01:22:24	No data
MF Async HK TwoStep GEI n=8	00:05:37 ± 00:01:21	00:18:44 ± 00:16:21	00:09:47 ± 00:01:14	00:10:33 ± 00:02:23	00:18:47 ± 00:10:21	00:31:16 ± 00:06:33	00:52:07 ± 00:09:52	00:17:01 ± 00:05:24	00:41:43 ± 00:19:56	05:43:39 ± 00:00:00	11:49:29 ± 00:26:06
MF Async HK TwoStep GEI n=16	00:03:52 ± 00:01:13	02:35:09 ± 04:47:45	00:06:47 ± 00:00:22	00:08:15 ± 00:01:42	00:11:36 ± 00:07:59	00:36:02 ± 00:11:13	00:41:42 ± 00:07:24	00:13:00 ± 00:03:14	00:27:40 ± 00:00:55	No successful run	06:47:03 ± 02:41:27

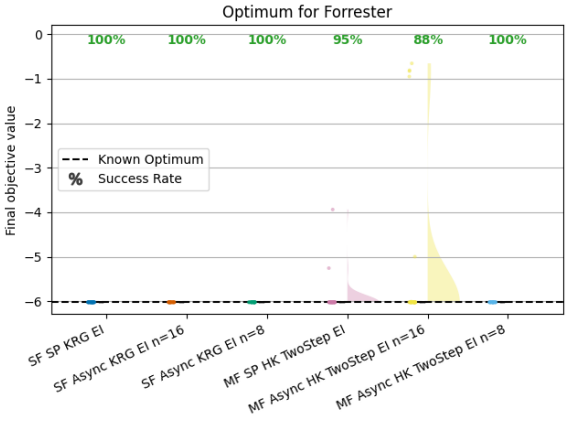
D.2. Result Visualisation

The convergence plot and raincloud plots for the resulting optimum, total function evaluations, function evaluations per rank, time elapsed and time elapsed per successful run are included in this section. The plots for each function are included in:

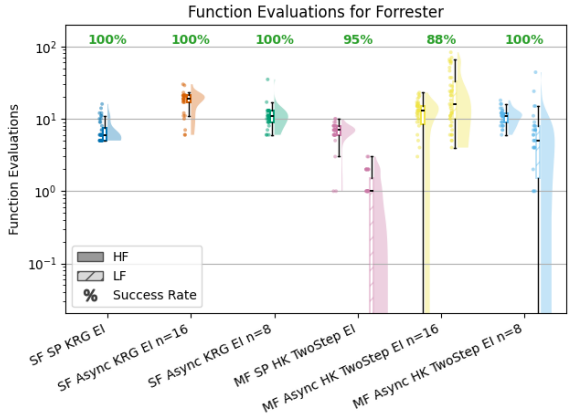
- ***Forrester***:
  - *EI based infill strategies*: Figure D.1
  - *GEI based infill strategies*: Figure D.2
- ***Bohachevsky***:
  - *EI based infill strategies*: Figure D.3
  - *GEI based infill strategies*: Figure D.4
- ***Booth***:
  - *EI based infill strategies*: Figure D.5
  - *GEI based infill strategies*: Figure D.6
- ***Branin***:
  - *EI based infill strategies*: Figure D.7
  - *GEI based infill strategies*: Figure D.8
- ***Currin (maximisation)***:
  - *EI based infill strategies*: Figure D.9
  - *GEI based infill strategies*: Figure D.10
- ***Himmelblau***:
  - *EI based infill strategies*: Figure D.11
  - *GEI based infill strategies*: Figure D.12
- ***Six Hump Camelback***:
  - *EI based infill strategies*: Figure D.13
  - *GEI based infill strategies*: Figure D.14
- ***Park91A***:
  - *EI based infill strategies*: Figure D.15
  - *GEI based infill strategies*: Figure D.16
- ***Park91B***:
  - *EI based infill strategies*: Figure D.17
  - *GEI based infill strategies*: Figure D.18
- ***Hartmann6***:
  - *EI based infill strategies*: Figure D.19
  - *GEI based infill strategies*: Figure D.20
- ***Borehole***:
  - *EI based infill strategies*: Figure D.21
  - *GEI based infill strategies*: Figure D.22



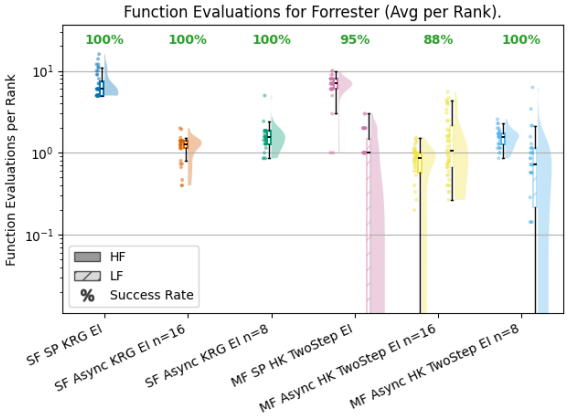
(a) The convergence plot.



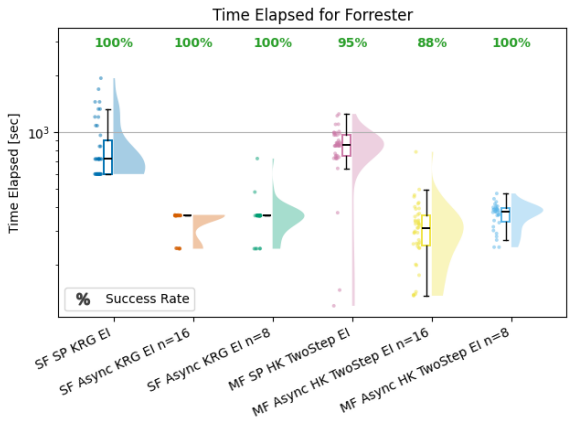
(b) Raincloud plot of the returned optimum.



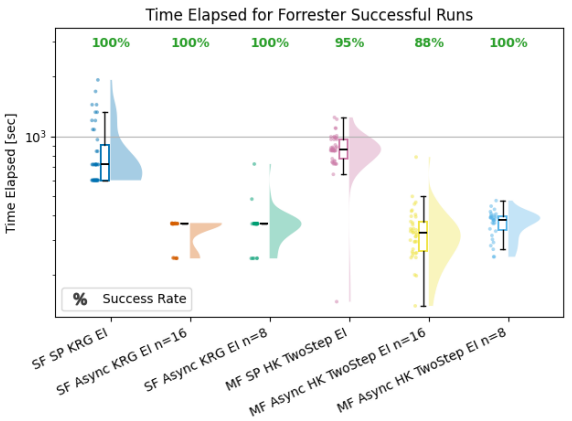
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

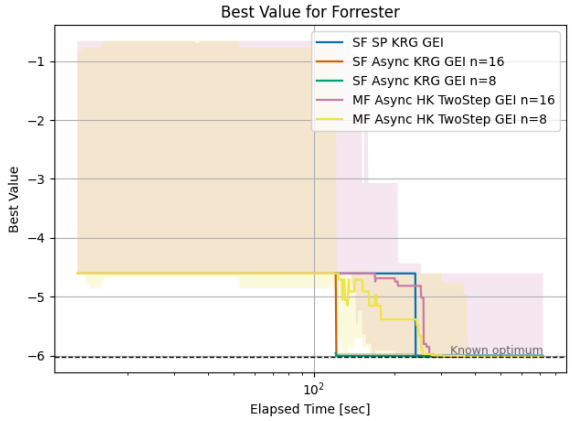


(e) Raincloud plot of the total time elapsed.

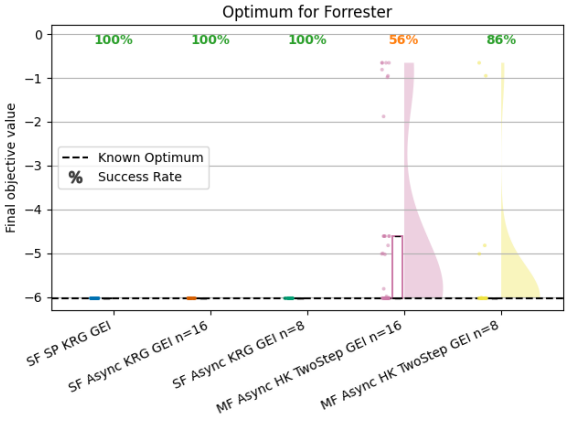


(f) Raincloud plot of the time elapsed in successful runs

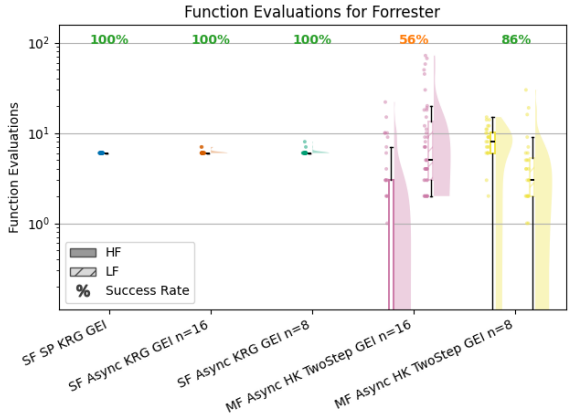
Figure D.1: The resulting plots for the Forrester function using EI based infill strategies.



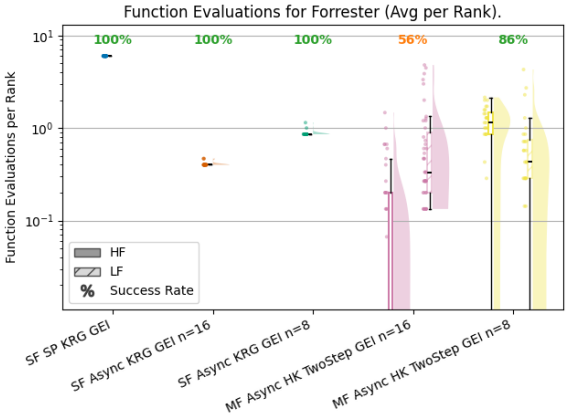
(a) The convergence plot.



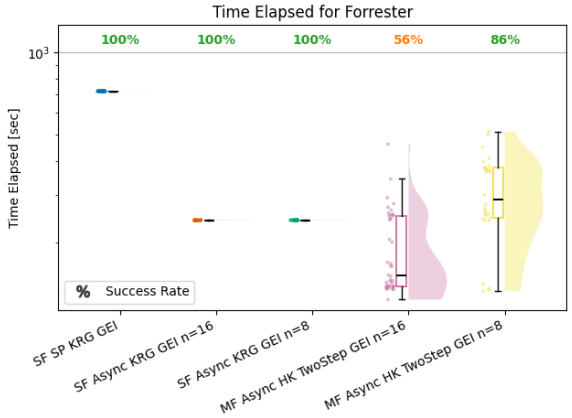
(b) Raincloud plot of the returned optimum.



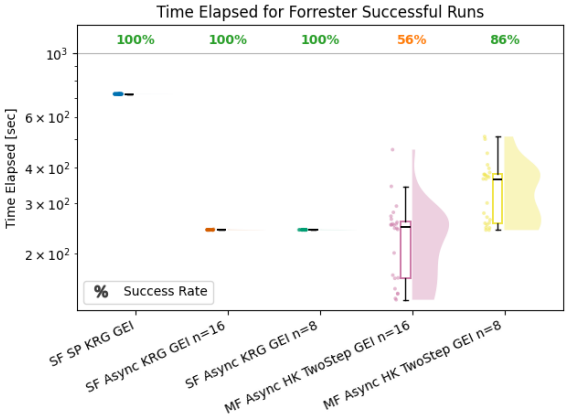
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

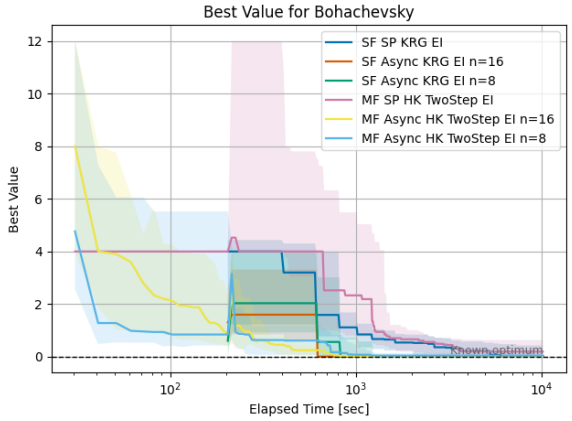


(e) Raincloud plot of the total time elapsed.

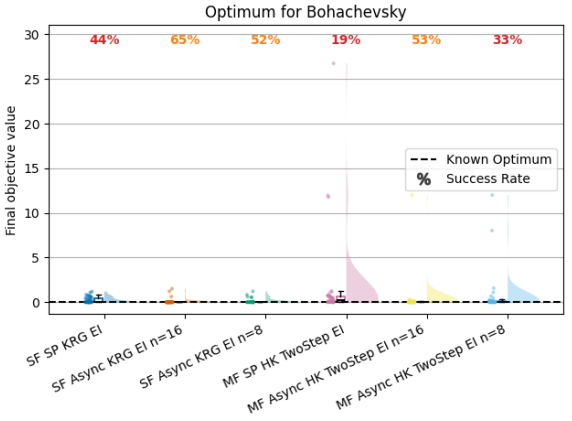


(f) Raincloud plot of the time elapsed in successful runs

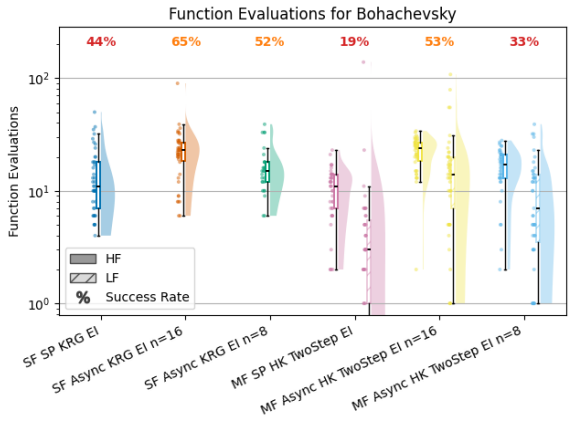
Figure D.2: The resulting plots for the Forrester function using GEI based infill strategies.



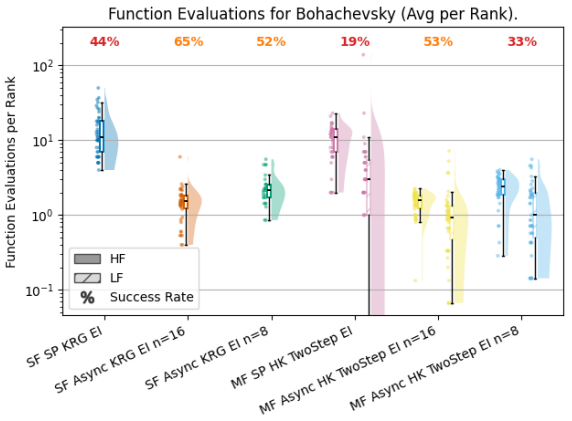
(a) The convergence plot.



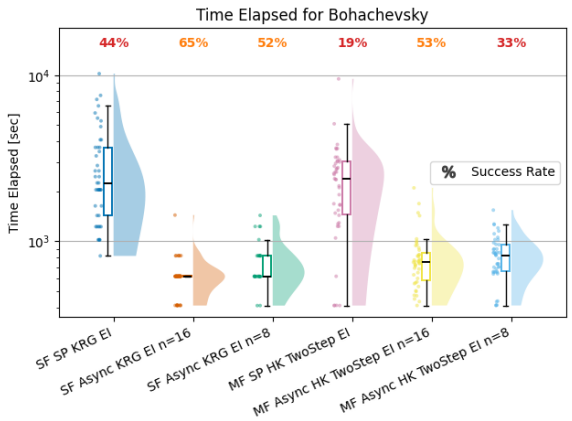
(b) Raincloud plot of the returned optimum.



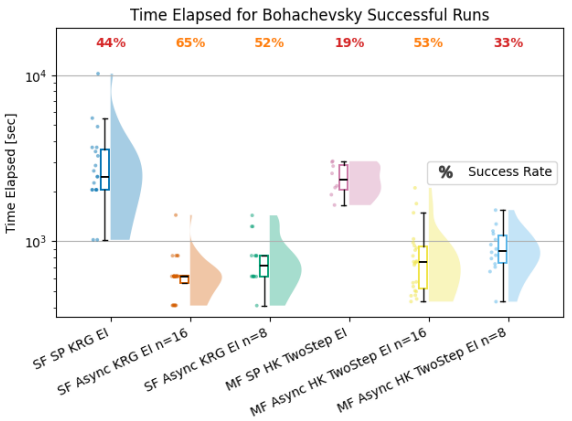
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

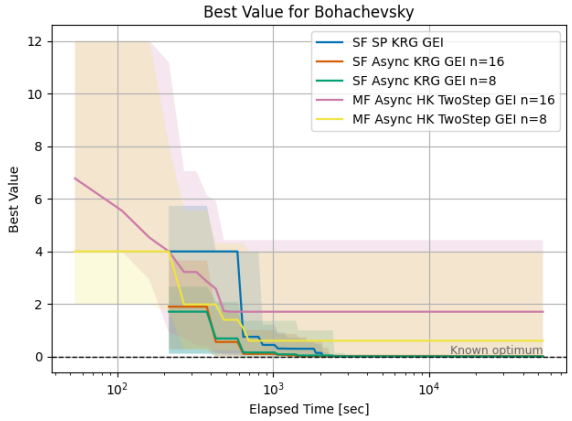


(e) Raincloud plot of the total time elapsed.

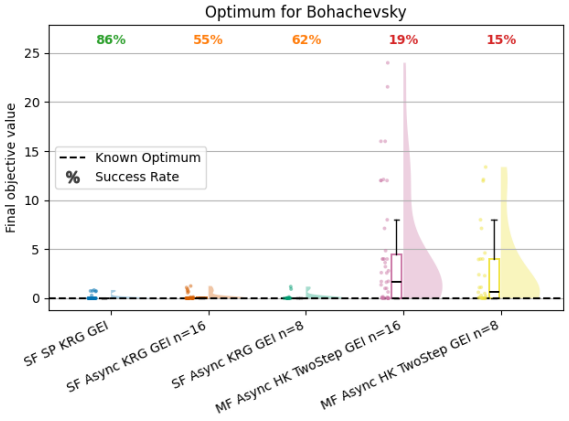


(f) Raincloud plot of the time elapsed in successful runs

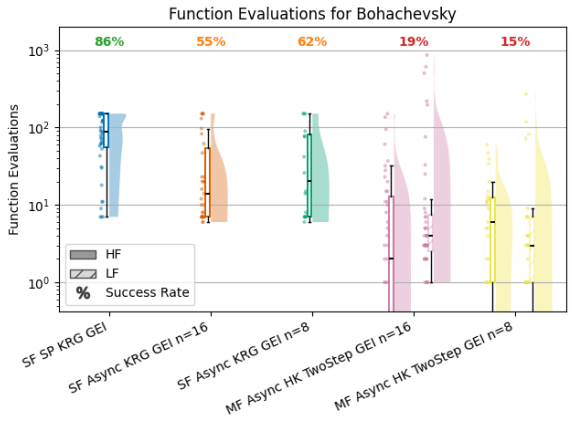
Figure D.3: The resulting plots for the Bohachevsky function using EI based infill strategies.



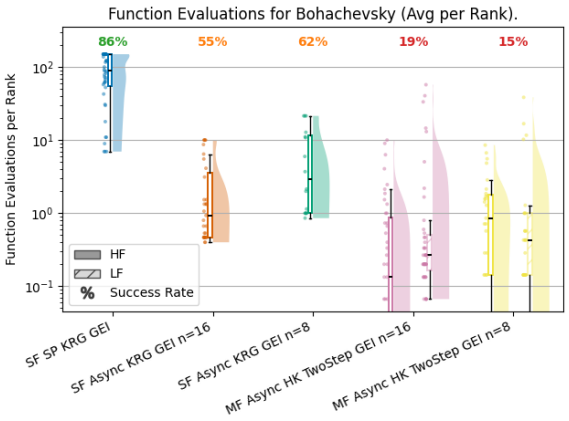
(a) The convergence plot.



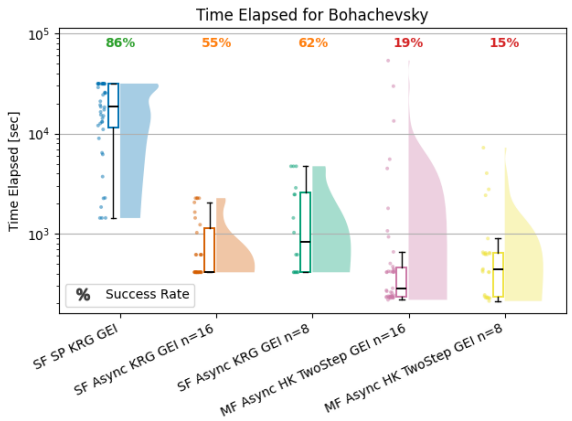
(b) Raincloud plot of the returned optimum.



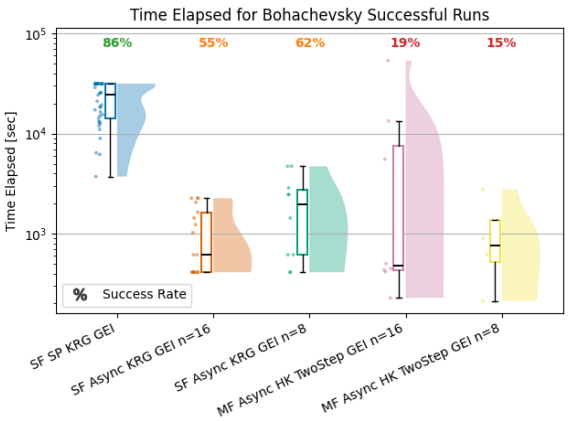
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

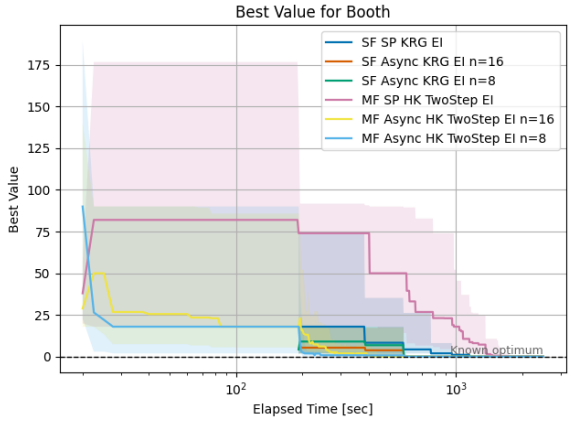


(e) Raincloud plot of the total time elapsed.

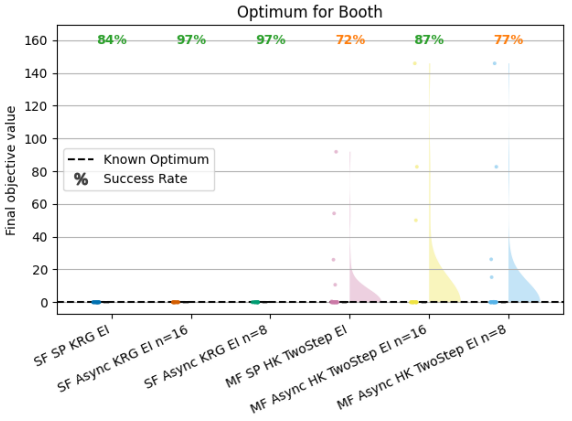


(f) Raincloud plot of the time elapsed in successful runs

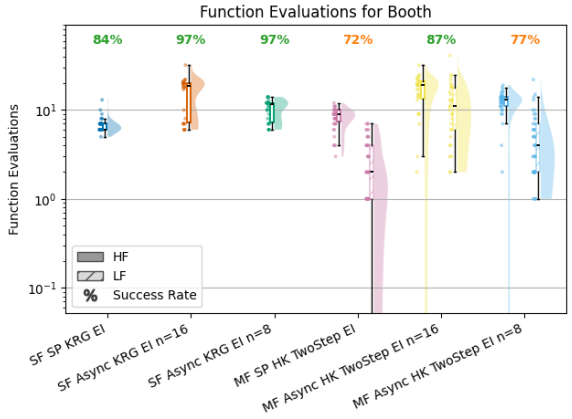
Figure D.4: The resulting plots for the Bohachevsky function using GEI based infill strategies.



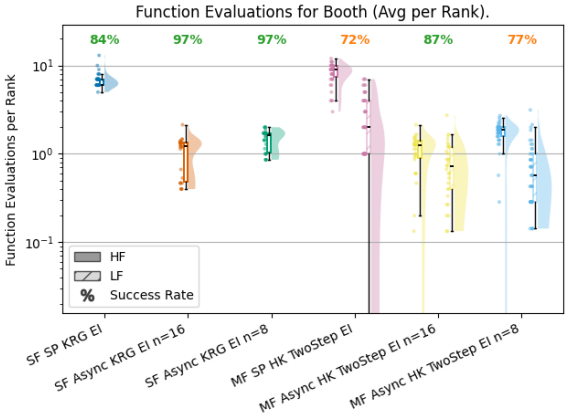
(a) The convergence plot.



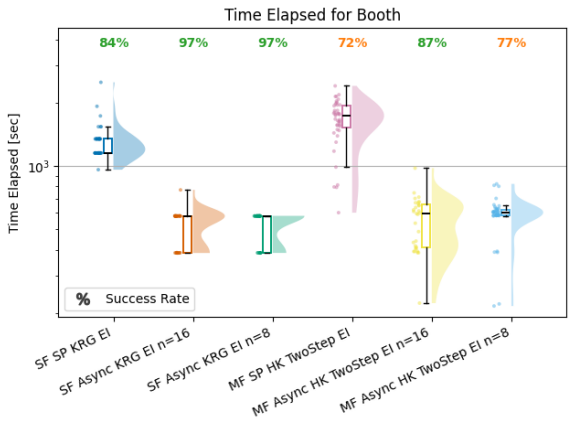
(b) Raincloud plot of the returned optimum.



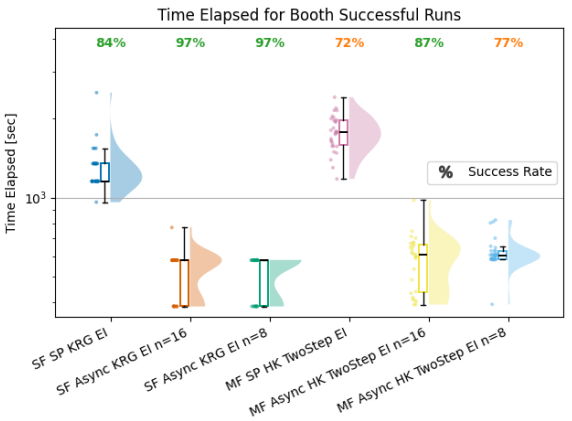
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

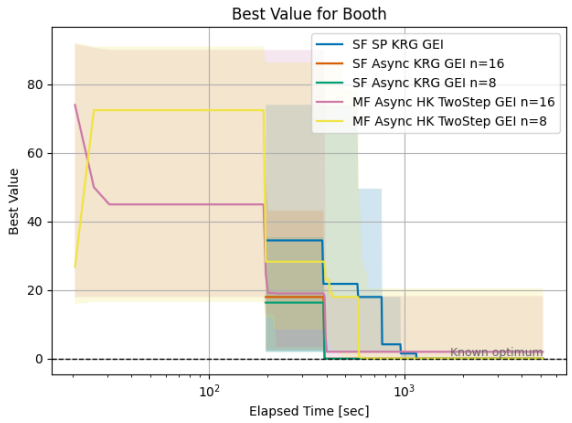


(e) Raincloud plot of the total time elapsed.

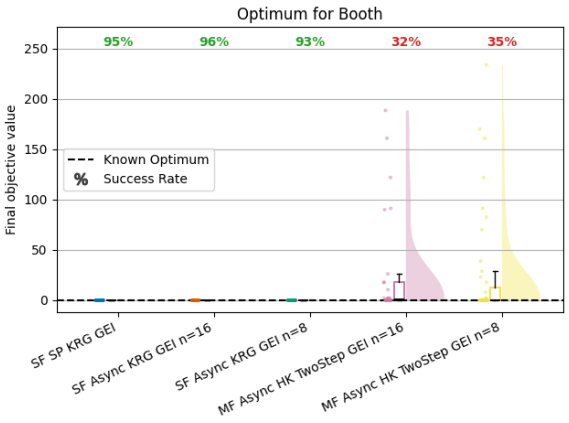


(f) Raincloud plot of the time elapsed in successful runs

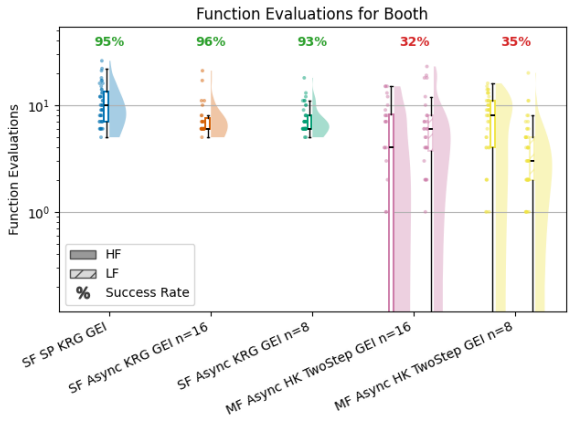
Figure D.5: The resulting plots for the Booth function using EI based infill strategies.



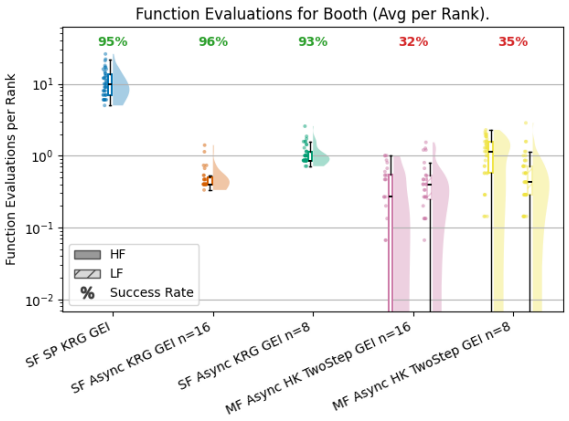
(a) The convergence plot.



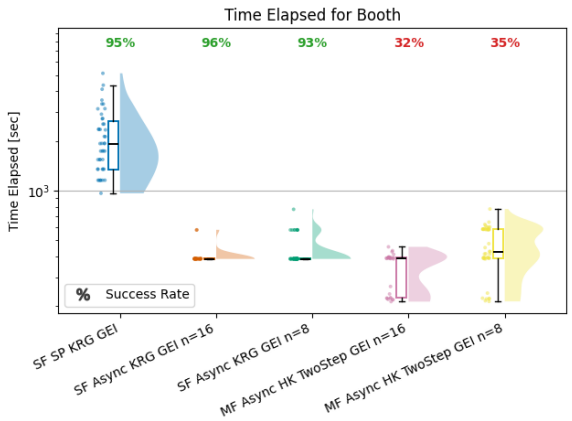
(b) Raincloud plot of the returned optimum.



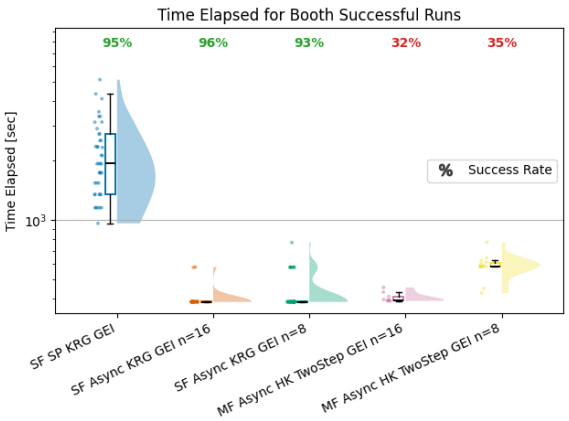
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

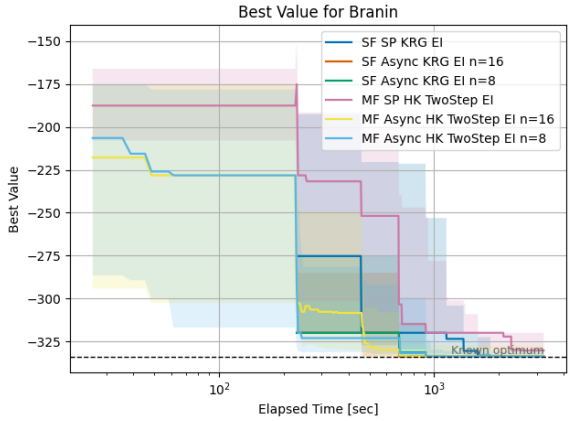


(e) Raincloud plot of the total time elapsed.

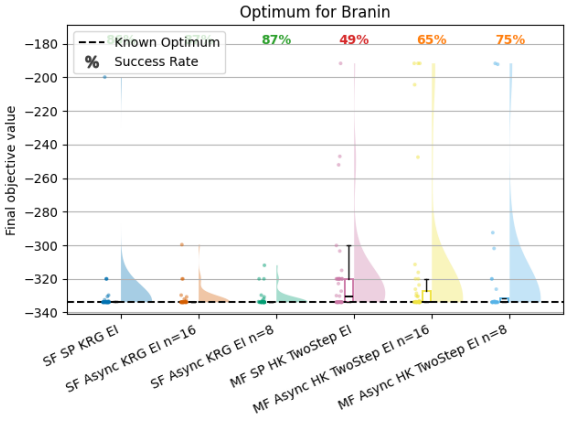


(f) Raincloud plot of the time elapsed in successful runs

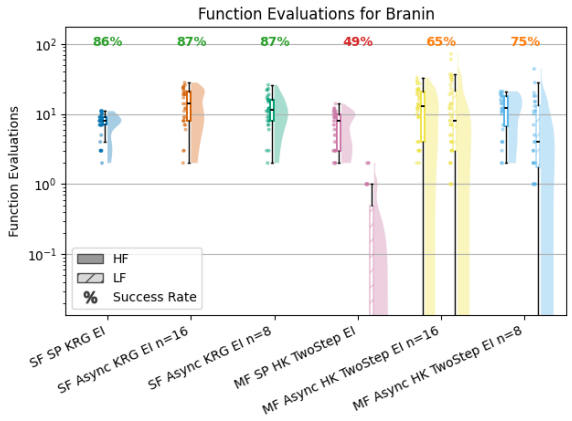
Figure D.6: The resulting plots for the Booth function using GEI based infill strategies.



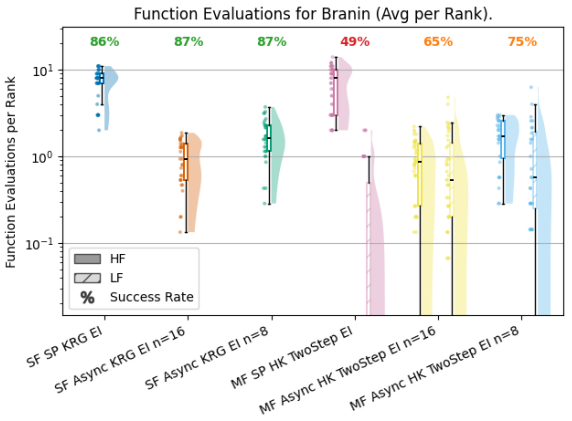
(a) The convergence plot.



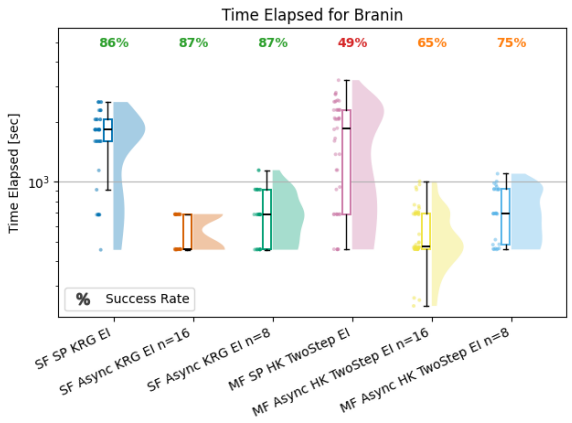
(b) Raincloud plot of the returned optimum.



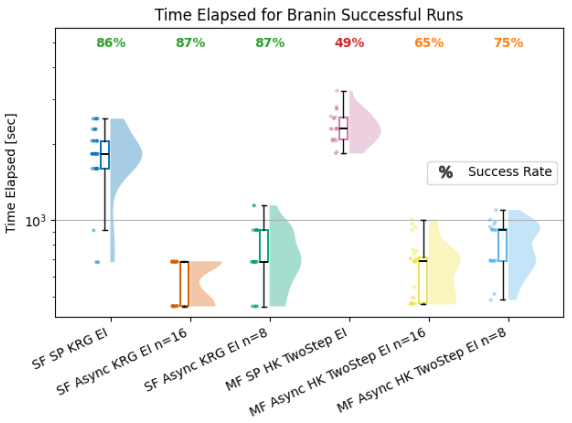
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

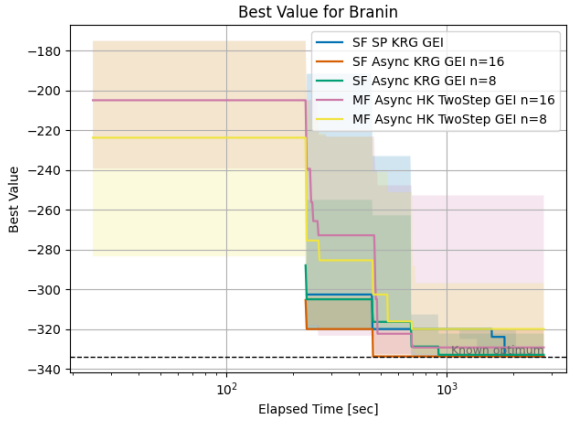


(e) Raincloud plot of the total time elapsed.

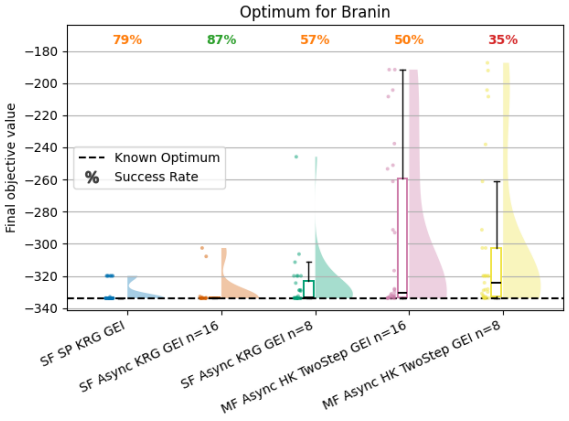


(f) Raincloud plot of the time elapsed in successful runs

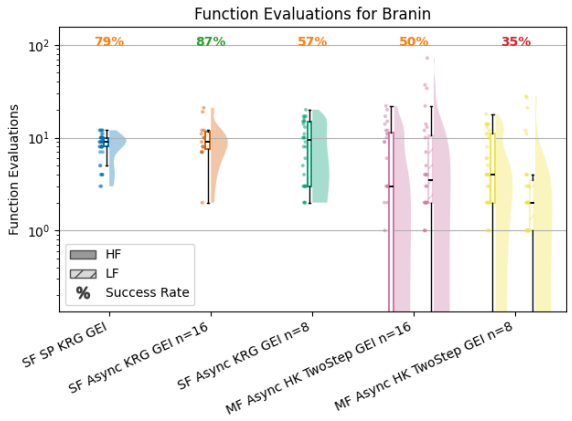
Figure D.7: The resulting plots for the Branin function using EI based infill strategies.



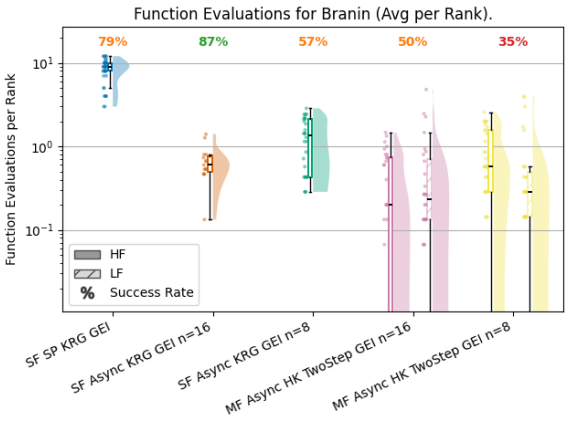
(a) The convergence plot.



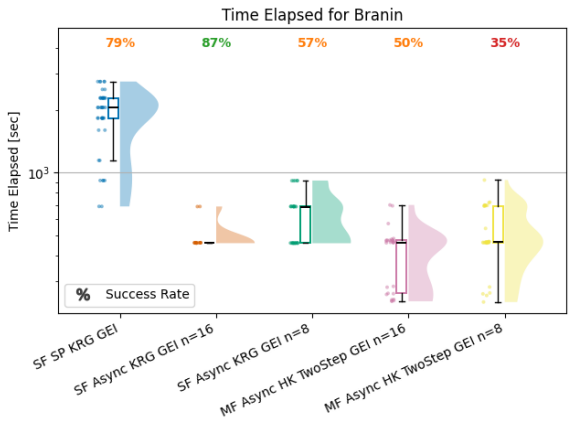
(b) Raincloud plot of the returned optimum.



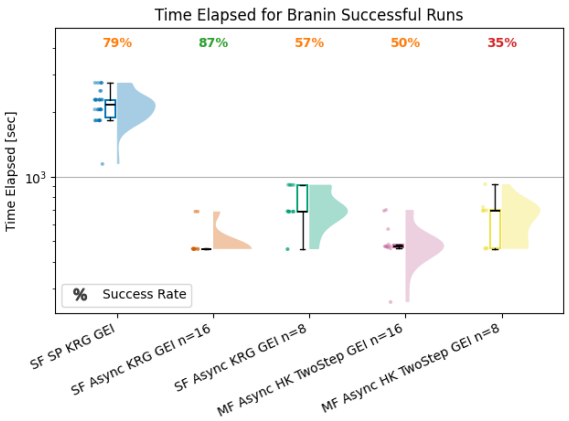
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

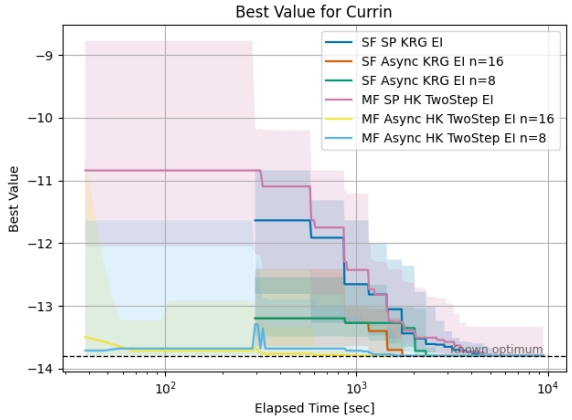


(e) Raincloud plot of the total time elapsed.

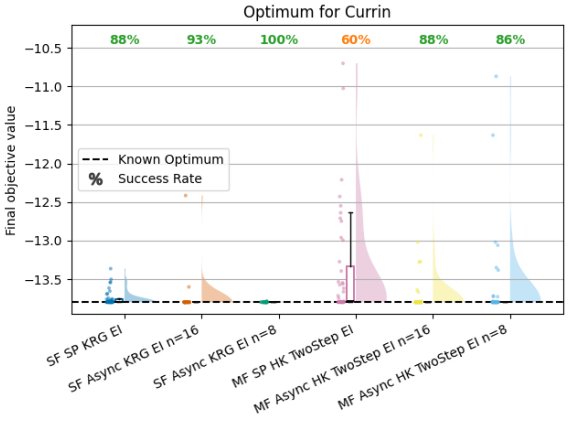


(f) Raincloud plot of the time elapsed in successful runs

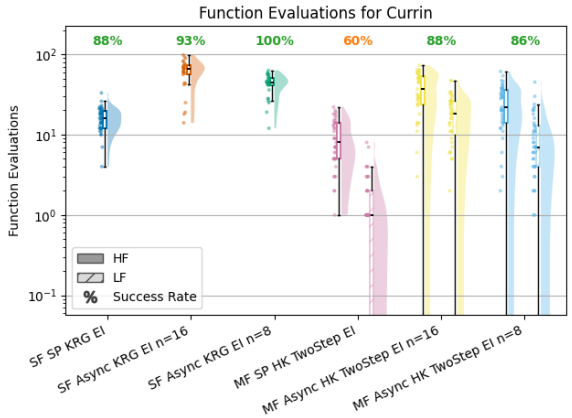
Figure D.8: The resulting plots for the Branin function using GEI based infill strategies.



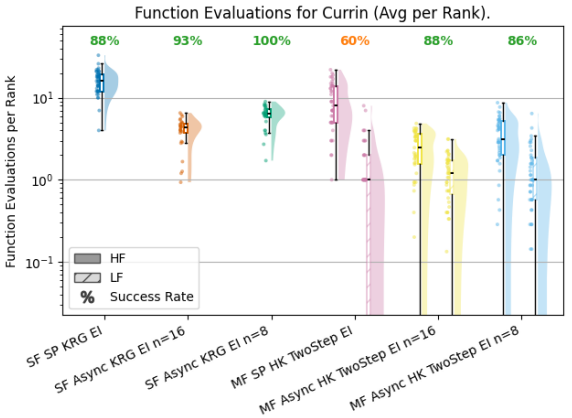
(a) The convergence plot.



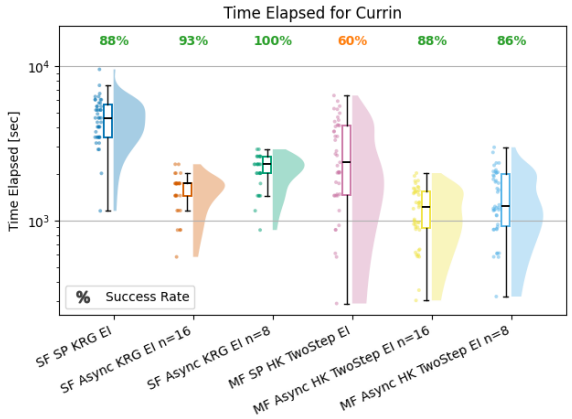
(b) Raincloud plot of the returned optimum (note the returned optimum is -1 times the actual value to convert the maximisation problem into a minimisation problem.



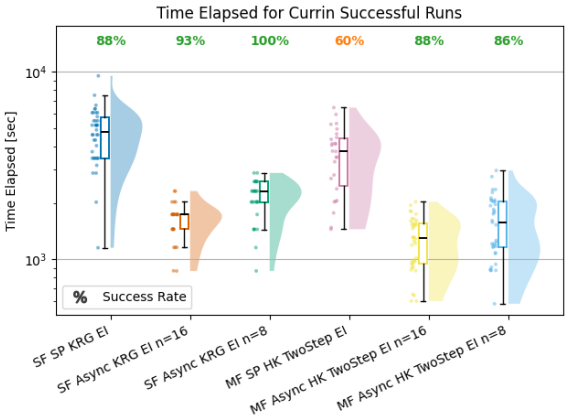
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

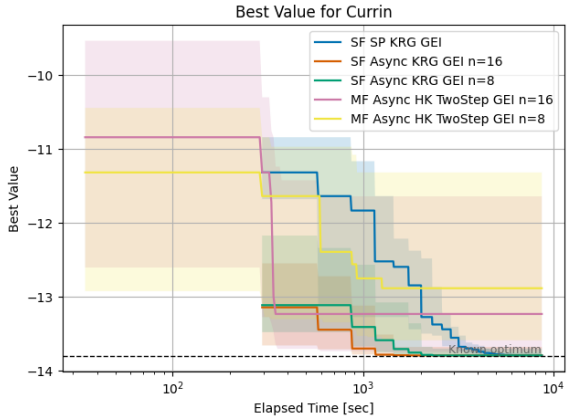


(e) Raincloud plot of the total time elapsed.

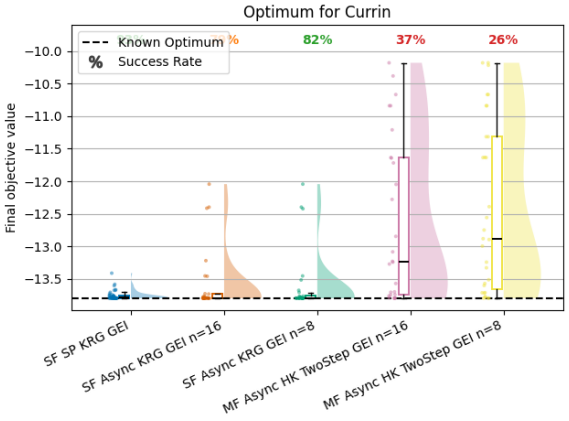


(f) Raincloud plot of the time elapsed in successful runs

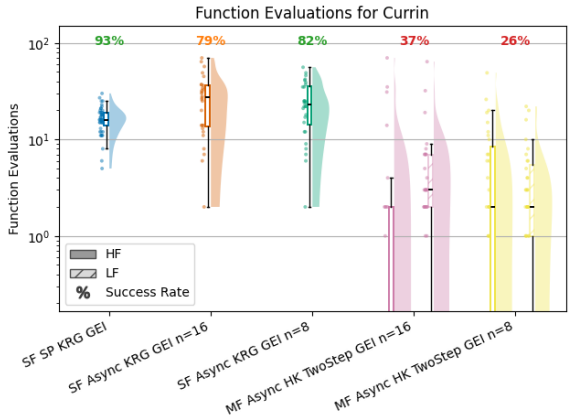
Figure D.9: The resulting plots for the Currin function using EI based infill strategies.



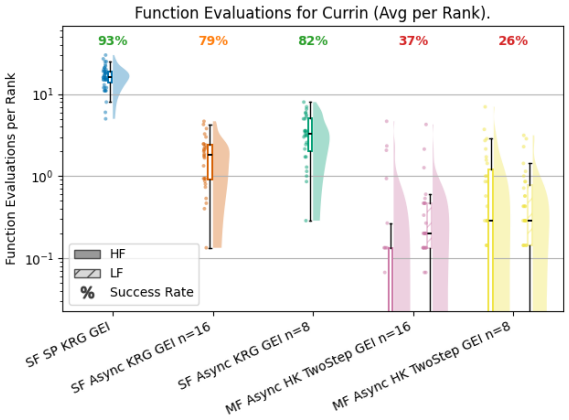
(a) The convergence plot.



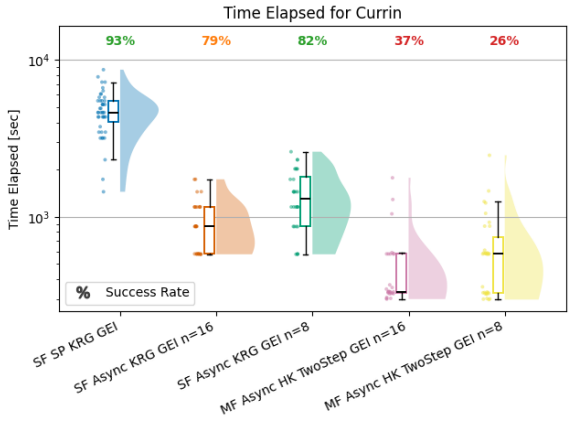
(b) Raincloud plot of the returned optimum (note the returned optimum is -1 times the actual value to convert the maximisation problem into a minimisation problem.



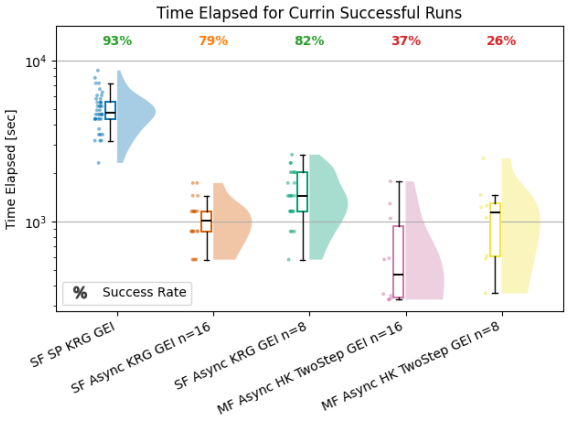
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

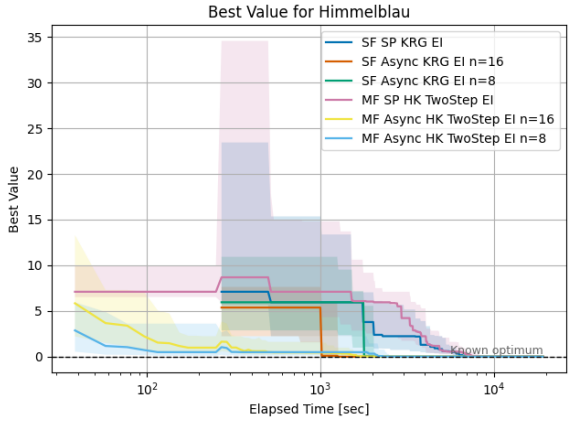


(e) Raincloud plot of the total time elapsed.

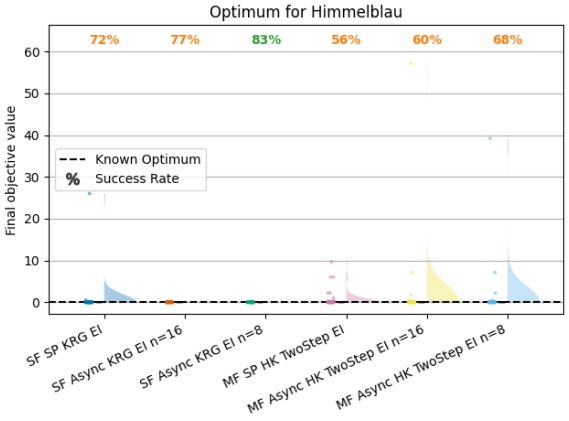


(f) Raincloud plot of the time elapsed in successful runs

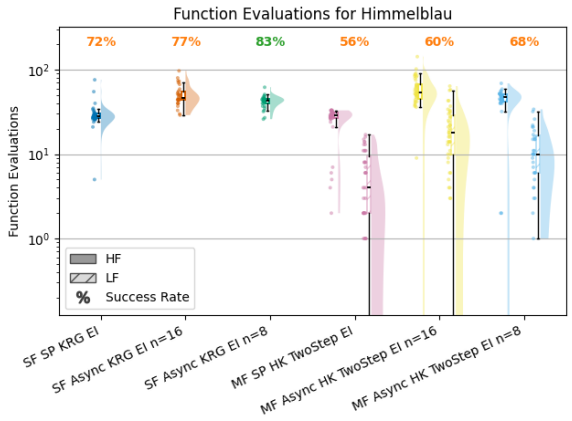
Figure D.10: The resulting plots for the Currin function using GEI based infill strategies.



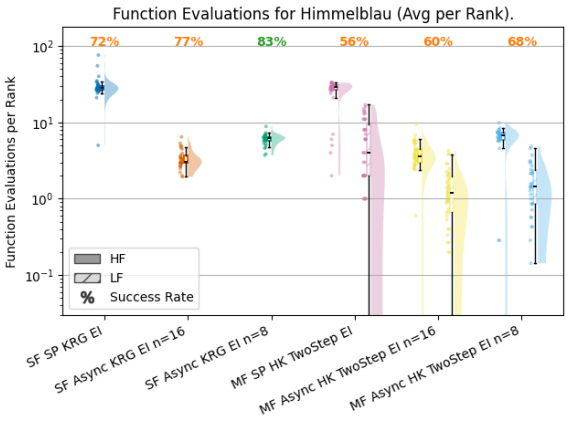
(a) The convergence plot.



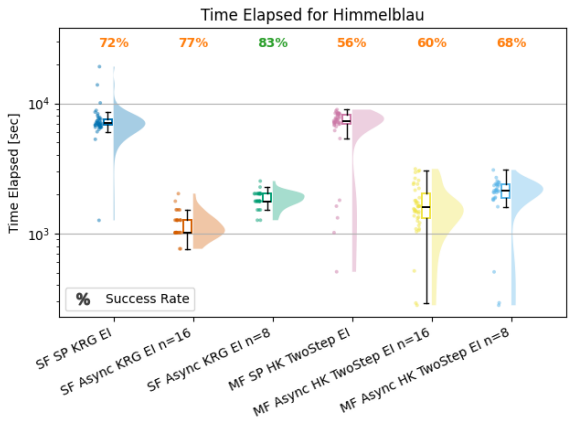
(b) Raincloud plot of the returned optimum.



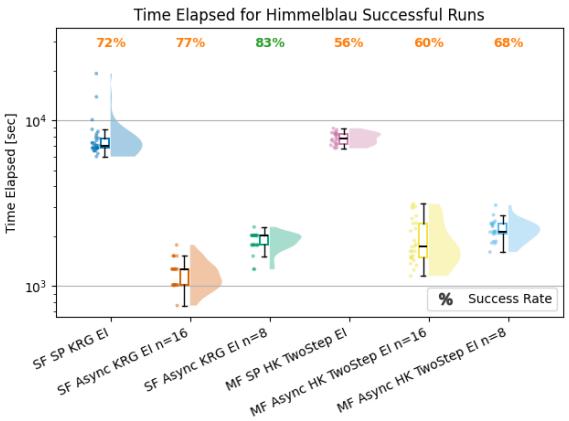
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

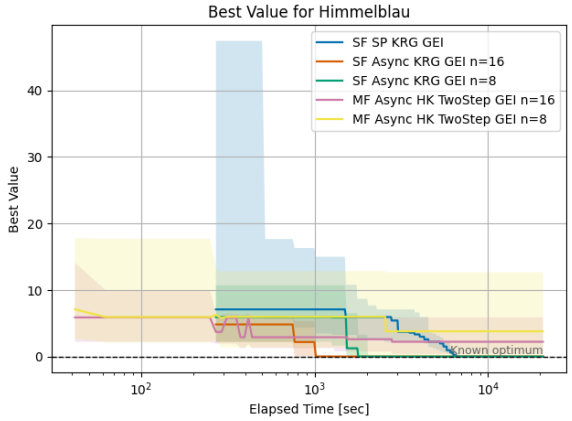


(e) Raincloud plot of the total time elapsed.

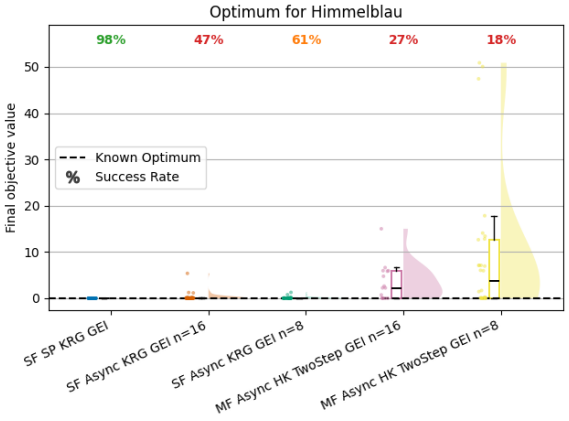


(f) Raincloud plot of the time elapsed in successful runs

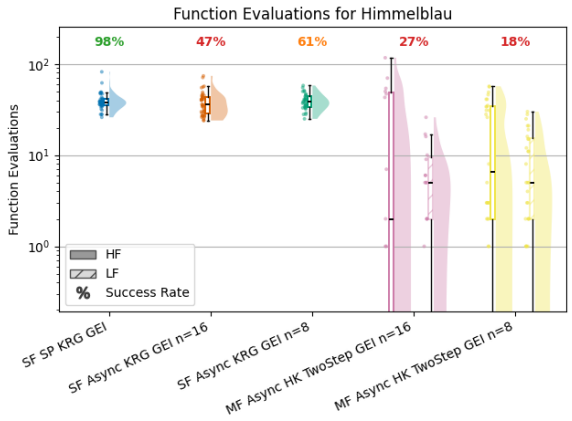
Figure D.11: The resulting plots for the Himmelblau function using EI based infill strategies.



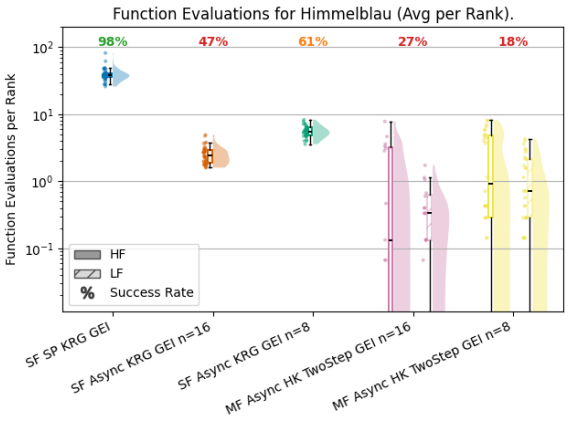
(a) The convergence plot.



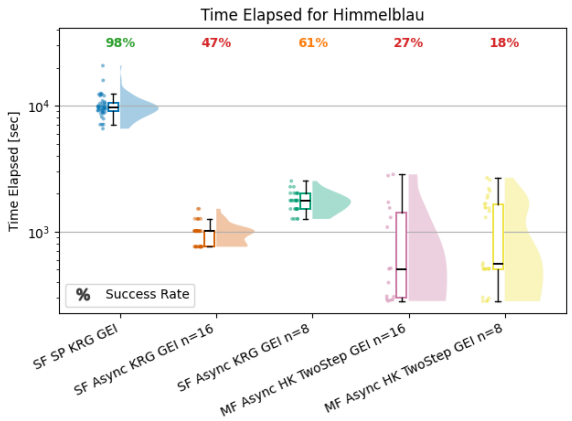
(b) Raincloud plot of the returned optimum.



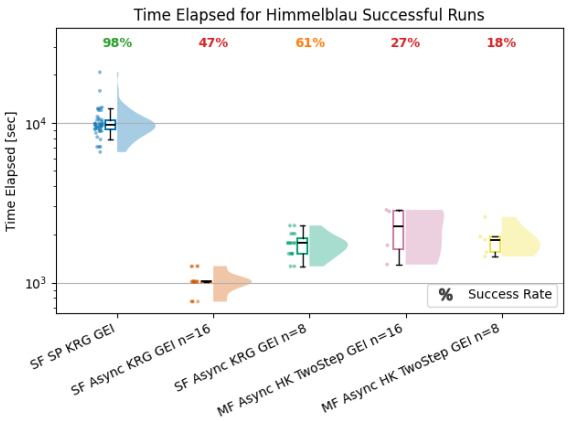
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

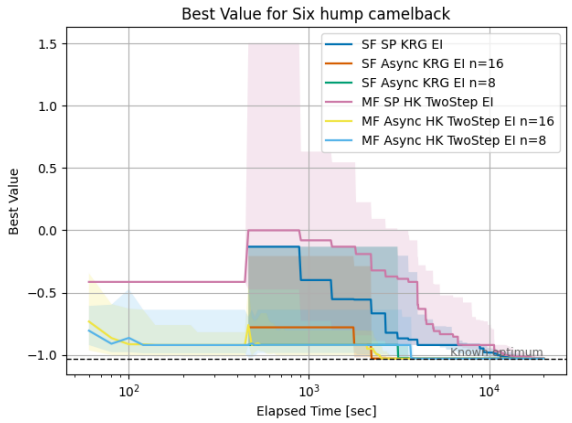


(e) Raincloud plot of the total time elapsed.

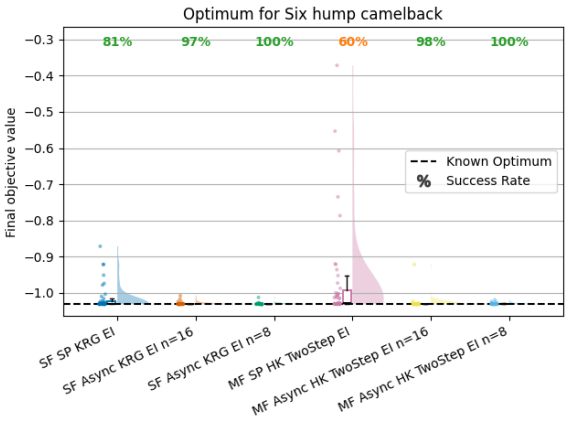


(f) Raincloud plot of the time elapsed in successful runs

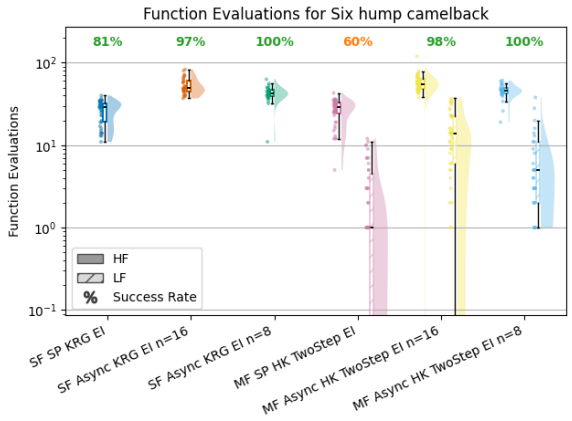
Figure D.12: The resulting plots for the Himmelblau function using GEI based infill strategies.



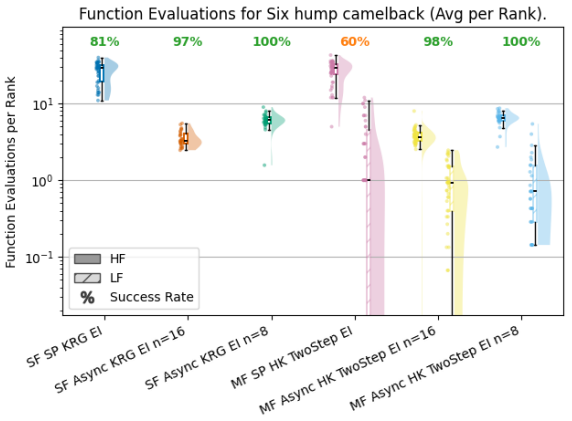
(a) The convergence plot.



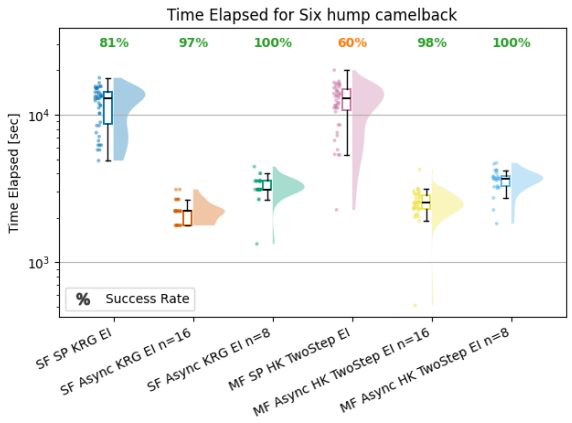
(b) Raincloud plot of the returned optimum.



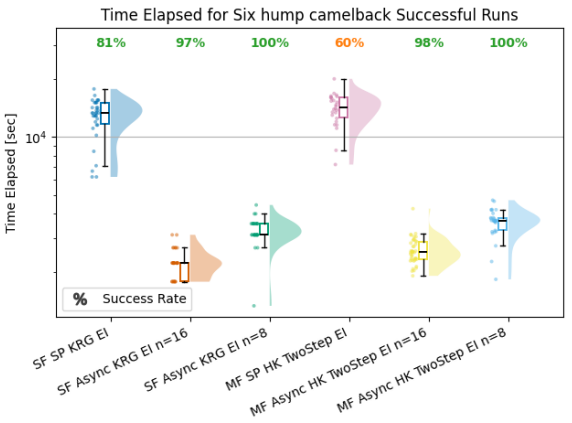
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

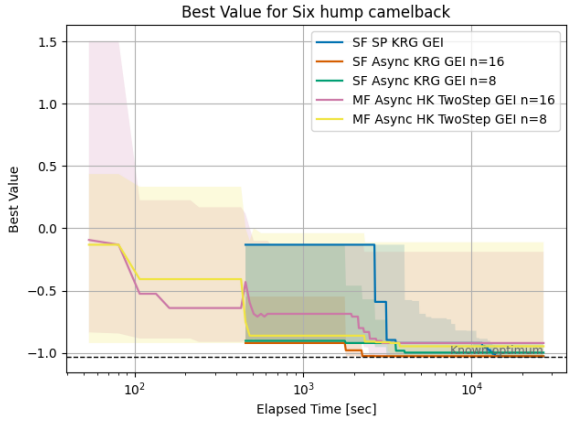


(e) Raincloud plot of the total time elapsed.

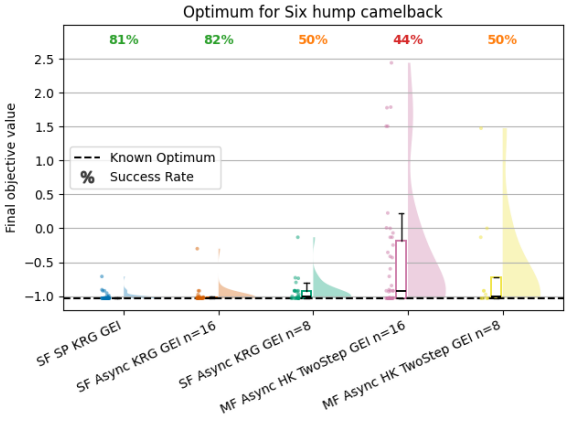


(f) Raincloud plot of the time elapsed in successful runs

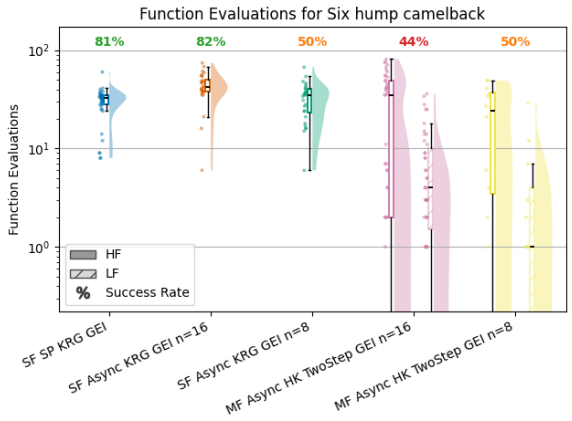
Figure D.13: The resulting plots for the Six Hump Camelback function using EI based infill strategies.



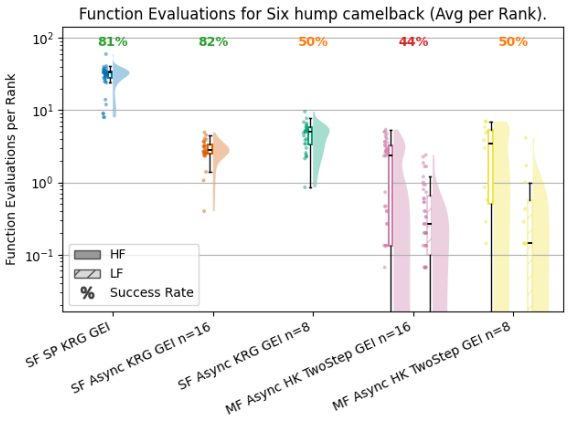
(a) The convergence plot.



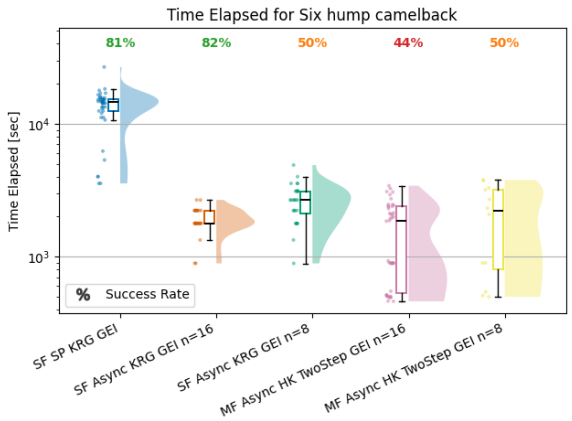
(b) Raincloud plot of the returned optimum.



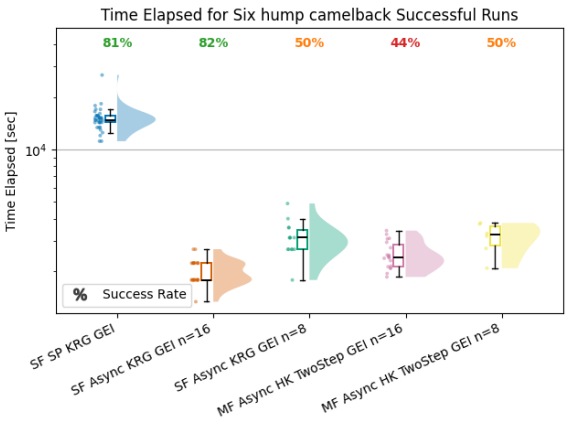
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

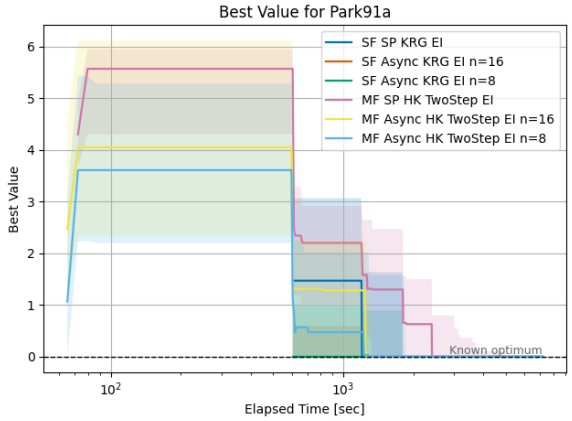


(e) Raincloud plot of the total time elapsed.

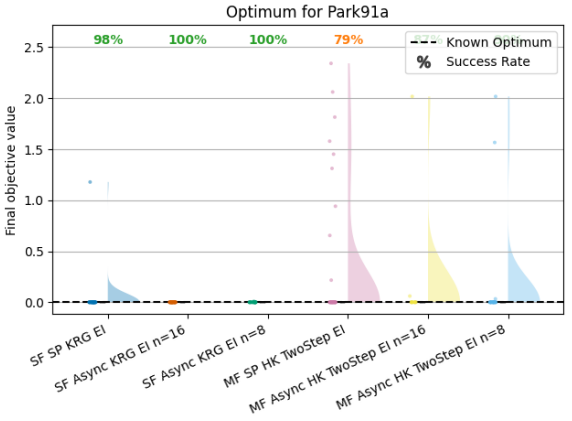


(f) Raincloud plot of the time elapsed in successful runs

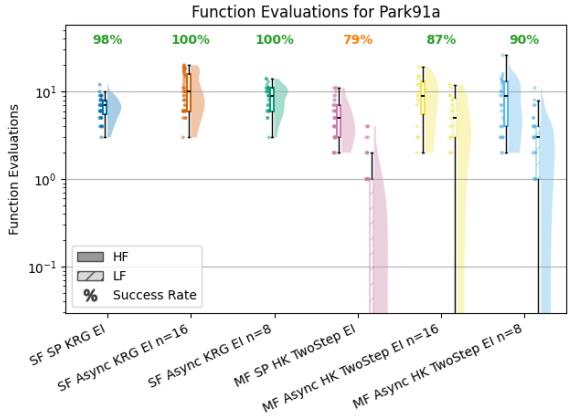
Figure D.14: The resulting plots for the Six Hump Camelback function using GEI based infill strategies.



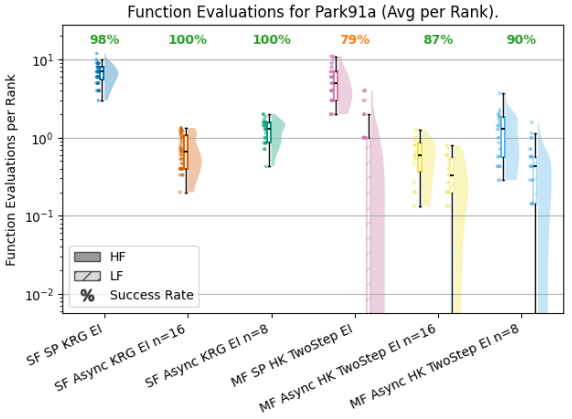
(a) The convergence plot.



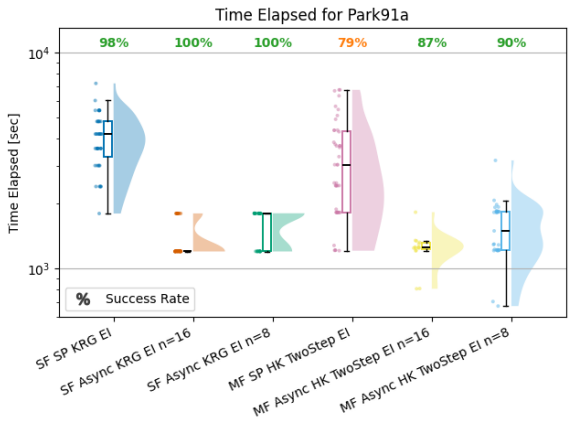
(b) Raincloud plot of the returned optimum.



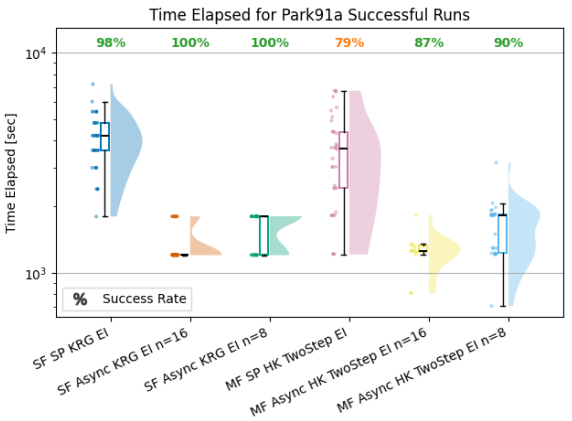
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

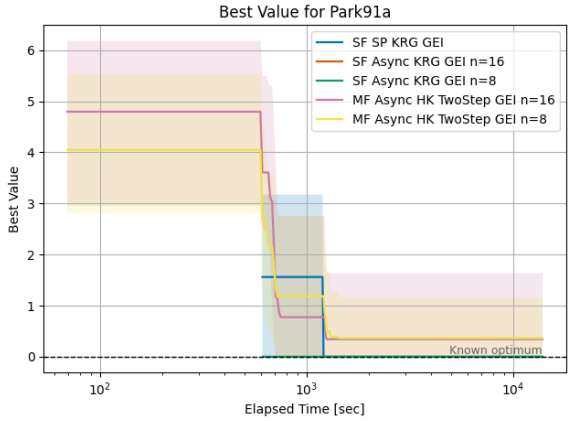


(e) Raincloud plot of the total time elapsed.

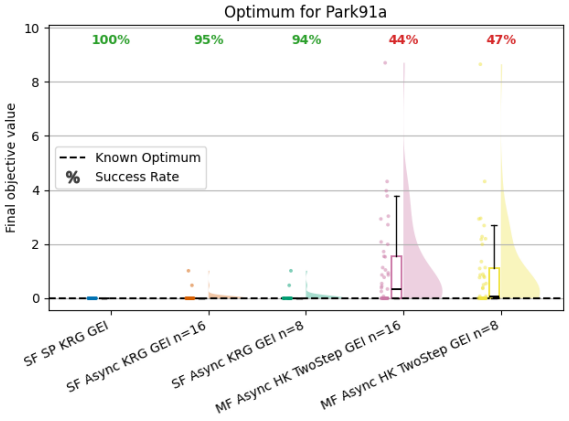


(f) Raincloud plot of the time elapsed in successful runs

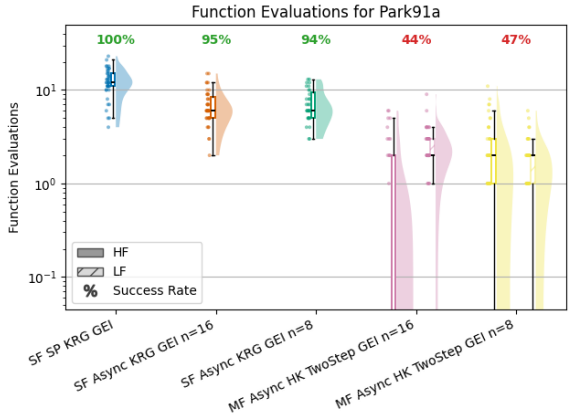
Figure D.15: The resulting plots for the Park91A function using EI based infill strategies.



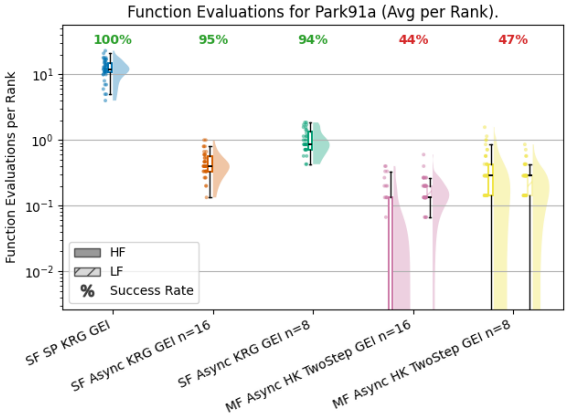
(a) The convergence plot.



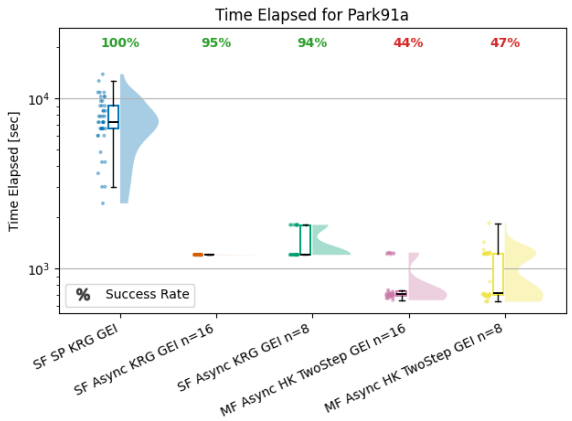
(b) Raincloud plot of the returned optimum.



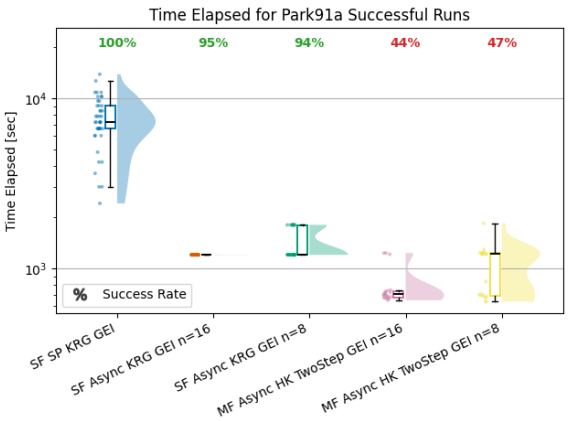
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

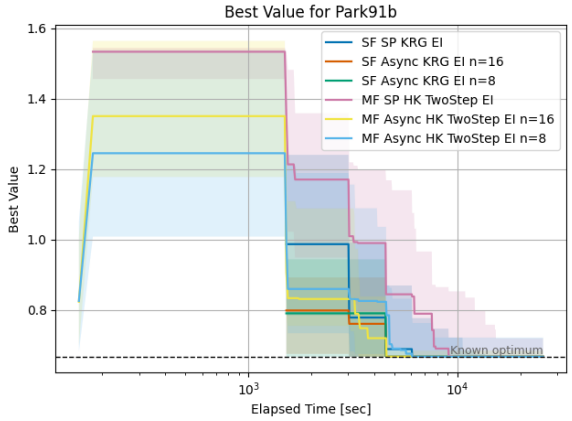


(e) Raincloud plot of the total time elapsed.

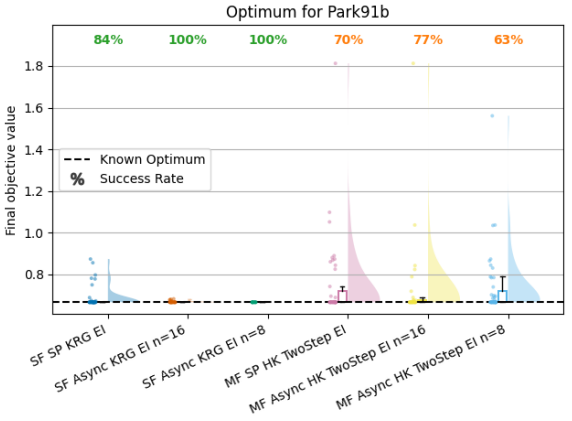


(f) Raincloud plot of the time elapsed in successful runs

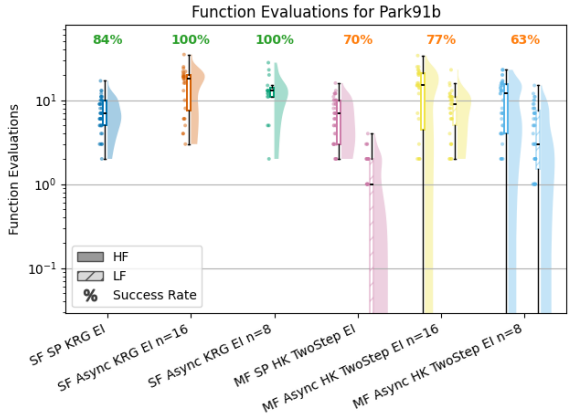
Figure D.16: The resulting plots for the Park91A function using GEI based infill strategies.



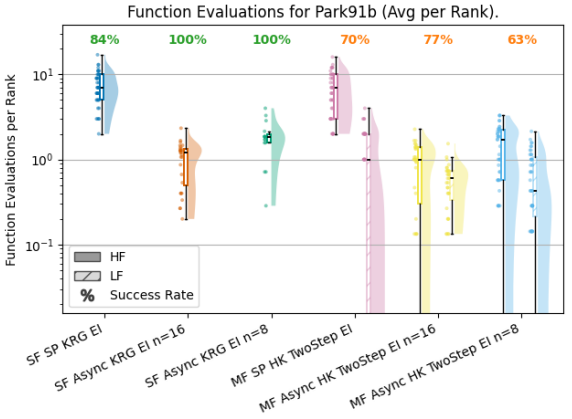
(a) The convergence plot.



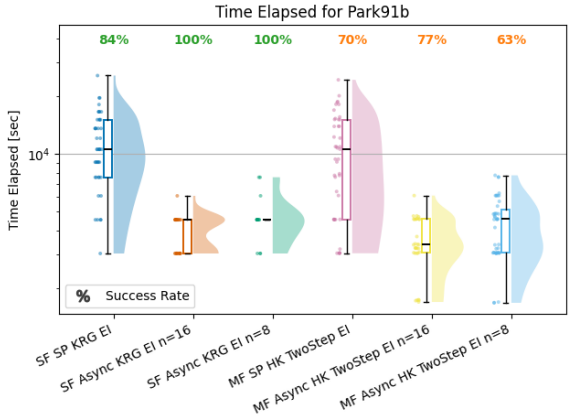
(b) Raincloud plot of the returned optimum.



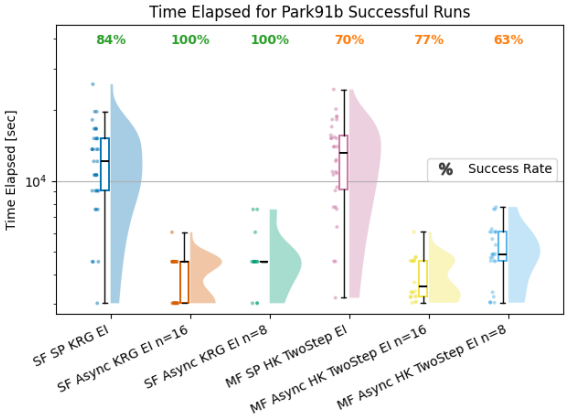
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

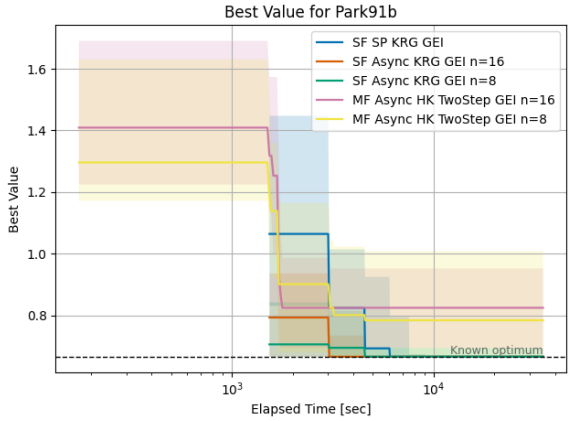


(e) Raincloud plot of the total time elapsed.

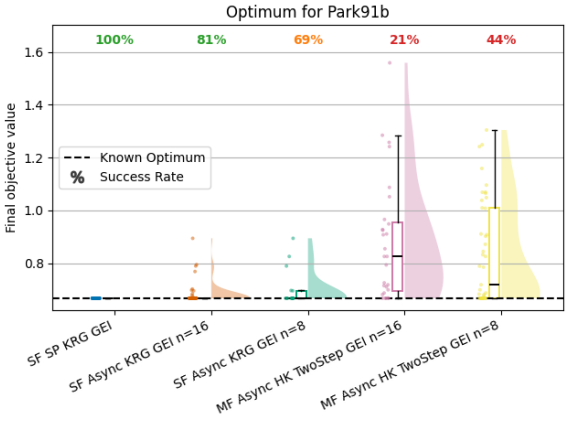


(f) Raincloud plot of the time elapsed in successful runs

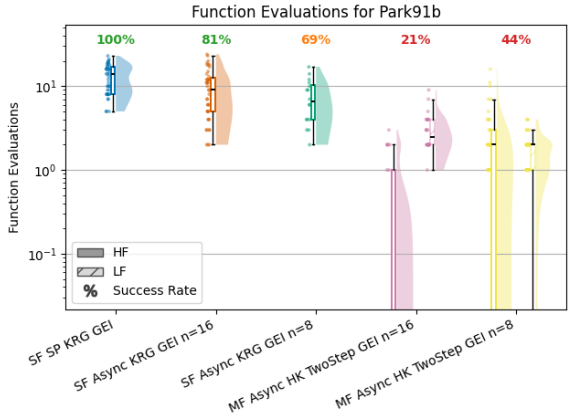
Figure D.17: The resulting plots for the Park91B function using EI based infill strategies.



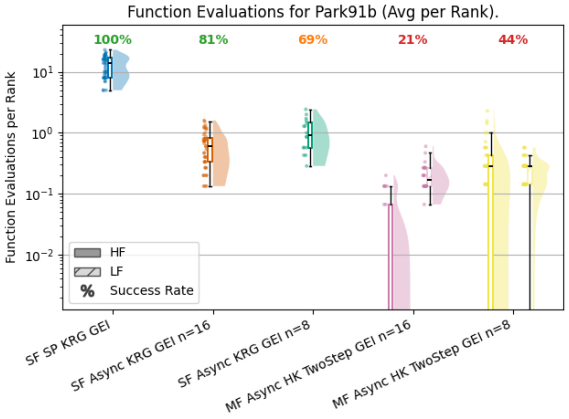
(a) The convergence plot.



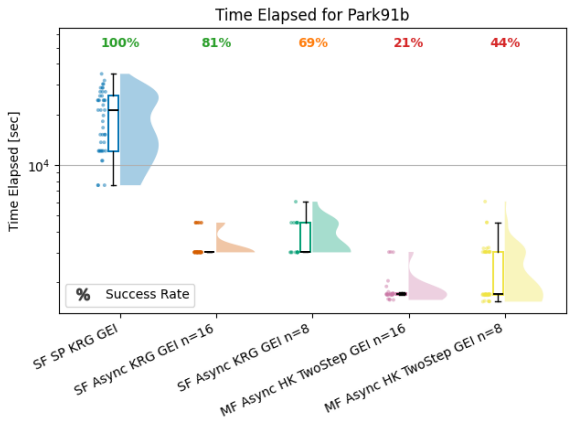
(b) Raincloud plot of the returned optimum.



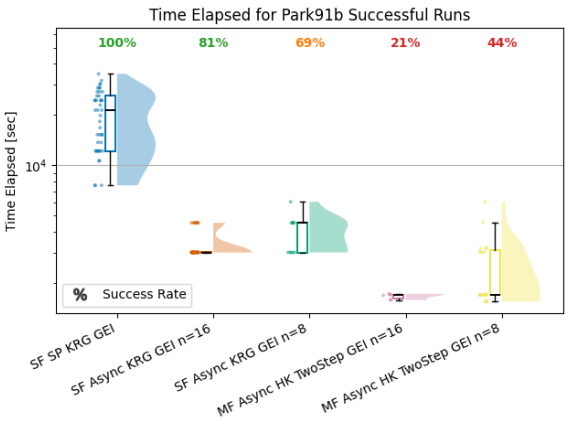
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

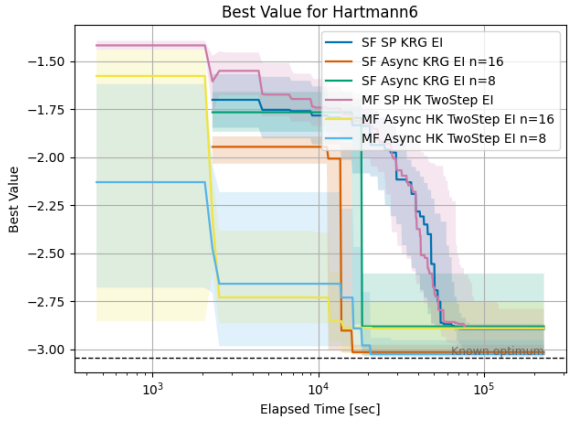


(e) Raincloud plot of the total time elapsed.

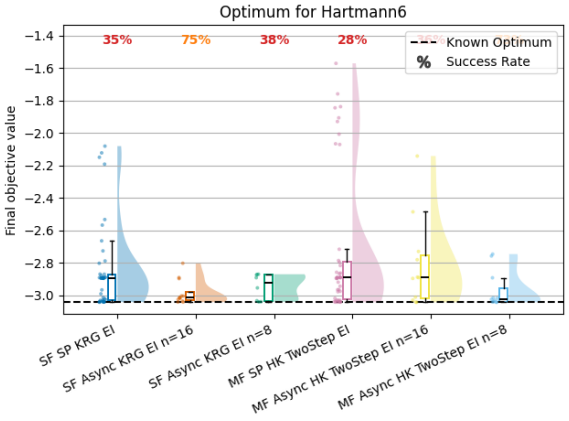


(f) Raincloud plot of the time elapsed in successful runs

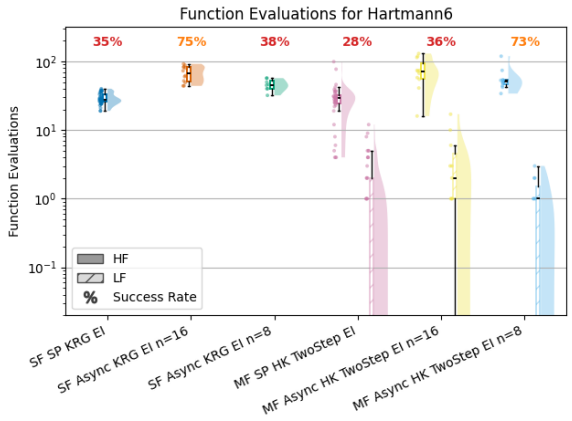
Figure D.18: The resulting plots for the Park91B function using GEI based infill strategies.



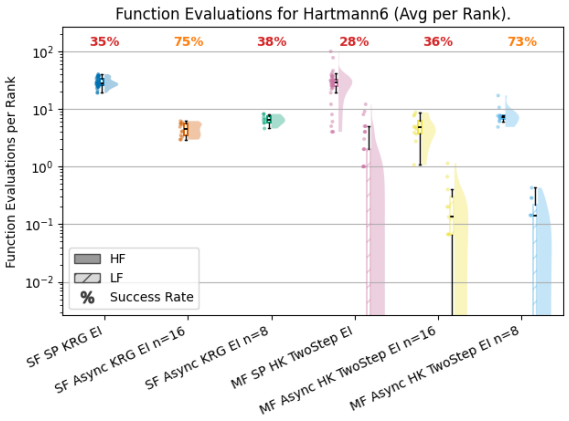
(a) The convergence plot.



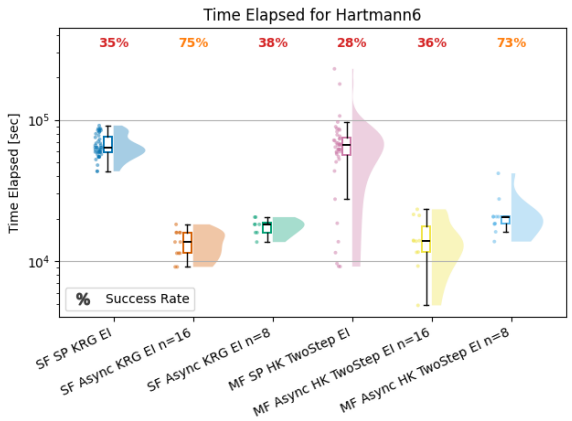
(b) Raincloud plot of the returned optimum.



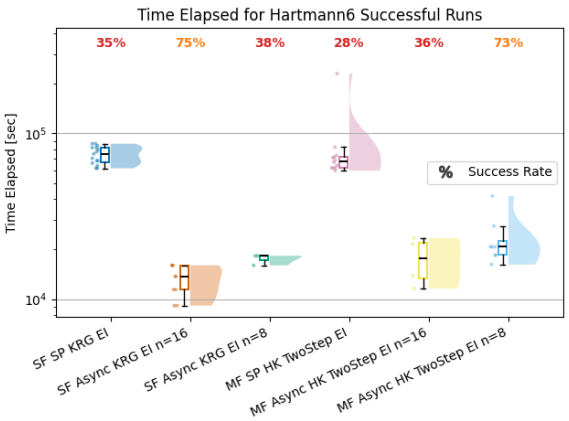
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

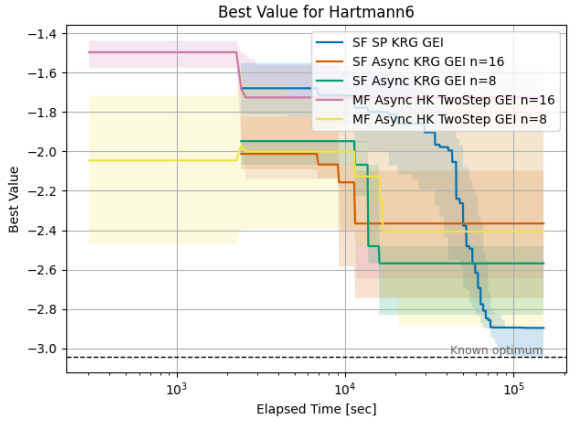


(e) Raincloud plot of the total time elapsed.

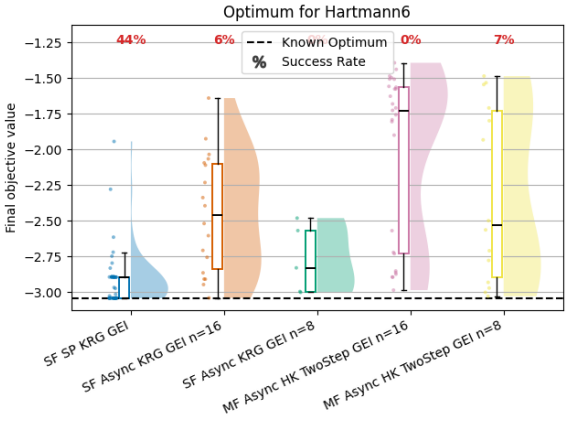


(f) Raincloud plot of the time elapsed in successful runs

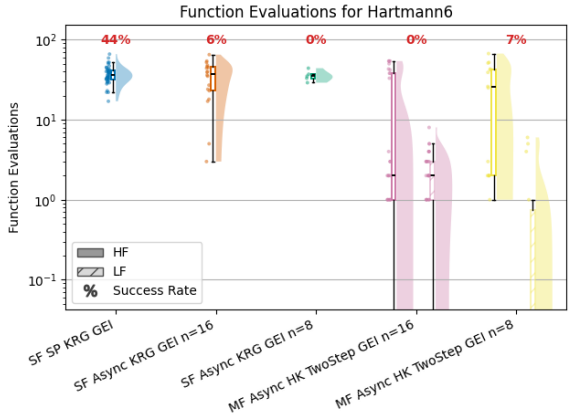
Figure D.19: The resulting plots for the Hartmann6 function using EI based infill strategies.



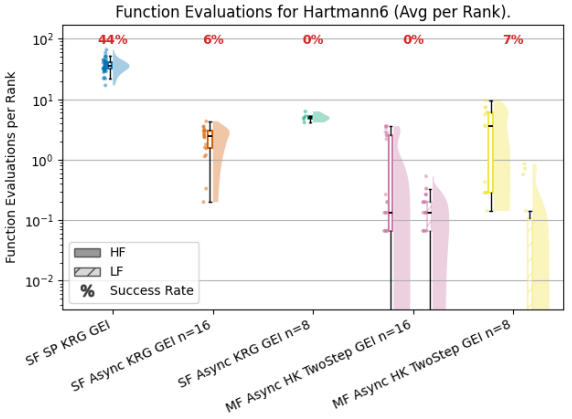
(a) The convergence plot.



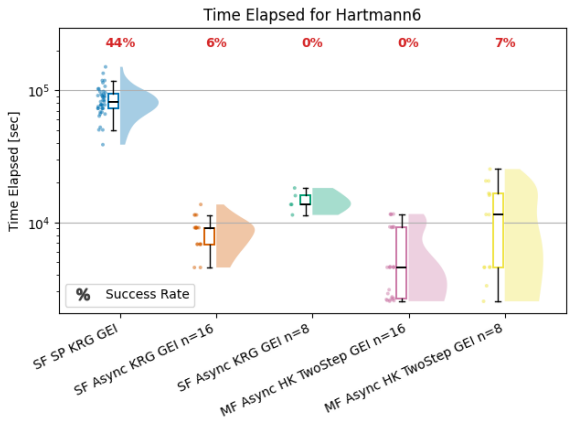
(b) Raincloud plot of the returned optimum.



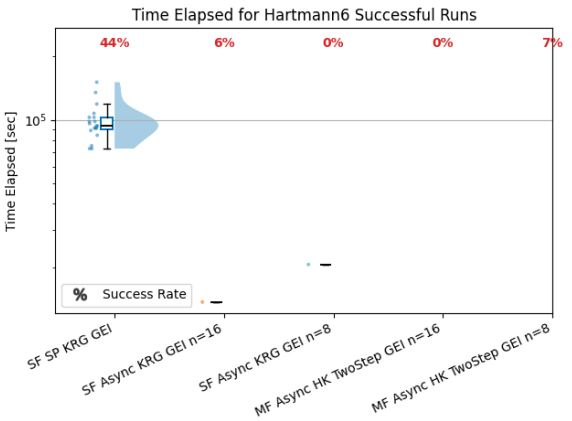
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

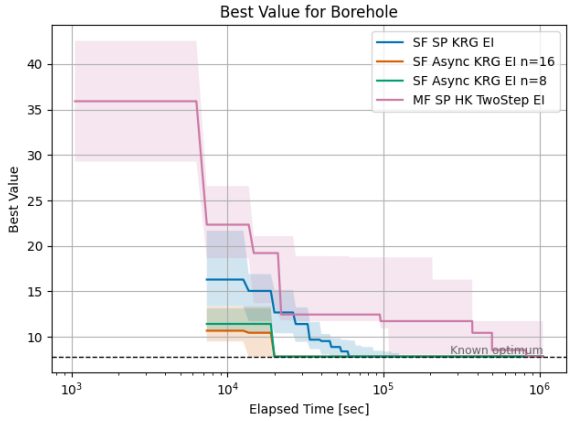


(e) Raincloud plot of the total time elapsed.

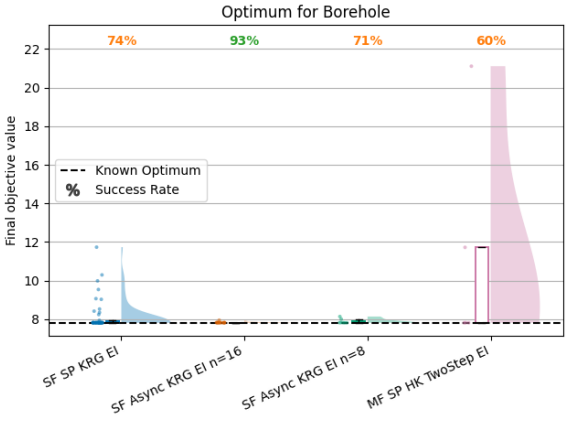


(f) Raincloud plot of the time elapsed in successful runs

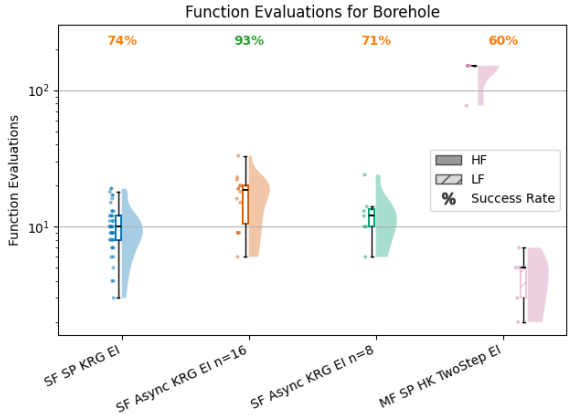
Figure D.20: The resulting plots for the Hartmann6 function using GEI based infill strategies.



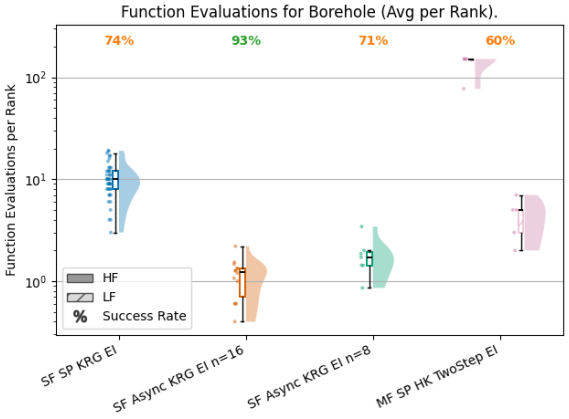
(a) The convergence plot.



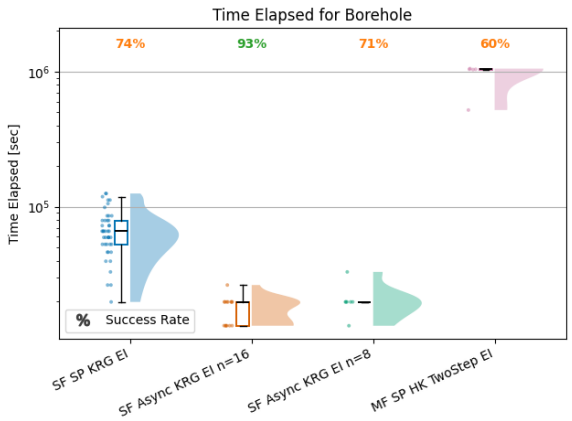
(b) Raincloud plot of the returned optimum.



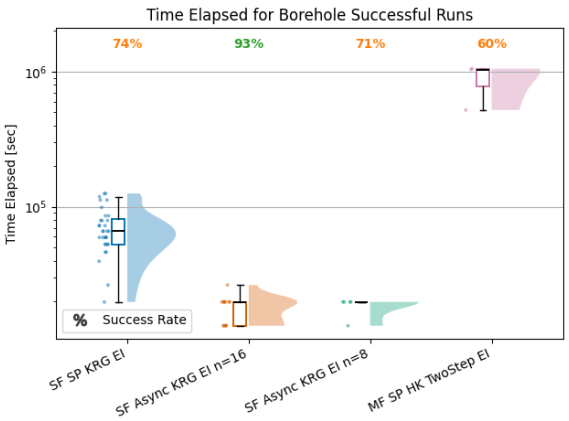
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.

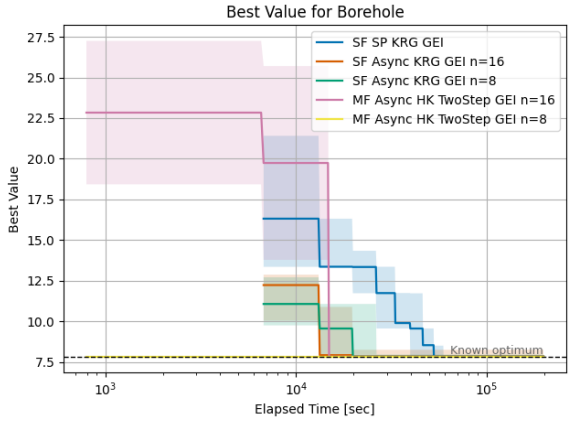


(e) Raincloud plot of the total time elapsed.

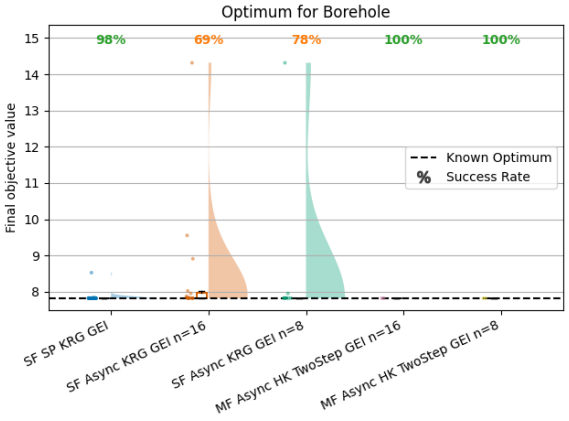


(f) Raincloud plot of the time elapsed in successful runs

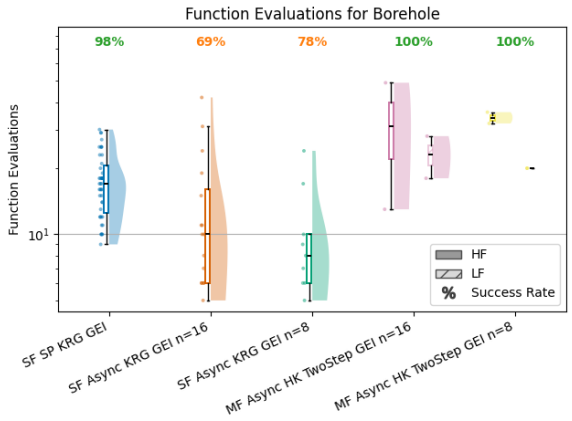
Figure D.21: The resulting plots for the Borehole function using EI based infill strategies.



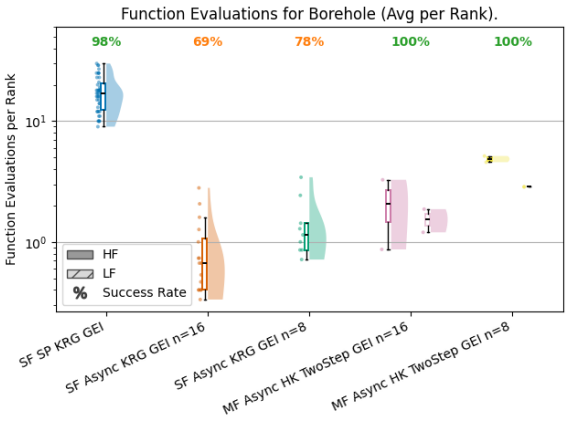
(a) The convergence plot.



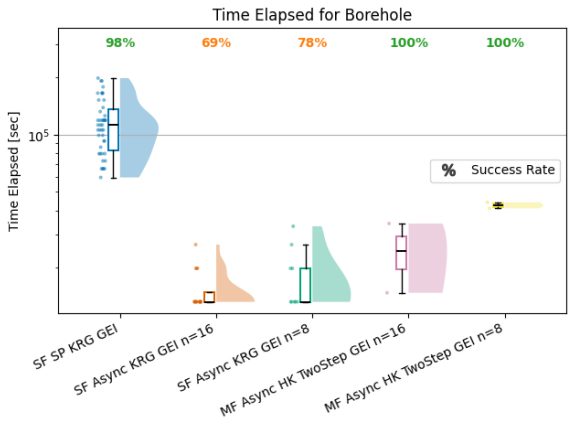
(b) Raincloud plot of the returned optimum.



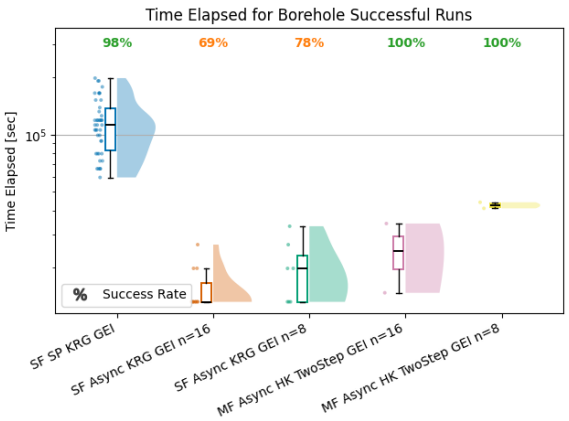
(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the average function calls per rank.



(e) Raincloud plot of the total time elapsed.



(f) Raincloud plot of the time elapsed in successful runs

Figure D.22: The resulting plots for the Borehole function using GEI based infill strategies.

E

Low Fidelity Forrester Function Definitions

This appendix introduces and visualises the Low Fidelity (LF) Forrester functions as examined in chapter 8, starting with the functions with a shift introduced in section E.1, followed by the linear interpolated functions in section E.2.

E.1. Shifted Low Fidelity functions

As discussed in section 8.1, the shifted LF functions are defined using the base Forrester function, with differing parameters for A, B and C. The base Forrester function is defined as follows:

$$f_b(x, A, B, C) = (6(x + A) - 2)^2 (1 + B) \sin(12(x + A) - 4) + C. \tag{E.1}$$

The adaptations introduced in section 8.1 are formulated and illustrated below.

E.1.1. Horizontal Offset 0.5

The horizontal shift has $A = 0.5$, with all other parameters set to 0, this results in;

$$f_b(x, 0.5, 0, 0) = (6(x + 0.5) - 2)^2 \sin(12(x + 0.5) - 4). \tag{E.2}$$

The resulting function is visualised in Figure E.1.

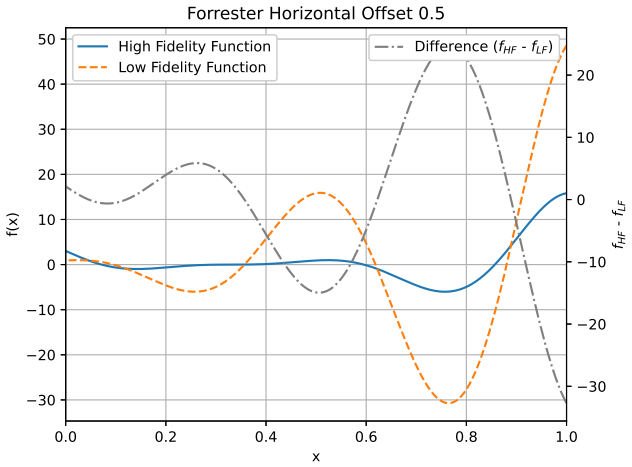


Figure E.1: LF Forrester with $A = 0.5$, $B = 0$ and $C = 0$.

E.1.2. Horizontal Offset -0.5

The opposite shift was investigated, which resulted in:

$$f_b(x, -0.5, 0, 0) = (6(x - 0.5) - 2)^2 \sin(12(x - 0.5) - 4), \tag{E.3}$$

which is visualised in Figure E.2

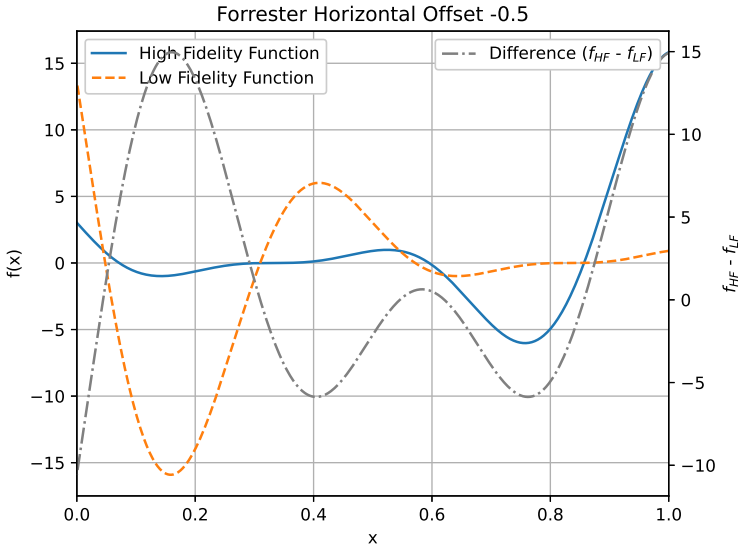


Figure E.2: LF Forrester with $A = -0.5$, $B = 0$ and $C = 0$.

E.1.3. Amplitude Change 5

The last change that was tested was a change in the amplitude of the sin, starting with a positive change of 5, resulting in;

$$f_b(x, 0, 5, 0) = 6(6x - 2)^2 \sin(12x - 4). \tag{E.4}$$

This function is figured in Figure E.3.

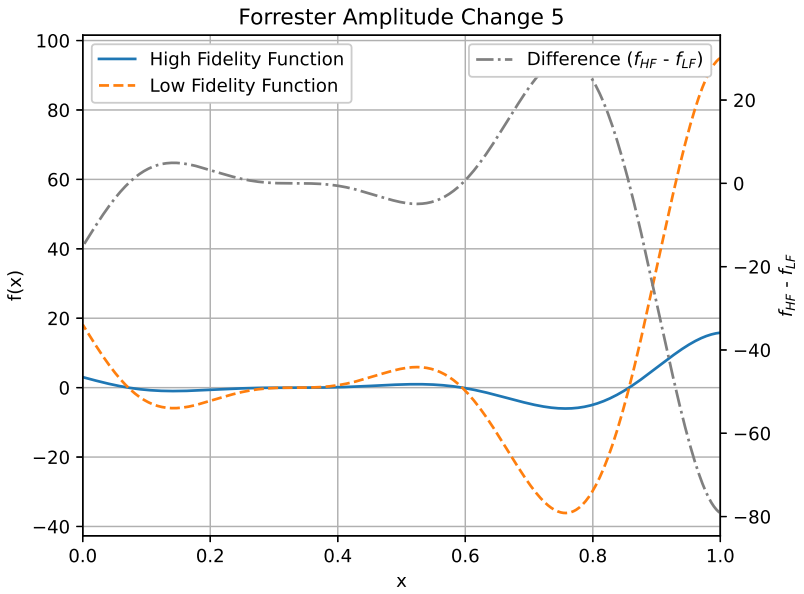


Figure E.3: LF Forrester with $A = 0$, $B = 5$ and $C = 0$.

E.1.4. Amplitude Change -5

Again, the negative change was investigated, resulting in:

$$f_b(x, 0, -5, 0) = -4(6x - 2)^2 \sin(12x - 4). \tag{E.5}$$

Which is pictured in Figure E.4.

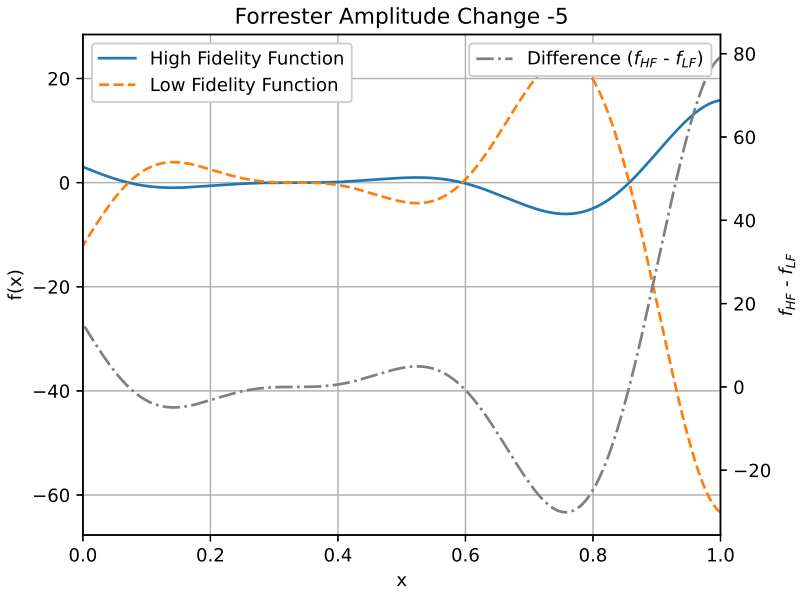


Figure E.4: LF Forrester with $A = 0$, $B = -5$ and $C = 0$.

E.1.5. Vertical Shift 5

The next shift, which was defined, was an upward vertical shift of 5 defined by:

$$f_b(x, 0, 0, 5) = (6x - 2)^2 \sin(12x - 4) + 5. \tag{E.6}$$

This resulted in the function illustrated in Figure E.5.

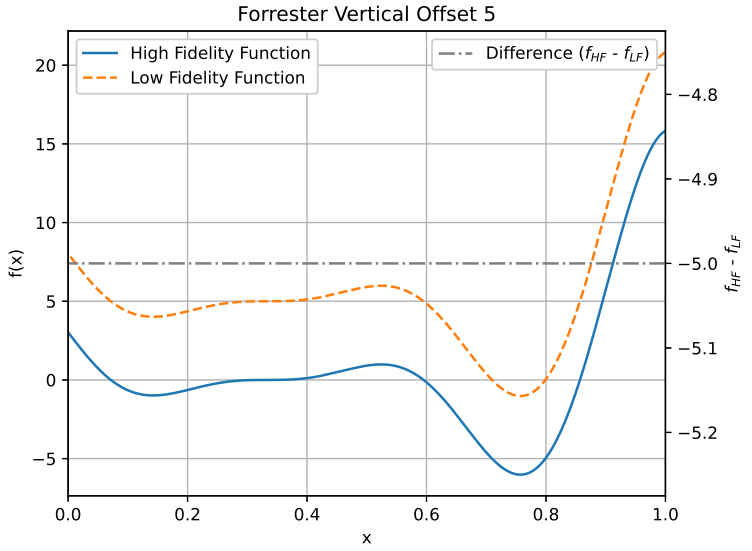


Figure E.5: LF Forrester with $A = 0$, $B = 0$ and $C = 5$.

E.1.6. Vertical Shift -5

The downward shift was also examined, using;

$$f_b(x, 0, 0, -5) = (6x - 2)^2 \sin(12x - 4) - 5. \tag{E.7}$$

Which is pictured in Figure E.6.

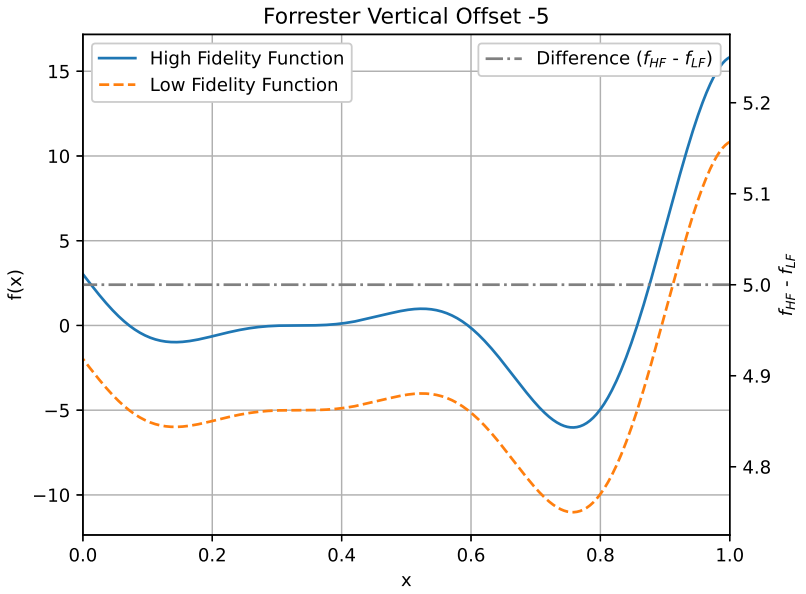


Figure E.6: LF Forrester with $A = 0$, $B = 0$ and $C = -5$.

E.1.7. Combined 5 and 0.5

Lastly, all previously discussed changes were combined in;

$$f_b(x, 0.5, 5, 5) = 6(6(x + 0.5) - 2)^2 \sin(12(x + 0.5) - 4) + 5, \tag{E.8}$$

resulting in the behaviour illustrated in Figure E.7.

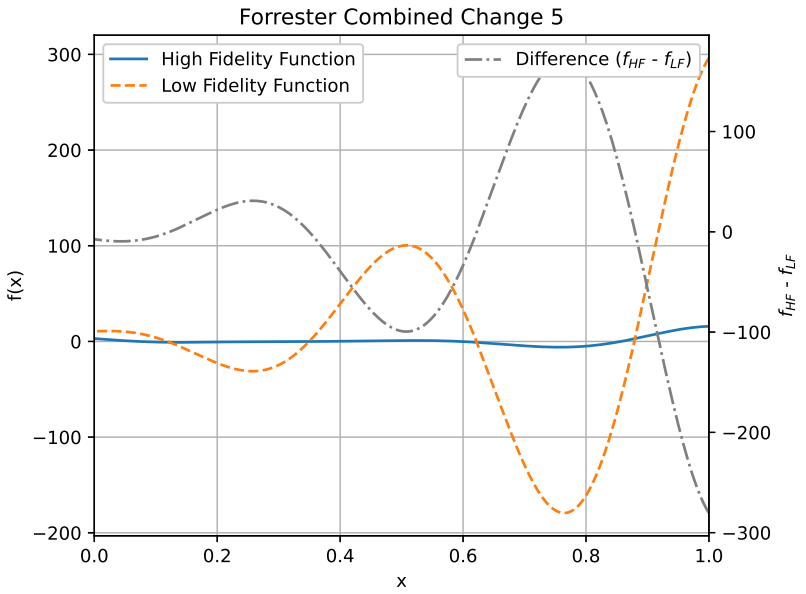


Figure E.7: LF Forrester with $A = 0.5$, $B = 5$ and $C = 5$.

E.1.8. Combined -5 and -0.5

The combination was also made for the negative changes resulting in;

$$f_b(x, -0.5, -5, -5) = -4(6(x - 0.) - 2)^2 \sin(12(x - 0.5) - 4) - 5, \tag{E.9}$$

which is figured in Figure E.8.

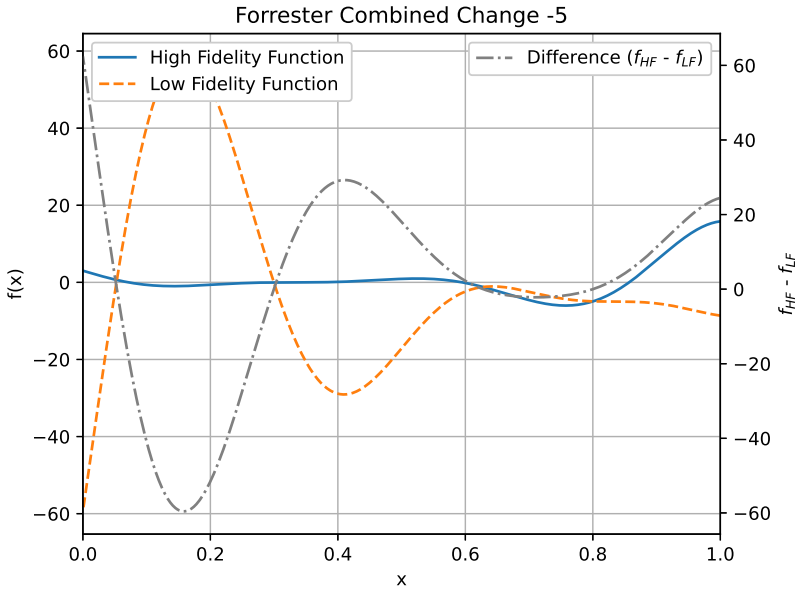


Figure E.8: LF Forrester with $A = -0.5$, $B = -5$ and $C = -5$.

E.2. Linear Interpolation Low Fidelity

The linear Interpolated LF is defined by dividing the design space $[0, 1]$ into n equal-sized bins, where $x_k = \frac{k}{n}$ for $k = 0, 1, \ldots, n$ are the bin edges. Within the k-th bin $[x_{k-1}, x_k]$, the LF function is defined by:

$$f_{\text{LF}}(x) = \frac{f_{\text{HF}}(x_k) - f_{\text{HF}}(x_{k-1})}{x_k - x_{k-1}}(x - x_{k-1}) + f_{\text{HF}}(x_{k-1}), \quad x \in [x_{k-1}, x_k]. \tag{E.10}$$

This results in the visualised functions in Figure E.9 for $n = 2$, Figure E.10 for $n = 5$ and lastly Figure E.11 for $n = 10$.

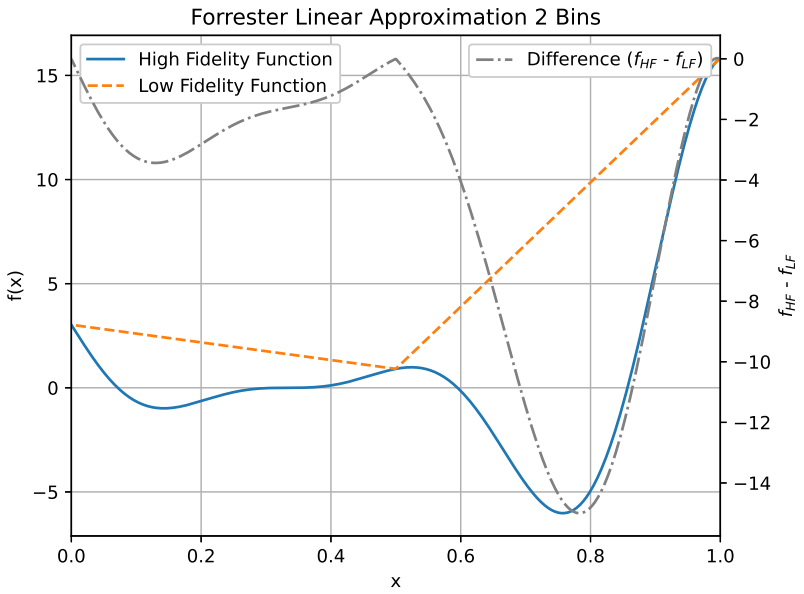


Figure E.9: Linearised LF Forrester Function for $n = 2$.

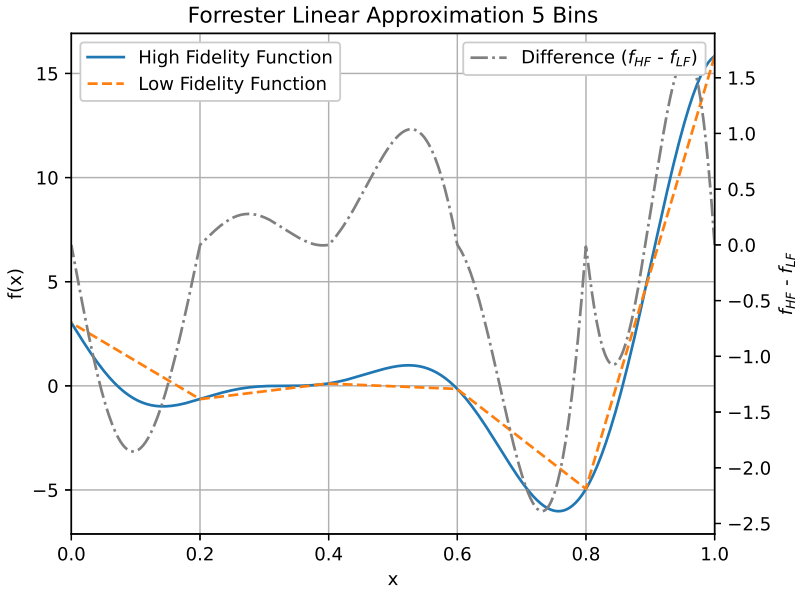


Figure E.10: Linearised LF Forrester Function for $n = 5$.

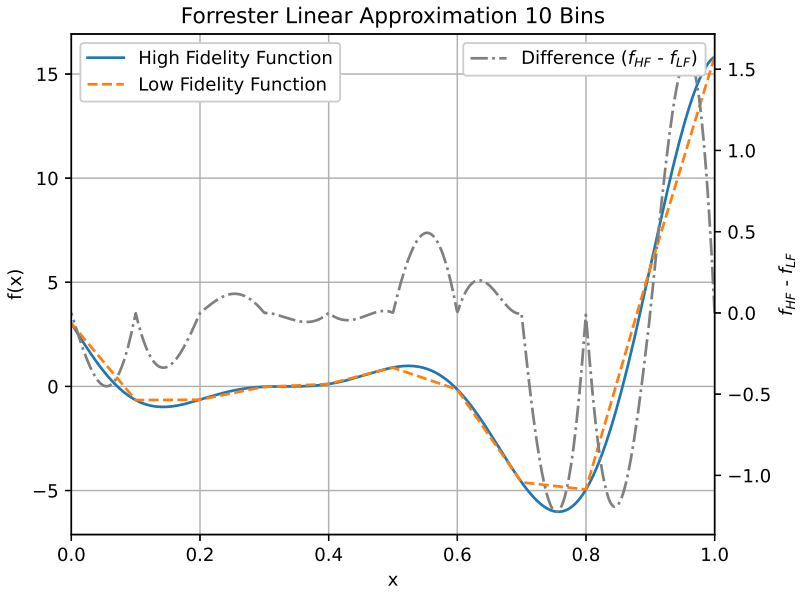


Figure E.11: Linearised LF Forrester Function for $n = 10$.

Low Fidelity Forrester Additional Results

This appendix includes all the results for the Single-Point (SP) Single-Fidelity (SF) Expected Improvement (EI) and SP Multi-Fidelity (MF) EI infill strategies for the customised Low Fidelity (LF) Forrester functions. The function are defined and visualised in section 8.1 and Appendix E. First the full tables of results are presented in section F.1, followed by the visualisation per LF function in section F.2.

F.1. Result Tables

The first table, Table F.1, tabulates the success rate per function - infill strategy combination. The colour coding in this table is as follows:

- $100 - 80\%$: Green
- $79 - 50\%$: Orange
- $49 - 1\%$: Red
- 0%: Dark red

The next table in Table F.2 presents the mean elapsed time per run $\pm$ the standard deviation. The same colour coding is used to signify the success rate. Lastly, in Table F.3, the mean elapsed time per successful run, $\pm$ the standard deviation, is tabulated.

Table F.1: Success rates per (function, algorithm) for the different LF Forrester functions. Note that the results for the SF SP infill strategy are equal, since this is solely using the HF function.

Test function	SF SP KRG EI	MF SP HK TwoStep EI
Forrester	100%	95%
Forrester horizontal offset 0.5	100%	86%
Forrester horizontal offset -0.5	100%	95%
Forrester amplitude change 5	100%	74%
Forrester amplitude change -5	100%	77%
Forrester vertical offset 5	100%	77%
Forrester vertical offset -5	100%	77%
Forrester combined change 5	100%	79%
Forrester combined change -5	100%	93%
Forrester linear approximation 2 bins	100%	93%
Forrester linear approximation 5 bins	100%	91%
Forrester linear approximation 10 bins	100%	79%

Table F.2: Mean elapsed time ± std, coloured by success rate (values are tabulated in Table F.1) for the LF Forrester functions. Note that the results for the SF SP infill strategy are equal, since this is solely using the HF function.

Test function	SF SP KRG EI	MF SP HK TwoStep EI
Forrester	00:13:49 ± 00:05:31	00:13:59 ± 00:03:34
Forrester horizontal offset 0.5	00:13:49 ± 00:05:31	00:12:51 ± 00:04:20
Forrester horizontal offset -0.5	00:13:49 ± 00:05:31	00:13:56 ± 00:03:30
Forrester amplitude change 5	00:13:49 ± 00:05:31	00:11:38 ± 00:04:40
Forrester amplitude change -5	00:13:49 ± 00:05:31	00:11:31 ± 00:04:37
Forrester vertical offset 5	00:13:49 ± 00:05:31	00:11:36 ± 00:04:40
Forrester vertical offset -5	00:13:49 ± 00:05:31	00:11:59 ± 00:04:40
Forrester combined change 5	00:13:49 ± 00:05:31	00:12:01 ± 00:04:40
Forrester combined change -5	00:13:49 ± 00:05:31	00:14:03 ± 00:03:33
Forrester linear approximation 2 bins	00:13:49 ± 00:05:31	00:14:27 ± 00:03:10
Forrester linear approximation 5 bins	00:13:49 ± 00:05:31	00:13:02 ± 00:04:30
Forrester linear approximation 10 bins	00:13:49 ± 00:05:31	00:11:37 ± 00:04:44

Table F.3: Mean elapsed time ± std for the successful runs for LF Forrester functions, coloured by success rate (values are tabulated in Table F.1). Note that the results for the SF SP infill strategy are equal, since this is solely using the HF function.

Test function	SF SP KRG EI	MF SP HK TwoStep EI
Forrester	00:13:49 ± 00:05:31	00:14:28 ± 00:02:52
Forrester horizontal offset 0.5	00:13:49 ± 00:05:31	00:14:17 ± 00:02:40
Forrester horizontal offset -0.5	00:13:49 ± 00:05:31	00:14:28 ± 00:02:35
Forrester amplitude change 5	00:13:49 ± 00:05:31	00:13:59 ± 00:02:36
Forrester amplitude change -5	00:13:49 ± 00:05:31	00:13:39 ± 00:02:47
Forrester vertical offset 5	00:13:49 ± 00:05:31	00:13:44 ± 00:02:48
Forrester vertical offset -5	00:13:49 ± 00:05:31	00:14:08 ± 00:02:48
Forrester combined change 5	00:13:49 ± 00:05:31	00:14:04 ± 00:02:42
Forrester combined change -5	00:13:49 ± 00:05:31	00:14:49 ± 00:02:18
Forrester linear approximation 2 bins	00:13:49 ± 00:05:31	00:14:59 ± 00:02:36

Continued on next page

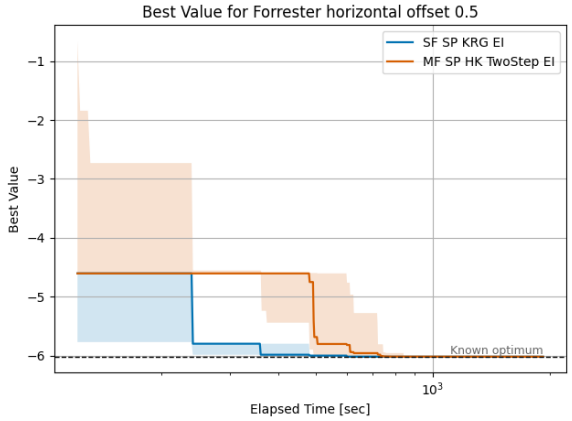
Table F.3 – continued from previous page

Test function	SF SP KRG EI	MF SP HK TwoStep EI
Forrester linear approximation 5 bins	00:13:49 ± 00:05:31	00:14:02 ± 00:03:20
Forrester linear approximation 10 bins	00:13:49 ± 00:05:31	00:13:33 ± 00:03:05

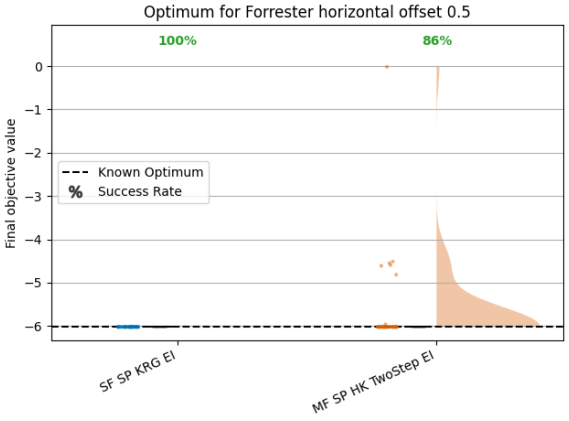
F.2. Result Visualisation

The convergence plot and raincloud plots for the resulting optimum, total function evaluations, time elapsed and time elapsed per successful run are included in this section. The plots for each function are included in:

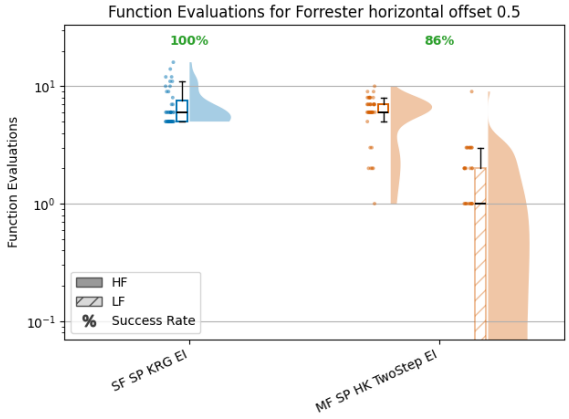
- ***Horizontal offset 0.5***: Figure F.1
- ***Horizontal offset -0.5***: Figure F.2
- ***Amplitude change 5***: Figure F.3
- ***Amplitude change -5***: Figure F.4
- ***Vertical offset 5***: Figure F.5
- ***vertical offset 5***: Figure F.6
- ***Combined change 5***: Figure F.7
- ***Combined change -5***: Figure F.8
- ***Linear approximation 2 bins***: Figure F.9
- ***Linear approximation 5 bins***: Figure F.10
- ***Linear approximation 10 bins***: Figure F.11



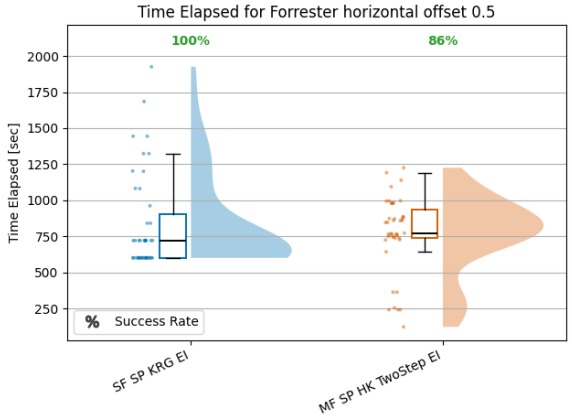
(a) The convergence plot.



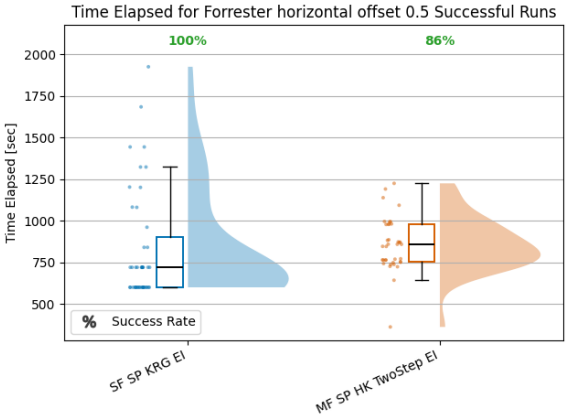
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

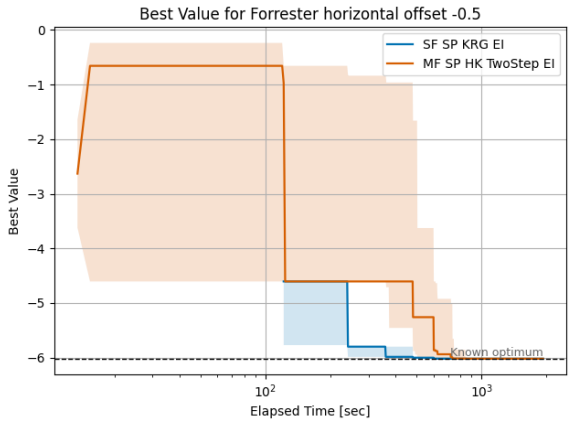


(d) Raincloud plot of the total time elapsed.

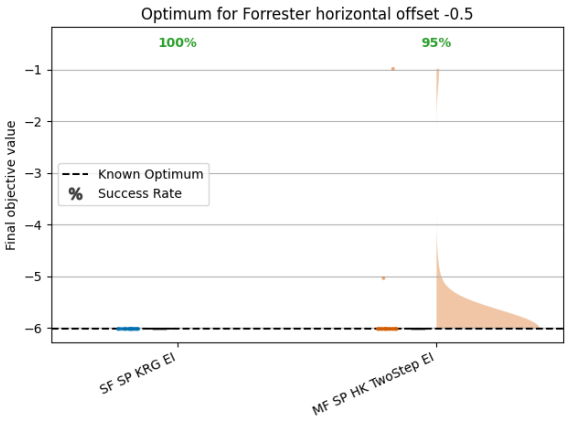


(e) Raincloud plot of the time elapsed in successful runs

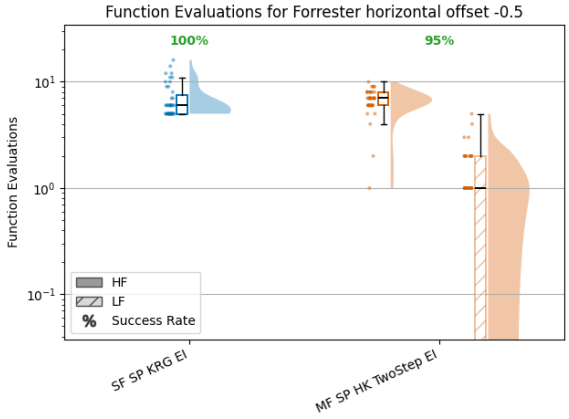
Figure F.1: The resulting plots for the horizontal offset 0.5 LF function.



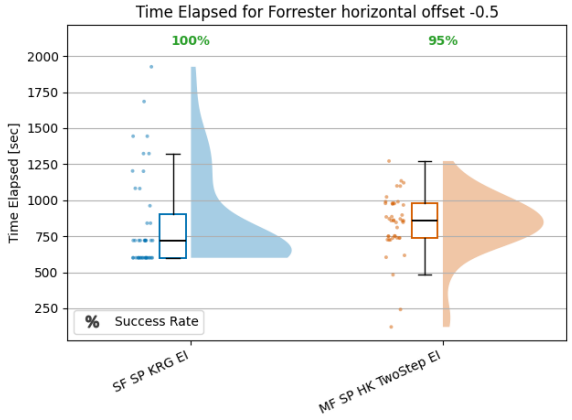
(a) The convergence plot.



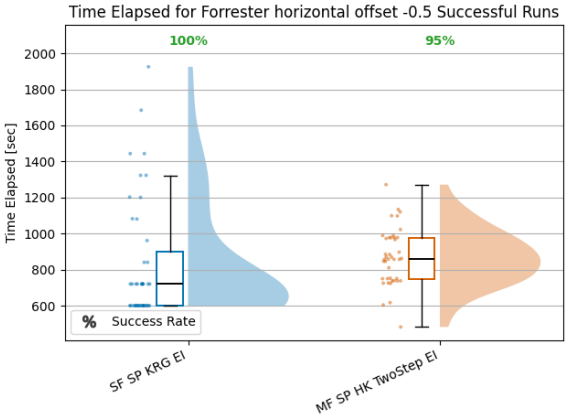
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

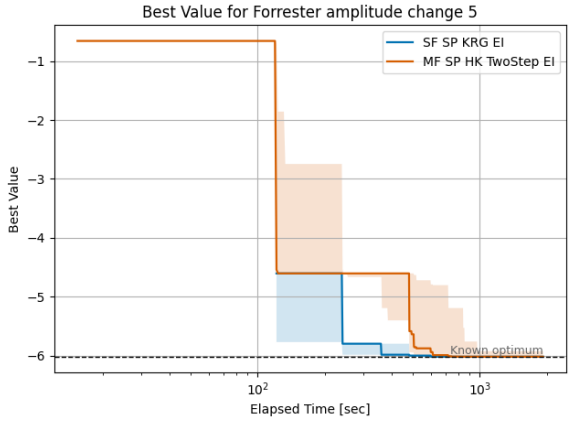


(d) Raincloud plot of the total time elapsed.

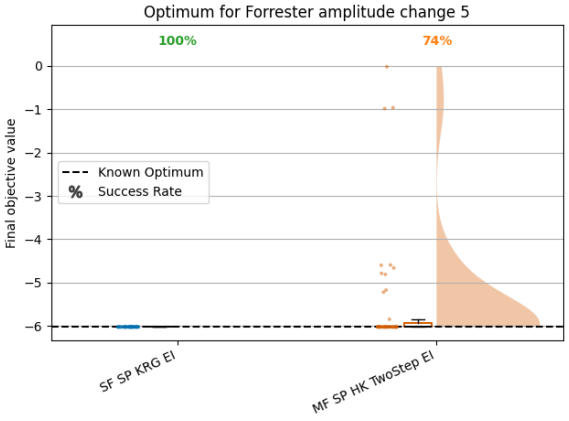


(e) Raincloud plot of the time elapsed in successful runs

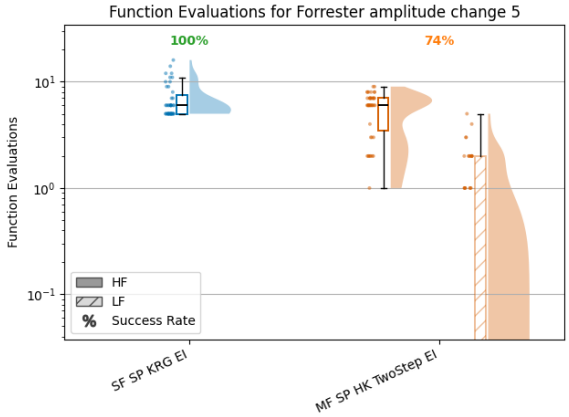
Figure F.2: The resulting plots for the horizontal offset -0.5 LF function.



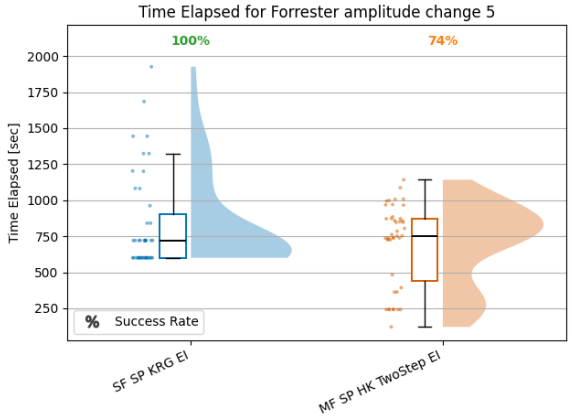
(a) The convergence plot.



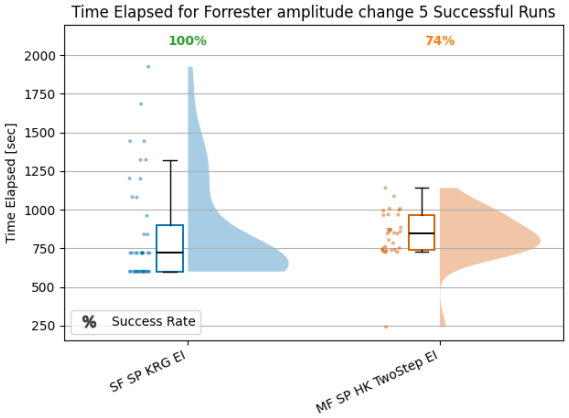
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

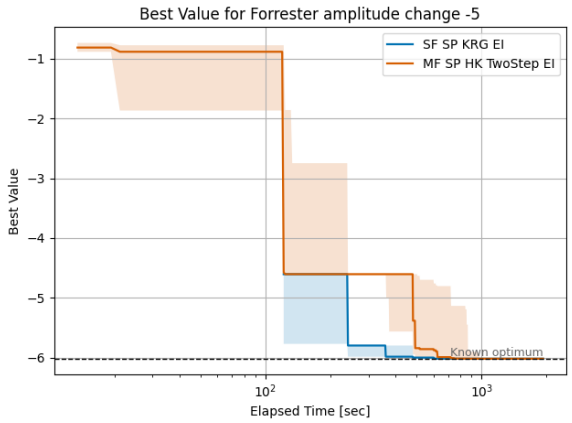


(d) Raincloud plot of the total time elapsed.

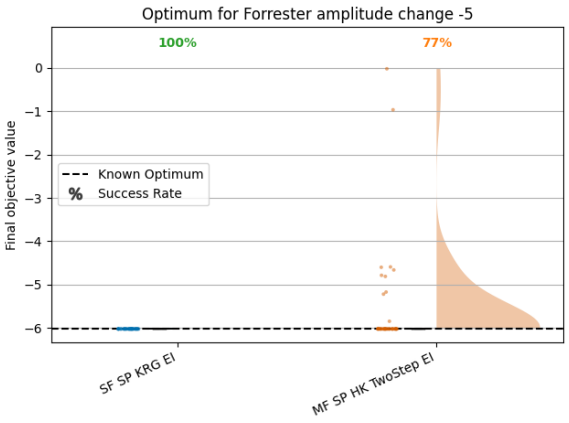


(e) Raincloud plot of the time elapsed in successful runs

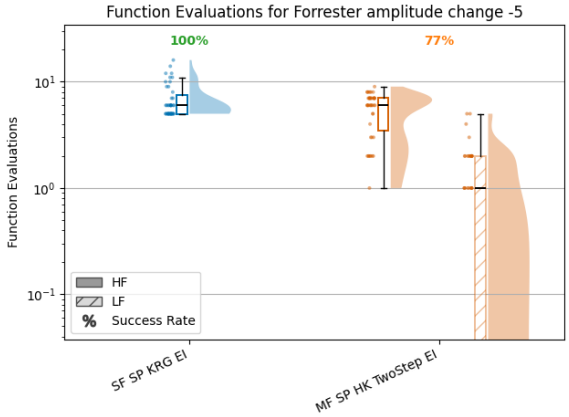
Figure F.3: The resulting plots for the amplitude change 5 LF function.



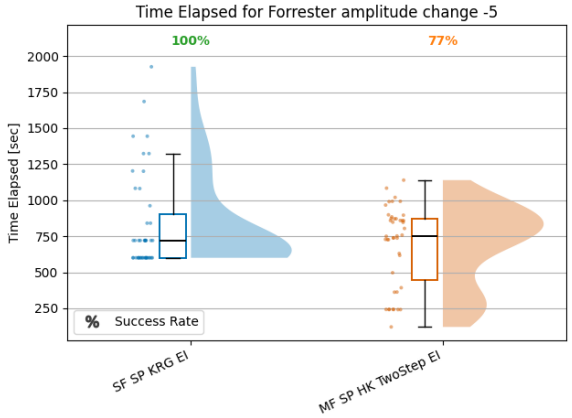
(a) The convergence plot.



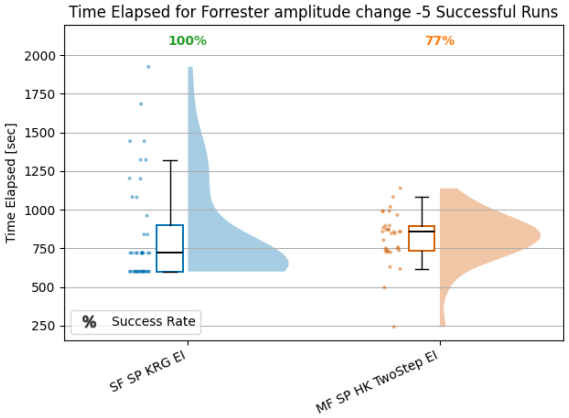
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

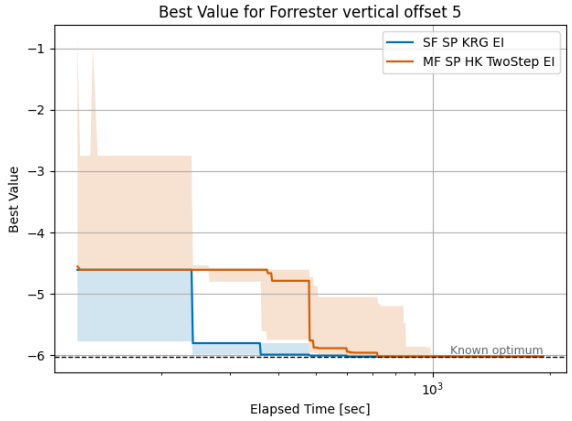


(d) Raincloud plot of the total time elapsed.

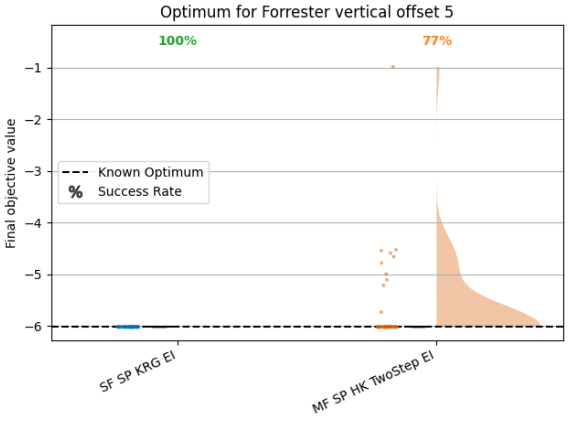


(e) Raincloud plot of the time elapsed in successful runs

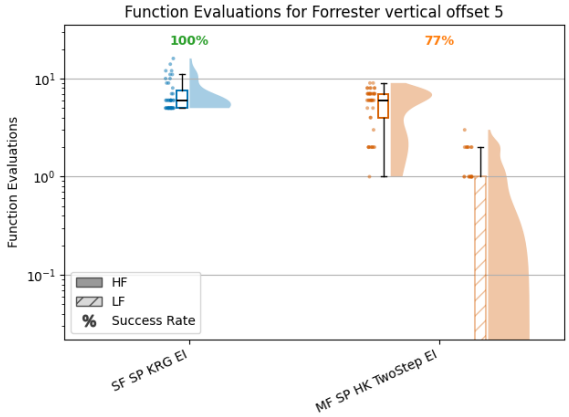
Figure F.4: The resulting plots for the amplitude change -5 LF function.



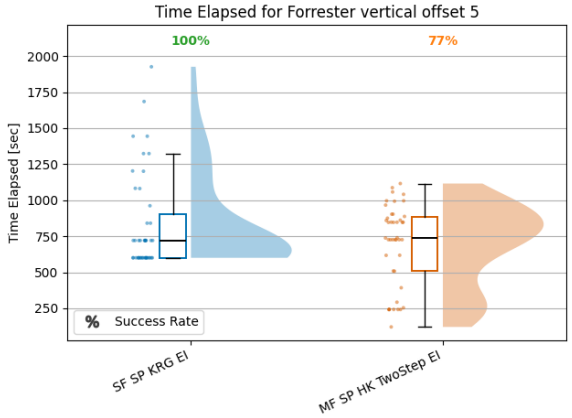
(a) The convergence plot.



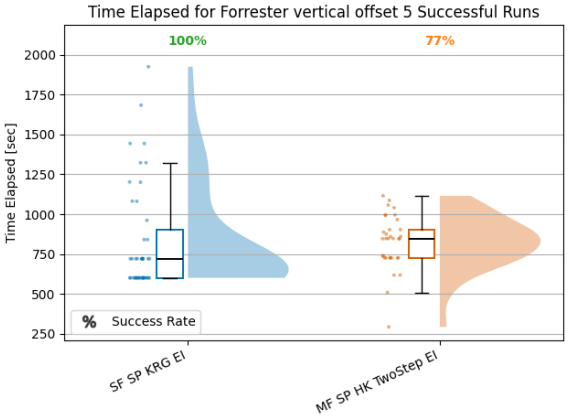
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

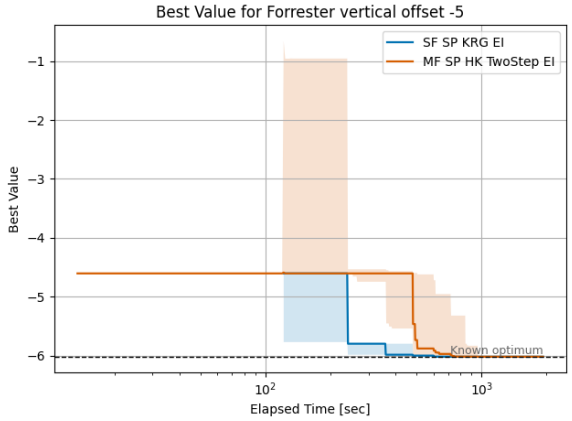


(d) Raincloud plot of the total time elapsed.

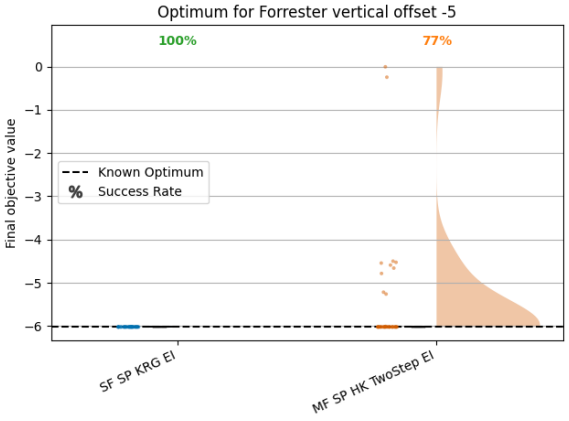


(e) Raincloud plot of the time elapsed in successful runs

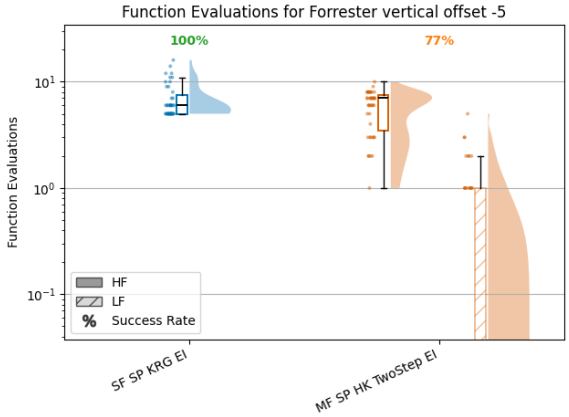
Figure F.5: The resulting plots for the vertical offset 5 LF function.



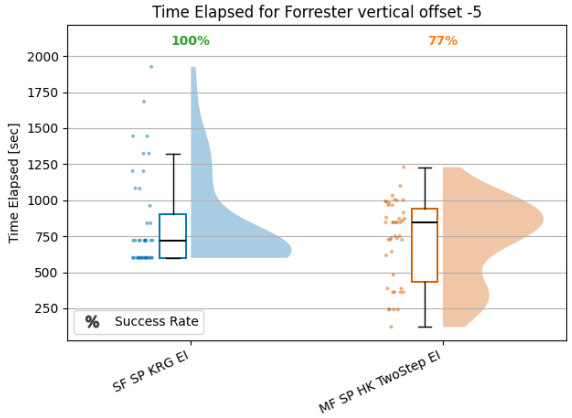
(a) The convergence plot.



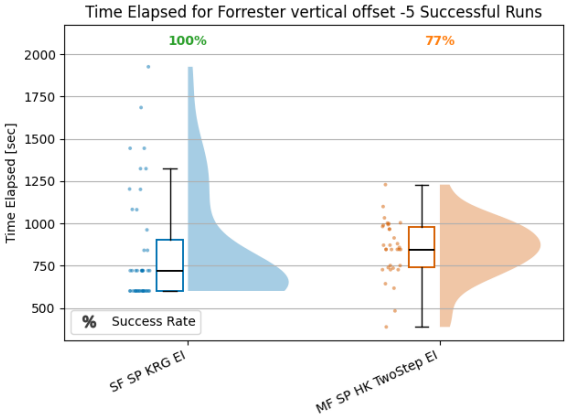
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

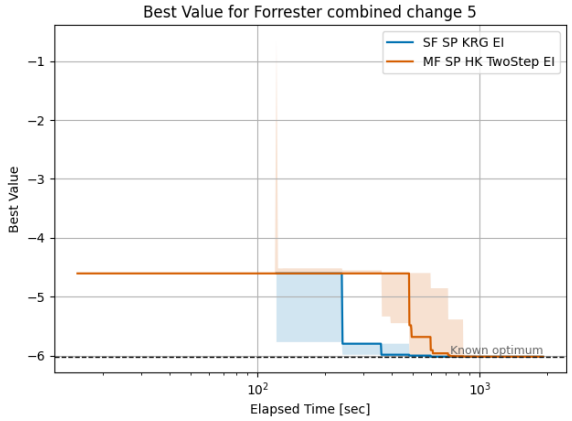


(d) Raincloud plot of the total time elapsed.

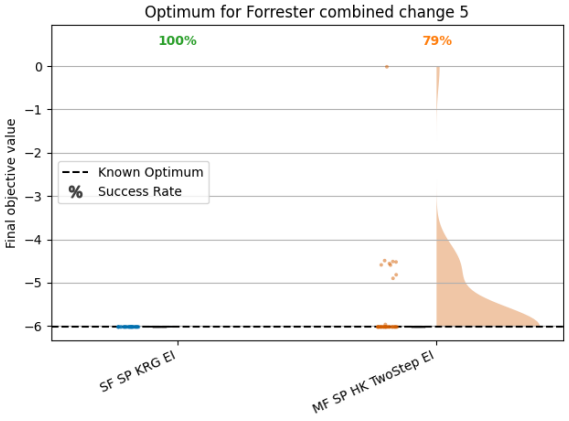


(e) Raincloud plot of the time elapsed in successful runs

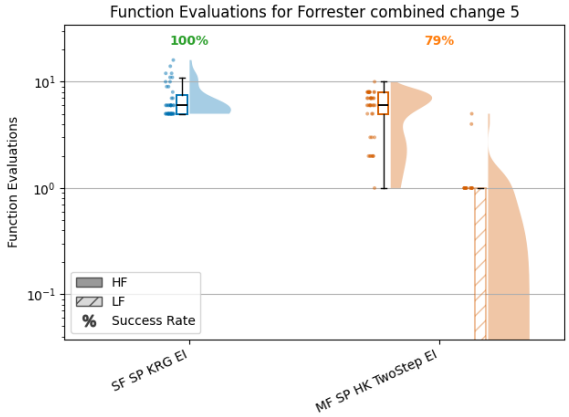
Figure F.6: The resulting plots for the vertical offset -5 LF function.



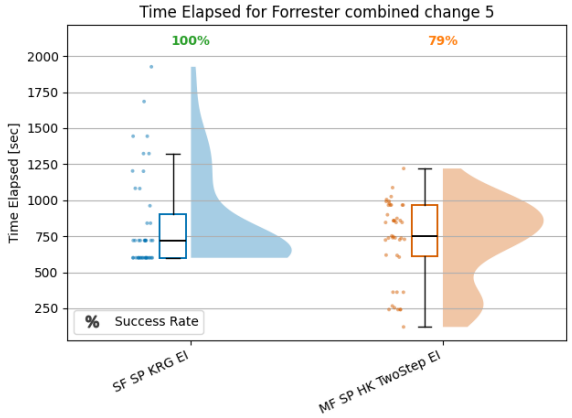
(a) The convergence plot.



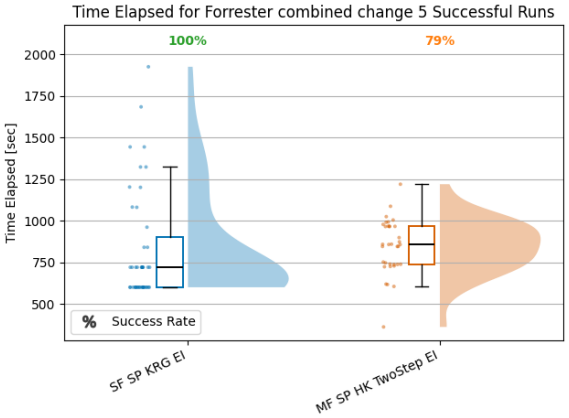
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

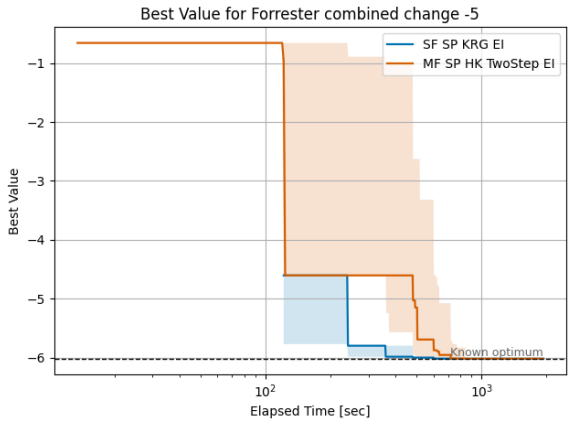


(d) Raincloud plot of the total time elapsed.

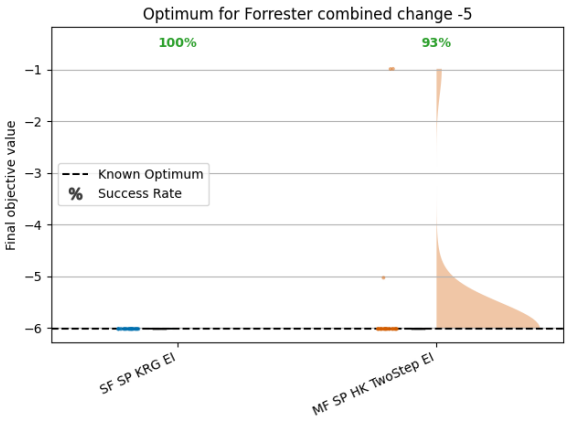


(e) Raincloud plot of the time elapsed in successful runs

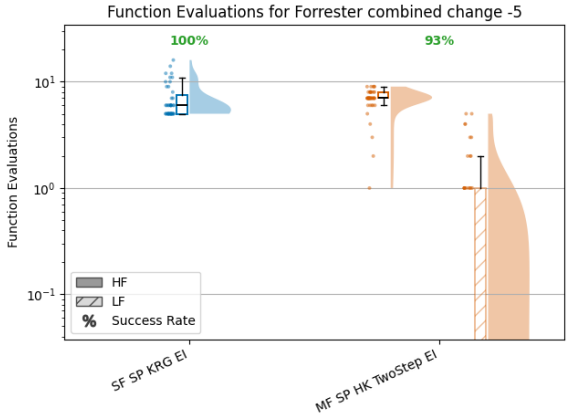
Figure F.7: The resulting plots for the combined change 5 LF function.



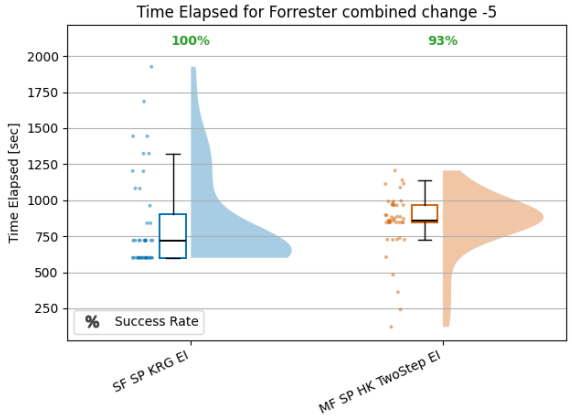
(a) The convergence plot.



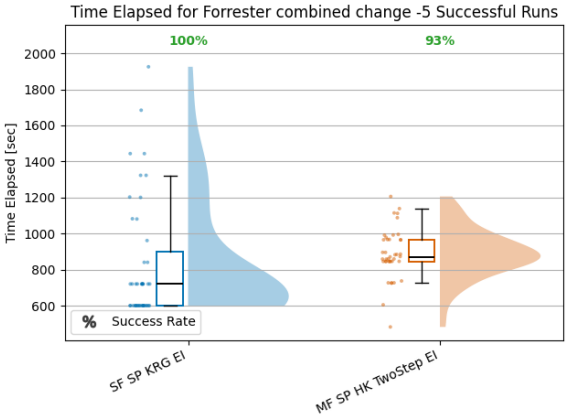
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

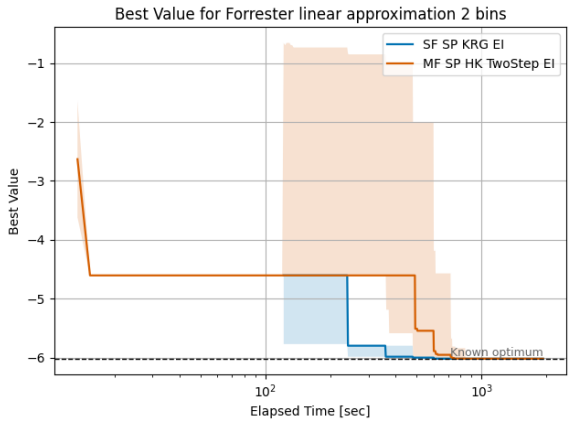


(d) Raincloud plot of the total time elapsed.

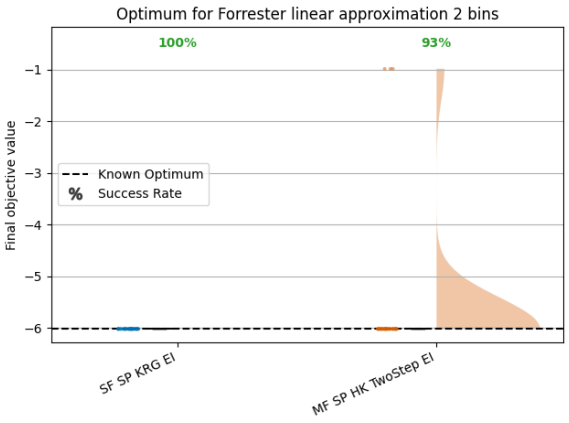


(e) Raincloud plot of the time elapsed in successful runs

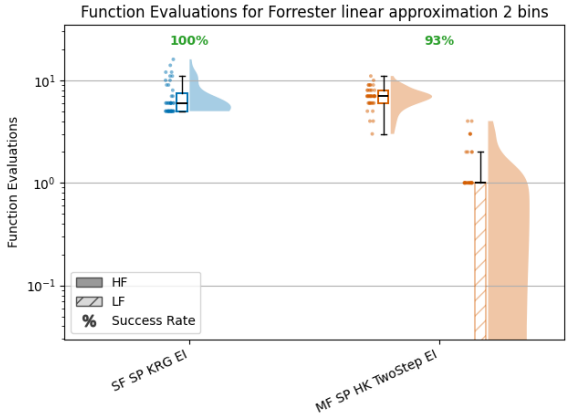
Figure F.8: The resulting plots for the combined change -5 LF function.



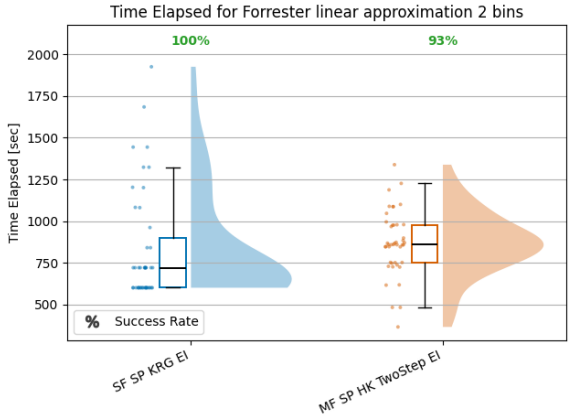
(a) The convergence plot.



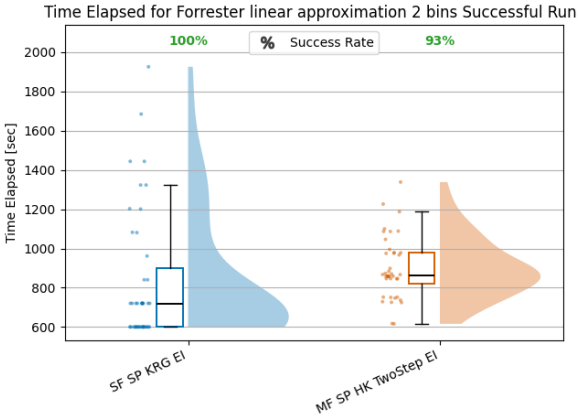
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

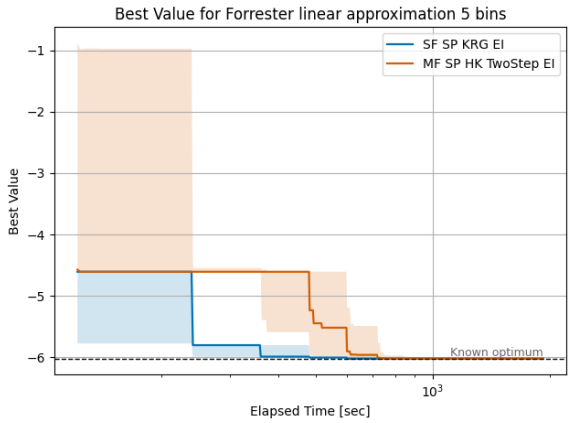


(d) Raincloud plot of the total time elapsed.

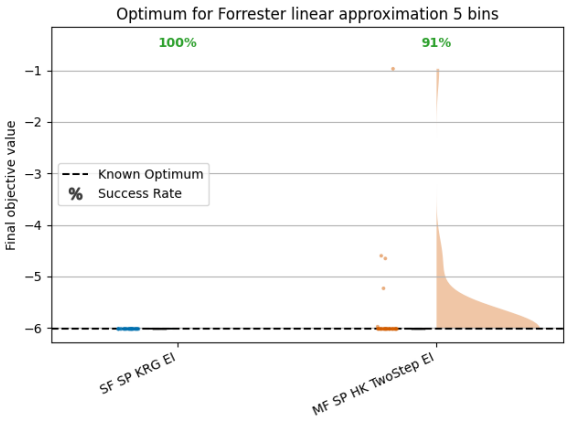


(e) Raincloud plot of the time elapsed in successful runs

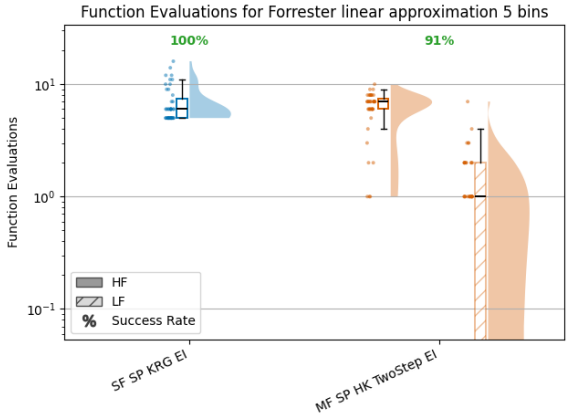
Figure F.9: The resulting plots for the linear approximation 2 bins LF function.



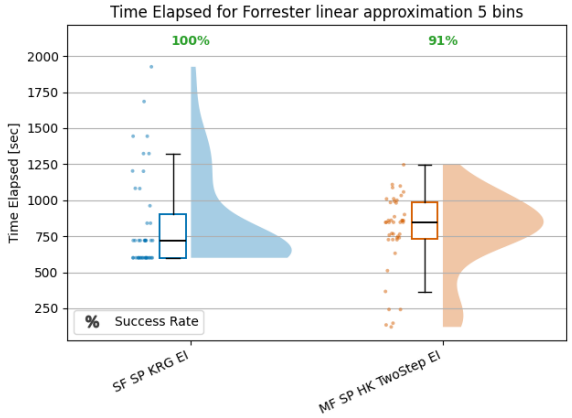
(a) The convergence plot.



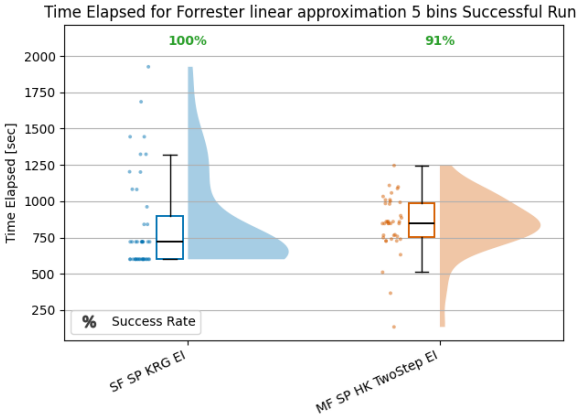
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.

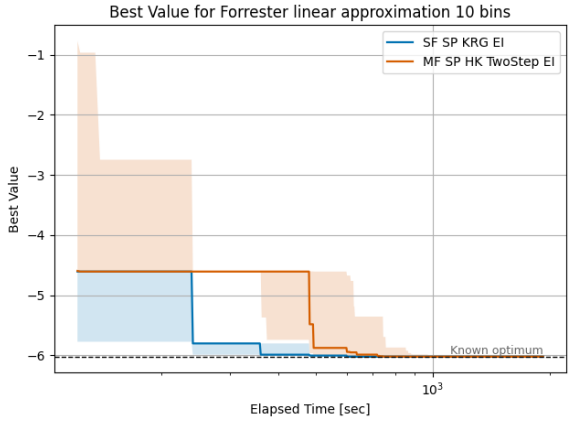


(d) Raincloud plot of the total time elapsed.

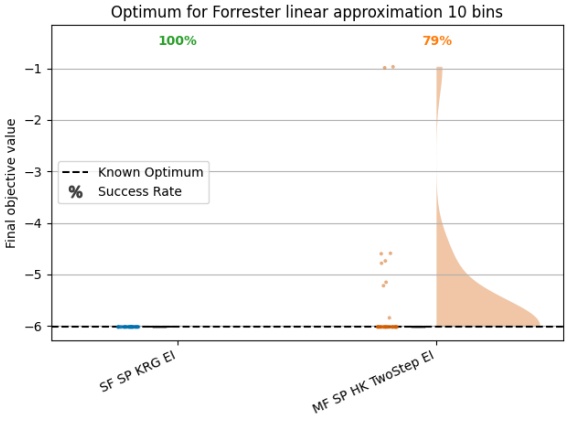


(e) Raincloud plot of the time elapsed in successful runs

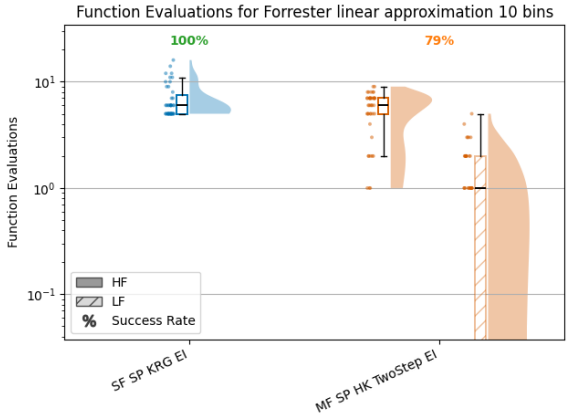
Figure F.10: The resulting plots for the linear approximation 5 bins LF function.



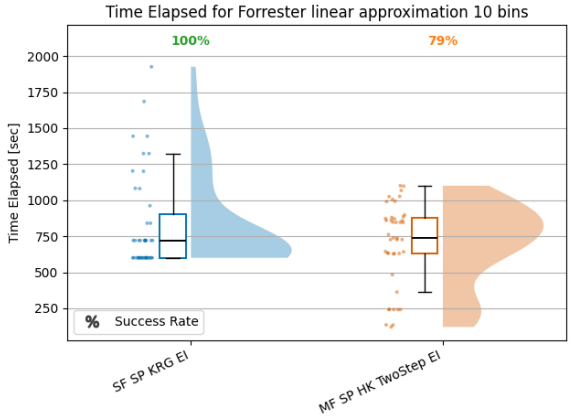
(a) The convergence plot.



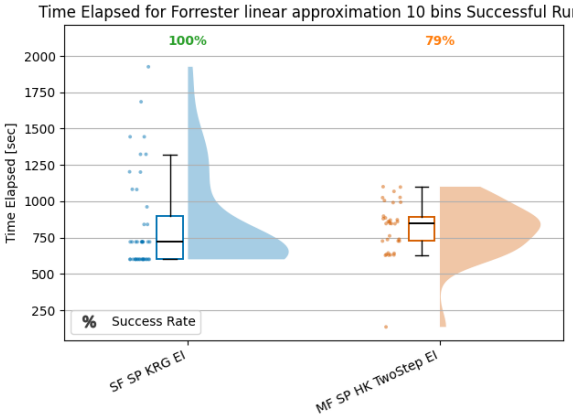
(b) Raincloud plot of the returned optimum.



(c) Raincloud plot of the total function calls.



(d) Raincloud plot of the total time elapsed.



(e) Raincloud plot of the time elapsed in successful runs

Figure F.11: The resulting plots for the linear approximation 10 bins LF function.